货

huò

https://scale.qihardware.org

2019 . Week 7 . Feb 17 - Feb 23

This page left intentionally blank
to power your imagination
of what interesting art, ads,
sponsorship, standard
frontmatter or blank space
should be included in
future editions of scale.

货 huò

Meaning 'goods', the word is composed by the cowry (bei 贝貝 bèi), an ancient Chinese money, and the character (hua 化) which means 'to transform'.

货 / huo is a verycommon character in Chinese language, which can mean 'goods' or 'commodity'. In the daily life, 货 represents any outcome of work, in manufacturing but also services and products. '干货' ganhuo, literally to 'make goods' is a popular way of saying 'to work' (official documents have even adopted, https://www.zhihu.com/question/22070909, the term).

For Chinese new year, people will stock their home with 年货 (nian huo) litteraly 'yearly goods' : fruits, vegetables, sunflower seeds and other delicacies for the celebrations. Beyond the simple act of producing, 货 is what is transformed by people to make life.

We have human rights and animal rights. Should there be computer rights?

Clément Renaud, Researcher
Answered 5d ago · Author has **302** answers and **713k** answer views

Interesting question. Let's first consider the rights of nonhuman beings, before going into the case of computers.

Animal rights' main goal is to prevent cruelty. In many religions, animals are considered sentient beings. The Shariah as well as religious laws from Indhuists, Buddhists and most Christians states since 18th century offers some form of protection for animals. Apart from protection, another legal situation arises when the plaintif is a human and an animal has to go on trial. During the middle ages in Europe, animals were subject to trials and judgement. There are many documented cases of pigs or cows that were condemned for hurting humans after being audited during a trial[1].

Most orignal sources we have of these trials were compiled by writers from 19th century willing to show the absurdity of medieval laws. For them, human rights could exist as opposed to animals, as they refuse to recognise first that some animals possess moral agency[2] in a rational world under human monopoly.

Back to the original question : should computers have rights? Computers are usually considered tools. Therefore their owner is liable for any incidents or possible legal outcomes related to its actions. The problems arise when, with some degree of self-determination, a computer perform a task without having being directly ordered to do so.

Here, the famous case of the monkey selfie dispute[3] provides a good precedent. The photographer claims ownership of the pic because he did setup cameras around macaques while the animal rights defendants claim that the creator is the monkey itself, a "non-legal" person, and the pic should therefore be in public domain.

To reframe our question a bit, we could ask : could computers be recognised as legal entities? There are already many non-human legal entities, such as companies, state institutions or even rivers[4]. In US law, corporations are legal persons with limited legal rights. Their shareholders' liability is generally limited to their investment. To recognise computers as legal entities may require a similar approach, which raise multiple questions : what are computers rights and obligations? What are their legal relationships with their programmers or manufacturers? How are cases settled?

One of the main problem here is that we don't have a proper legal infrastructure (court, jurisprudence, etc) to assess possible cases that may arise by the actions of computers, algorithms, robots, etc. In common discourse, private algorithms are often considered as "black boxes" i.e. impossible to decipher and therefore evaluate. Recently, researchers have raised against this "fallacy of inscrutability"[5] of computers and argued that we may not need to understand all the inner workings of a machine or computer to evaluate its actions. We can rely on a legality of its outcomes to judge it. After all, a human killer is a black box as well as there are no way to know its motives for sure.

> " we have many ways to evaluate an algorithm after its outcomes. We can know it in depth and make many reliable predictions just by analyzing its outputs. This is not free, it comes at an additional cost to developing the algorithm itself, but it does not require to understand how it works, how it thinks. "[6]

For this to happen, we need to recognise first that computers have some form of agency of their own, which opens a whole new world of questions in the domain of computer ethics. Fortunately, this is not a new field of research. The venerable Norbert Wiener himself has an entire book[7] on the topic.

Beyond computers, this question of responsibility could be extended to all non-humans, and therefore benefit for the large body of reflection leads by ecologists in the fight against pollution and climate change. The idea of law itself has evolved with millenaries of writing

systems to consider new entities, living beings and abstractions. Enforcement as well has transformed radically. The People Republic of China has turn digital technologies into a core social and political infrastructure, and is experimenting with new forms of prescriptive computer regulations[8]. The Chinese “social credit system” shows how a legal system originally designed for industrial pollution can be extended to other domains such as financial and civil regulation.

Another recent example is the crypto economy that has defined a mission of rebuilding a new financial system for our planet. The whole space has been crippled by scams and ripoffs of all kinds with virtually no way to settle disputes. What the crypto enthusiasts have been learning in the process is that central banks of nation states does not exist without the other functions of political power : political organisation, justice, enforcement and ultimately taxes to fund it all. To build a functioning human system requires some form of agreement and justice - a legal system - and proper ways to enforce (legal) decisions.

The judicial system itself is often represented by a blind lady - a "trustless" figure - serving the law. Ultimately, justice and politics are two sides of the same coin that allow human organisations to function. While the material forms of crypto trials still to be defined, defining proper mechanisms for conflict resolutions seems the most important step for them to develop into viable global institutions.

Footnotes

[1] http://Animal trial - Wikipedia ...
[2] http://Should Animals That Do Ba...
[3] http://Monkey selfie copyright d...
[4] http://Whanganui River - Wikiped...
[5] https://www.researchgate.net/pub...
[6] Getting back our imagination about the regulation of algorithms
[7] The Human Use of Human Beings - Wikipedia
[8] Clément Renaud's answer to What do you think about China's Social Credit System?

37 Views · View 1 Upvoter

View 1 other answer to this question

About the Author

Clément Renaud
Hack writer and researcher

Researcher
Studied at Telecom ParisTech
Lives in Lyon, France
713k answer views
18.4k this month
Knows Mandarin Chinese
FR Active in French
27 Answers

More Answers from Clément Renaud

View More

Why did France never lay claim to the Channel Islands of Jersey and Guernsey from the UK?
12.6k Views

What is the difference between "ontology" and "epistemology"?
212.4k Views

Which language learning program is more effective: Pimsleur or Rosetta Stone?
22.7k Views

What do you think about China's Social Credit System?

7.1k Views

How do I read console window errors from Chrome using JavaScript?

7.7k Views

OPEN LETTER TO THE EUROPEAN COMMISSION ARTIFICIAL INTELLIGENCE AND ROBOTICS

Sign the letter

We, Artificial Intelligence and Robotics Experts, industry leaders, law, medical and ethics experts, confirm that establishing EU-wide rules for Robotics and Artificial Intelligence is pertinent to guarantee a **high level of safety and security to the European Union citizens** while fostering **innovation**.

As human-robot interactions become

2. The creation of a **Legal Status** of an "electronic person" for "autonomous", "unpredictable" and "self-learning" robots is justified by the incorrect affirmation that damage liability would be impossible to prove.

From a **technical** perspective, this statement offers many bias based on an overvaluation of the actual capabilities

common place, the European Union needs to offer the appropriate framework to **reinforce Democracy and European Union values.** In fact, the Artificial Intelligence and Robotics framework must be explored not only through economic and legal aspects, but also through its societal, psychological and ethical impacts. In this context, we are **concerned** by the European Parliament Resolution on Civil Law Rules of Robotics, and its recommendation to the European Commission in its paragraph 59 f):

> "Creating a specific **legal status for robots** in the long run, so that at least the most sophisticated autonomous robots could be established as having the status of electronic persons responsible for making good any damage they may cause, and possibly applying electronic personality to cases where robots make autonomous decisions or otherwise interact with third parties

of even the most advanced robots, a superficial understanding of unpredictability and self-learning capacities and, a robot perception distorted by Science-Fiction and a few recent sensational press announcements.

From an **ethical** and **legal** perspective, creating a legal personality for a robot is inappropriate whatever the legal status model:

a. A legal status for a robot can't derive from the **Natural Person model**, since the robot would then hold human rights, such as the right to dignity, the right to its integrity, the right to remuneration or the right to citizenship, thus directly confronting the Human rights. This would be in contradiction with the **Charter of Fundamental Rights of the European Union** and the **Convention for the Protection of Human Rights and Fundamental Freedoms.**

b. The legal status for a robot can't derive from the **Legal Entity model**, since it implies the existence of human persons behind the legal person to represent and direct it. And this is not the case for a robot.

c. The legal status for a robot can't derive

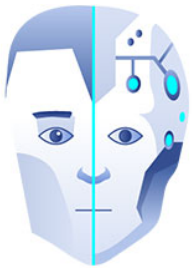
> independently;"

WE BELIEVE THAT:

1. The economical, legal, societal and ethical impact of AI and Robotics must be considered without haste or bias. The benefit to all humanity should preside over the framework for EU civil law rules in Robotics and Artificial Intelligence.

from the **Anglo-Saxon Trust model** also called ***Fiducie*** or ***Treuhand*** in Germany. Indeed, this regime is extremely complex, requires very specialized competences and would not solve the liability issue. More importantly , it would still imply the existence of a human being as a last resort – the trustee or fiduciary – responsible for managing the robot granted with a Trust or a *Fiducie*.

CONSEQUENTLY, WE AFFIRM THAT



> The European Union must prompt the development of the AI and Robotics industry insofar as to limit health and safety risks to human beings. The protection of robots' users and third parties must be at the heart of all EU legal provisions.

> The European Union must create an actionable framework for innovative and reliable AI and Robotics to spur even greater benefits for the European peoples and its common market.

TO ADD YOUR NAME TO THE BELOW EU SIGNATORIES:

EU Country

Position *

Name *

First name

Last name

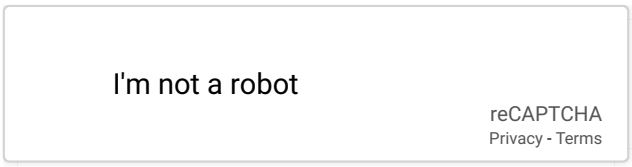
Email Address *

Email will remain unpublished. Please provide a valid email address so we can get back to you.

Organisation's name *

Describe your position

Professional organization, ethics committee, syndicates, industrial federation, IEEE or else …



Your e-mail will remain undisclosed at all times.

Sign the letter

We respect your privacy : your data will not be used for any other purpose than the OpenLetter to the European Commission. Your e-mail will remain undisclosed at all times.

THE SIGNATORIES

> Nathalie Nevejans, Lecturer in Law, University of Artois (**France**), Member of the **CNRS** Ethics Committee **COMETS**. Expert in Ethics in Robotics at the **European Parliament**, Member of the Institute for the Study of Man-Robot Relations (IERHR).

> Raja Chatila, Institute of Intelligent Systems and Robotics – Sorbonne University and **CNRS** Professor (**France**). **Director of the Institute of Intelligent Systems and Robotics (ISIR).** Former **President** of the **IEEE** Robotics and Automation Society (2014-2015). Member of the French Commission on the Ethics of Research on Digital Science and Technology (CERNA). **Chair of the IEEE** Global Initiative on Ethics of Autonomous and Intelligence Systems

> **Jozef Glasa**, **Chair** Ethics Committee, **Ministry of Health** (**Slovak Republic**); Delegate Member, Committee on Bioethics (DH-BIO, formerly CDBI), **Council of Europe**; Slovak Medical University in Bratislava, **Institute of Health Care Ethics/Institute of Clinical and Experimental Pharmacology Head/Deputy Head**. Professor of Laboratory Medicine

> **Noel Sharkey**, Emeritus Professor AI and Robotics (**United Kingdom**) **Foundation for Responsible Robotics**

> **Alexandre Pereira**, Faculty of Law Professor at University of Coimbra, (**Portugal**). IT Law & **Cybersecurity** researcher and professor.

> **Sanja Dogramadzi**, Professor of Medical Robotics at UWE, **Bristol Robotics Laboratory** (**United Kingdom**). Member of **British Standard Institute** contributing to developments of Robotics standards for Service and Medical Robotics

> **Véronique Aubergé**, **CNRS** Researcher at LIG Grenoble (**France**). Scientific Head of Living Lab DOMUS-LI, Scientific co-Head of **Robo'ethics** Rectorate of Grenoble, INP Foundation **President of Ethics Comittee for Social Robotics** of SFTAG.

> **Max Dauchet**, Emeritus professor, University of Lille (**France**) – **Chair** of the **French Commission on the Ethics of Research on Digital Science and Technology** (**CERNA**)

> **Wolfgang M. Schröder**, Professor of Philosophy, Institute for Systematic Theology, University of Würzburg (**Germany**) – Member of the **AG Digital Ethics** / Initiative D21 Berlin – Member of the **FAG Political Theory & Philosophy**, DGPhil.

> **Hugues Bersini**, Professor of Artificial Intelligence, co-Director of

Institute or Interdisciplinary Research and AI Development **IRIDIA** , Université Libre of Brussels ULB (**Belgium**) – Member of the **Belgium Academy of Science**

> Georg Martius, Research Group Leader **Max Planck Institute for Intelligent Systems** (MPG) (**Germany**)

> Koen Hindriks, Interactive Robotics Associate Professor Interactive Intelligence at Delft University of Technology (**Netherlands) – CEO of Interactive Robotics**

> Ulrich Borgolte, Senior lecturer in Robotics and Mechatronics at FernUniversität in Hagen (**Germany)**

> Benjamin Frugier, **Executive Director** of **French Federation of Mechanical Engineering Industries FEM** (**France**). The FEM is in charge of economic and technical interests of 25 trade associations, representing companies in the three following fields: Equipment, Transformation and Precision

> Gyorgy Cserey, **Head** of the Sensory Robotics Lab. at the **Faculty of Information Technology and Bionics** at Pazmany Peter Catholic University Associate professor (**Hungary**)

> Domenico G. Sorrenti, Bicocca Associate Professor, Dept. Computer Science, Universita di Milano (**Italy**) – H**ead of the Robotic Perception Laboratory**

> Margo Dessertenne, Trade groups manager at **Symop Robotics** (**France**). Symop is a French professional organization gathering 270 SME enterprises and large corporations in Automation, robotics

> David Pavlovic , Head of digital strategy & innovation, **Heineken** (**Netherlands**).

> Lionel Sublet, CEO, Techplus , Symop Robotic Group General Manager **(France).**

> Yves Poullet, Dean and Emeritus Professor of Law, University of Namur and Catholic University of Lille (**Belgium**). Expert in ethics at **UNESCO** and **Council of Europe**

> Laurence Devillers, Professor of Artificial Intelligence at Sorbonne University, (**France**) Researcher at LIMSI « Computer Science Laboratory for Mechanics and Engineering Sciences », **CNRS**, Member of the French Commission on the Ethics of Research on Digital Science and Technology (**CERNA**), **Member of the IEEE Global Initiative on Ethics of Autonomous and Intelligence Systems.**

> Jean-Claude Heudin, Professor of Artificial Intelligence, University Paris Sud (**France**), Former Director and co-founder of the Institute of Internet and Multimedia, **Expert to the European Union** on the «**Future Emerging technologies**» project.

> Ante Čović, D**irector at the Centre of excellence for Integrative Bioethics**, Vice-Rector for Organisation, Human Resources Development and Cross-University Cooperation (**Croatia**). .

> **Serge Tisseron**, Psychiatrist, Université Paris VII Denis Diderot (**France**) – **Member of the Academy of Technologies** (Institute for the Study of Robot-Human relationship, IERHR)

> Jean-Paul Laumond, **CNRS** Research Director (**France**). Member of the **French Academy of Technologies and** the **French Academy of Sciences**

> Jasna Lipozenčić, **President** of the **Academy of Medical Sciences** of Croatia (**Croatia**)

> John Michael Robson, Emeritus Professor, University of Bordeaux **(France).** Researcher in Algorithms, Distributed Computing and Theory of Computation at **LaBRI** (Laboratory of Bordeaux Research in Computing)

> Joanna Bryson, Reader (Associate Professor), University of Bath (**United Kingdom**). Expert in both AI, safety and transparency in intelligent systems, and AI ethics.

> Alan Winfield, Professor of Robot Ethics at **UWE Bristol**, Robotics Laboratory, (**United Kingdom**). Member of the British Standards Institute working group on Robot Ethics; Member of the EC **Human Brain Project** Ethics Advisory Board; Member, Executive committee, **IEEE** Global Initiative on Ethics of Autonomous and Intelligence Systems and **Chair, IEEE Standards Working Group P7001** on Transparency in Autonomous Systems; Member, WEF Global Futures Council on The Future of Technology, Values and Policy

> Kathleen Richardson, Professor of **Ethics and Culture of Robots and AI**, De Montfort University (**United Kingdom)**, Founder of **the Campaign Against Sex Robots**

> Miguel Enrique Burguete, PhD. Professor of **Philosophical Anthropology and Biopolitics**, Institute of Life Sciences and Observatory of Bioethics of the Catholic University of Valencia (**Spain)**

> Calum MacKellar, **Director of Research**, **Scottish Council on Human Bioethics**, Ethics Committee (**Scotland)**

> Richard Everson, Professor of **Machine Learning** , University of Exeter (**United Kingdom)**, **Director** of the Exeter University **Institute for Data Science and Artificial Intelligence**

> **Richard Ashcroft**, Professor of Bioethics, School of Law, Queen Mary University of London (**United Kingdom**), **Director of the LLM in Medical Law**

> **Isabelle Poirot-Mazères**, Public law and Heath Professor University of Toulouse I Capitole, (**France**)

> **Thierry Magnin**, Professor in Ethics of Sciences and Technologies, **Rector** of the Lyon Catholic University (**France**). Member of The **French National Academy of Technologies.**

> **Jean-Michel Besnier**, Emeritus Philosophy Professor at University of Paris-Sorbonne, (**France**), **Member of CNRS** and **INRA Ethics Committees**.

> **Christopher Markou,** The University of **Cambridge**, Lecturer (**United Kingdom**). **Legal Expert Committee, Responsible for Robotics**.

> **David Doat**, **Chair of Ethics & Transhumanism,** Chair of Lille Catholic University **(France)**

> **Rónán Kennedy**, Lecturer in Law, National University of Ireland Galway (**Ireland**).

See more signatories

WHY YOU SHOULD SIGN:

The European Commission has planned to issue a communication on Robotics and AI as a consequence of the Resolution on Civil Law Rules of Robotics Resolution.

If you are :

> From one of the 28 Member-States of the European Union

> An AI/ Robotics Scientist or manufacturer, a University Scholar in Law, Medicine, a Member of a Professional Organization or of an Ethics Committee

> Concerned by the way Artificial Intelligence and Robots will change our daily lives and our civil, commercial and criminal laws

Please join us in the Open Letter to protect EU innovation, EU values as well as human safety, security and health.

Sign the letter

WHO IS THIS ROBOTICS OPEN LETTER ADDRESSED TO:

This Robotics Open Letter is addressed to the European Commission and namely:

European Commission President – Jean Claude Juncker
Commissioner Research, Science and Innovation – Carlos Moedas
Commissioner Digital Economy and Society – Mariya Gabriel
Commissioner Justice, Consumers and Gender Equality – Věra Jourová
Commissioner Employment, Social Affairs, Skills and Labour Mobility – Marianne Thyssen
Commissioner Transport – Violeta Bulc
Directorate General CONNECT
Directorate General GROW
Directorate General JUST
Directorate General RTD

CIVIL LAW RULES ON ROBOTICS

On February 16 2017, the European Parliament adopted a Resolution on Civil Law Rules of Robotics.

This resolution reads in its paragraph 59 f) : *"Creating a specific legal status for robots in the long run, so that at least the most sophisticated autonomous robots could be established as having the status of electronic persons responsible for making good any damage they may cause, and possibly applying electronic personality to cases where robots make autonomous*

"In the long run, determining responsibility in case of an accident will probably become increasingly complex as the most sophisticated autonomous and self-learning robots will be able to take decision which cannot be traced back to a human agent. For these cases, the report asks the Commission to carry out an impact assessment for a

decisions or otherwise interact with third parties independently;"

In fact, a delegating amendment for §59 f) was tabled and 285 Members of Parliament voted in favor of its deletion.

Prior to the vote, Mrs Delvaux the Luxemburgese Member of the European Parliament who drafted the Resolution, wrote a communication to all members of Parliament clarifying her intentions in the Resolution:

compulsory insurance scheme, which includes the possible idea of giving the legal status of an electronic personality to robots in order to facilitate compensation for victims when human responsibility cannot be fully attributed. Liability is in fact a central part of this report, because it is indispensable for citizens' trust."

WHO ARE WE ?

We are Political Leaders, AI/robotics researchers and industry leaders, Physical and

Mental Health specialists, Law and Ethics experts gathered to voice our concern about the negative consequences of a legal status approach for robots in the European Union.

Fostering an actionable framework for civil law rules on robotics and AI consequently addressing the issue of liability of "autonomous" robots is our goal. However, we believe that creating a legal status of electronic "person" would be ideological and non-sensical and non-pragmatic.

The European Economic and Social Committee clearly stated in its opinion "The consequences of Artificial Intelligence on the (digital) single market, production, consumption, employment and society" §3.33 that they were opposed to any form of legal status for robots or AI.

Similarly, UNESCO's COMEST * report on Robotics Ethics of 2017, share a similar point of view : in the article 201 where they find "highly counterintuitive to call them 'persons' as long as they do not possess some additional qualities typically associated with human persons, such as freedom of will, intentionality, self-consciousness, moral agency or a sense of personal identity. It should be mentioned in this context, however, that the Committee on Legal Affairs of the European Parliament, in its 2016 Draft Report with recommendations to the Commission on Civil Law Rules on Robotics, already considers the possibility of "creating a specific legal status for robots, so that at least the most sophisticated autonomous robots could be established as having the status of electronic persons with specific rights and obligations, including that of making good any damage they may cause, and applying electronic personality to cases where robots make smart autonomous decisions or otherwise interact with third parties independently" (JURI, 2016, section 59.f).

* : COMEST is the World Commission on the Ethics of Scientific Knowledge and Technology from UNESCO

Contact us

Proposing the Satoshi Oath for developers

Klara Jaya Brekke

Sep 20, 2016 · 6 min read

In a time that is rife with corruption scandals and disillusionment in existing financial and political institutions the blockchain has a set of properties that has made it so attractive for so many people across industries as well as across the political spectrum: decentralisation, immutability and neutrality. But in the messy real world that we inhabit, these properties are only a set of ideal values that people might strive towards or use to their advantage rather than features that are always and fully guaranteed by the architecture.

This much can be learned from the past few years of governance crises across the major blockchain projects (see for example the DAO / Ethereum hack and Bitcoin block size conflict). Somewhere along the line, someone is making decisions and writing code, somewhere there is someone with a different idea or version of decentralisation or neutrality, and not everybody has the same types of capacities to engage with and act in the system. That is not necessarily a bad thing. Disagreement and differences should not be repressed. But what this does mean is that the problem of governance, of power and authority is not simply resolved through technical architecture, it has just radically changed shape and character to include algorithmic processes and cryptographic proof.

The Satoshi Oath for developers is designed to draw out the edges and limitations of the technical architecture in guaranteeing these values and to help each person clarify what exactly they mean for each given project they might work on. It is a tool to think through how a project might relate to the people and environments that it affects and to think beyond code.

To do this, it takes each of these three central properties of the blockchain and adds two helpful concepts to start questioning how values like immutability, neutrality and decentralisation relate to the social contexts where they are being applied: Power, Change, Delegation, Disclosure, Dissensus and Exodus.

- Immutability is a central attribute of the blockchain to ensure trust in the system and integrity of the data. But not every situation can be anticipated: the world changes, systems need to adapt and data might be incorrect and even hurt someone (take the example of medical records, credit ratings or identity systems). There might be situations where the rule of immutability needs to be flexible, where the protocol or the data in fact does need to be changed. Ask yourself, who (or what) has the **power** to **change** the protocol in such a situation? What are the processes for this? What is the full range of people and environments that might be affected by this change? How will a non-technical user be made aware of changes that might affect them? In what ways can they influence this?

- Neutrality is an ideal that should be strived for but is never fully realised because all systems operate on a set of assumptions about how the world works, what people want and how they behave. A design is usually open for negotiation in the early days, before one version wins out over another and becomes normalised and possibly made into a standard. When these assumptions are encoded into a system

they become automatically reinforced, invisible and harder to change. Ask yourself, what are the assumptions and behaviours that are **delegated** to the infrastructure to automatically reinforce? What needs to be **disclosed** in order for people to be able to understand, influence and possibly even change the assumptions of the system down the line? What languages (code or human-based) are needed for such explanations to make it understandable to those affected?

- Decentralisation could be said to be the most fundamental principle of blockchain technology. But in practice, one version of decentralisation might clash with another, and if it becomes compulsory to take part in a decentralised system, the system itself becomes an oppressive authority in its own right.—The only difference being that it is less obvious than centralised authorities of the past because power is executed across multiple actors in the network. Ask yourself, how does the system deal with **dissensus**?—Is it possible to disagree with the development of/changes to a system? How can it be expressed, by whom and with what consequences? Is **exodus** possible?—To leave the system entirely? And what are the different consequences for different types of users and contributors for disagreeing or leaving?

The Satoshi Oath for developers is a tool to make visible the informal social processes that are part of the infrastructure, to consider who or what benefits or who or what is left behind by different design decisions. A decentralised computer network does not guarantee decentralised power, transparency does not guarantee legibility and finally, code and cryptography does not guarantee neutrality.

You can read and sign the Oath on the Ethereum Blockchain. The text is hosted on IPFS.

Read Satoshi Oath here

ipfs.b9lab.com

If you are interested in the code, you can find it here:
https://etherscan.io/address/0x49311a711ea4aff7fea3e0c32066e732fe4652ba#code

Join the discussion on Reddit:
https://www.reddit.com/r/ethereum/comments/53sau2/proposing_the_satoshi_oath_f

or_developers/

Satoshi Oath *for developers*

Satoshi—name meaning clear thinking, quick witted, wise

When you are developing your own blockchain based application you are not just making another app or involved in another start-up, you are taking part in creating a new form of society. This oath is designed as a practical guide and a symbolic initiation into a community of developers with an expanded view of blockchain development. By taking this oath, you pledge to consider the full extent and range of the people and environments your work might affect, with special attention to the most marginalised and those not represented, and to consider each point below in the design and development of new applications.

The blockchain has its roots in Bitcoin, which was a response to corrupt centralized and opaque power structures that became explicit in the financial crisis of 2008. The core values that inspired its design are therefore decentralisation, openness and neutrality, but these values cannot be fully guaranteed by the technical architecture alone and must be considered for each project that is undertaken. A decentralized computer network does not guarantee decentralized power; code and cryptography does not guarantee neutrality; openness does not guarantee legibility.

IMMUTABILITY *is essential for decentralised trust, but change is inevitable and necessary: not every contingency can be anticipated, data might be incorrect or hurt someone, contexts change and protocols need to adapt over time.*

CHANGE*—I pledge to consider who or what has the power to determine change (in the protocol or data), and how to make this power visible and accountable.*

POWER*—I pledge to think of the full range of people and environments that might be affected by changes, and to consider how their voices might be represented.*

NEUTRALITY *is an ideal that should be strived for but is never fully realized because all systems operate on a set of assumptions about how the world works, what people want and how they behave. When these assumptions are encoded into a system they become automatically reinforced, invisible and harder to change.*

DELEGATION*—I pledge to consider the assumptions and biases of the*

app/platform I am developing and how these are delegated to and enforced by the code.

***DISCLOSURE**—I pledge to make these assumptions and biases visible in all the languages (code and human) necessary to make these understandable to all those who might be affected.*

***DECENTRALISATION** is the most central attribute of the blockchain, but in practice, one version of decentralisation might clash with another. If it becomes compulsory to take part in a decentralized system, the system itself becomes an oppressive authority in its own right.*

***DISSENSUS**—I pledge to consider the possibility of and consequences for disagreeing with decisions and developments of the system for all types of people.*

***EXODUS**—I pledge to consider the possibility and consequences of leaving the system/platform/app for all types of people.*

***And beyond**—I pledge to consider my limitations and seek advice from colleagues in other fields relevant to my work when needed.*

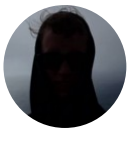
JKB—distributingchains.info

Elias, B9lab

Blockchain Ethereum

150 claps

1



Klara Jaya Brekke

Follow

B9lab blog

B9lab delivers quality instructor-led education, training and talent in blockchain and decentralized applications

Follow

Never miss a story from B9lab blog

GET UPDATES

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

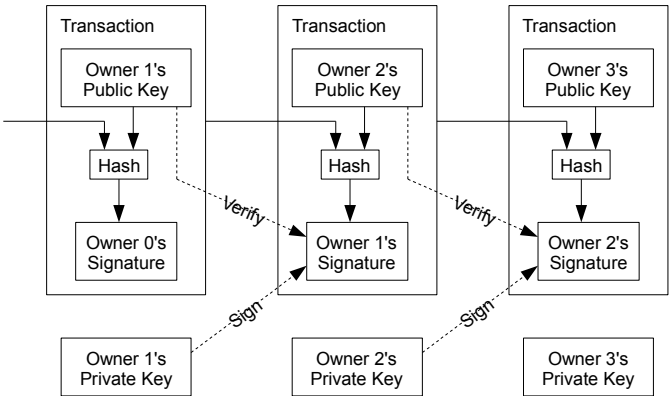
1. Introduction

Commerce on the Internet has come to rely almost exclusively on financial institutions serving as trusted third parties to process electronic payments. While the system works well enough for most transactions, it still suffers from the inherent weaknesses of the trust based model. Completely non-reversible transactions are not really possible, since financial institutions cannot avoid mediating disputes. The cost of mediation increases transaction costs, limiting the minimum practical transaction size and cutting off the possibility for small casual transactions, and there is a broader cost in the loss of ability to make non-reversible payments for non-reversible services. With the possibility of reversal, the need for trust spreads. Merchants must be wary of their customers, hassling them for more information than they would otherwise need. A certain percentage of fraud is accepted as unavoidable. These costs and payment uncertainties can be avoided in person by using physical currency, but no mechanism exists to make payments over a communications channel without a trusted party.

What is needed is an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Transactions that are computationally impractical to reverse would protect sellers from fraud, and routine escrow mechanisms could easily be implemented to protect buyers. In this paper, we propose a solution to the double-spending problem using a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions. The system is secure as long as honest nodes collectively control more CPU power than any cooperating group of attacker nodes.

2. Transactions

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

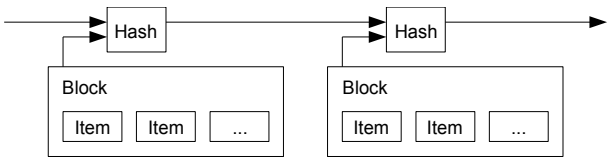


The problem of course is the payee can't verify that one of the owners did not double-spend the coin. A common solution is to introduce a trusted central authority, or mint, that checks every transaction for double spending. After each transaction, the coin must be returned to the mint to issue a new coin, and only coins issued directly from the mint are trusted not to be double-spent. The problem with this solution is that the fate of the entire money system depends on the company running the mint, with every transaction having to go through them, just like a bank.

We need a way for the payee to know that the previous owners did not sign any earlier transactions. For our purposes, the earliest transaction is the one that counts, so we don't care about later attempts to double-spend. The only way to confirm the absence of a transaction is to be aware of all transactions. In the mint based model, the mint was aware of all transactions and decided which arrived first. To accomplish this without a trusted party, transactions must be publicly announced [1], and we need a system for participants to agree on a single history of the order in which they were received. The payee needs proof that at the time of each transaction, the majority of nodes agreed it was the first received.

3. Timestamp Server

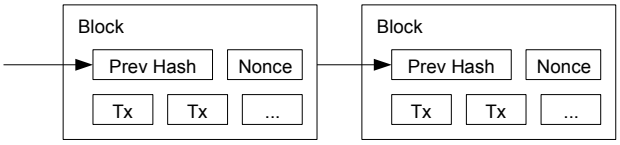
The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



4. Proof-of-Work

To implement a distributed timestamp server on a peer-to-peer basis, we will need to use a proof-of-work system similar to Adam Back's Hashcash [6], rather than newspaper or Usenet posts. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash.

For our timestamp network, we implement the proof-of-work by incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits. Once the CPU effort has been expended to make it satisfy the proof-of-work, the block cannot be changed without redoing the work. As later blocks are chained after it, the work to change the block would include redoing all the blocks after it.



The proof-of-work also solves the problem of determining representation in majority decision making. If the majority were based on one-IP-address-one-vote, it could be subverted by anyone able to allocate many IPs. Proof-of-work is essentially one-CPU-one-vote. The majority decision is represented by the longest chain, which has the greatest proof-of-work effort invested in it. If a majority of CPU power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains. To modify a past block, an attacker would have to redo the proof-of-work of the block and all blocks after it and then catch up with and surpass the work of the honest nodes. We will show later that the probability of a slower attacker catching up diminishes exponentially as subsequent blocks are added.

To compensate for increasing hardware speed and varying interest in running nodes over time, the proof-of-work difficulty is determined by a moving average targeting an average number of blocks per hour. If they're generated too fast, the difficulty increases.

5. Network

The steps to run the network are as follows:

1) New transactions are broadcast to all nodes.
2) Each node collects new transactions into a block.
3) Each node works on finding a difficult proof-of-work for its block.
4) When a node finds a proof-of-work, it broadcasts the block to all nodes.
5) Nodes accept the block only if all transactions in it are valid and not already spent.
6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it. If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first. In that case, they work on the first one they received, but save the other branch in case it becomes longer. The tie will be broken when the next proof-of-work is found and one branch becomes longer; the nodes that were working on the other branch will then switch to the longer one.

New transaction broadcasts do not necessarily need to reach all nodes. As long as they reach many nodes, they will get into a block before long. Block broadcasts are also tolerant of dropped messages. If a node does not receive a block, it will request it when it receives the next block and realizes it missed one.

6. Incentive

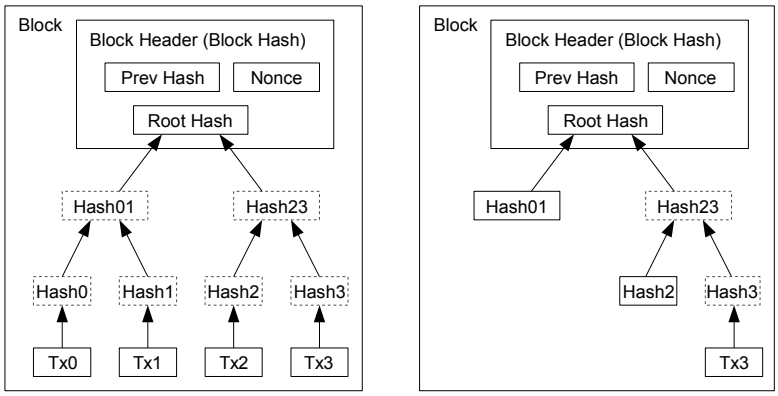
By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

7. Reclaiming Disk Space

Once the latest transaction in a coin is buried under enough blocks, the spent transactions before it can be discarded to save disk space. To facilitate this without breaking the block's hash, transactions are hashed in a Merkle Tree [7][2][5], with only the root included in the block's hash. Old blocks can then be compacted by stubbing off branches of the tree. The interior hashes do not need to be stored.



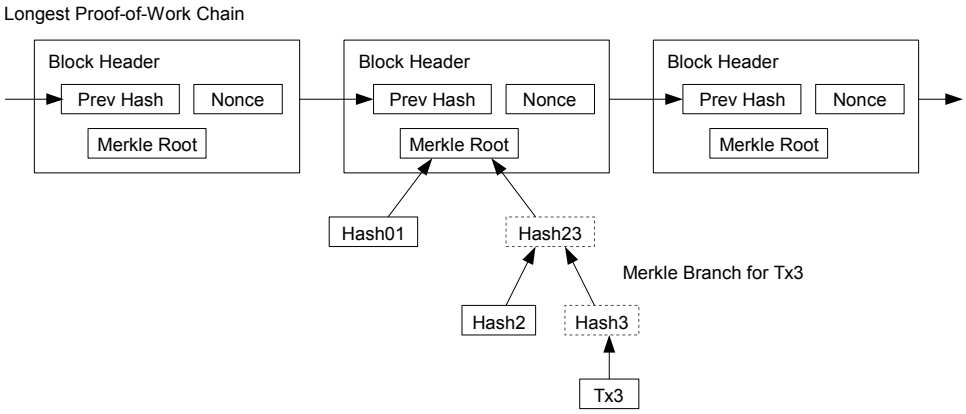
Transactions Hashed in a Merkle Tree

After Pruning Tx0-2 from the Block

A block header with no transactions would be about 80 bytes. If we suppose blocks are generated every 10 minutes, 80 bytes * 6 * 24 * 365 = 4.2MB per year. With computer systems typically selling with 2GB of RAM as of 2008, and Moore's Law predicting current growth of 1.2GB per year, storage should not be a problem even if the block headers must be kept in memory.

8. Simplified Payment Verification

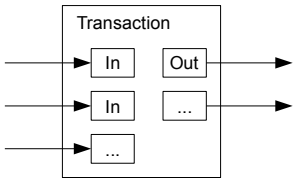
It is possible to verify payments without running a full network node. A user only needs to keep a copy of the block headers of the longest proof-of-work chain, which he can get by querying network nodes until he's convinced he has the longest chain, and obtain the Merkle branch linking the transaction to the block it's timestamped in. He can't check the transaction for himself, but by linking it to a place in the chain, he can see that a network node has accepted it, and blocks added after it further confirm the network has accepted it.



As such, the verification is reliable as long as honest nodes control the network, but is more vulnerable if the network is overpowered by an attacker. While network nodes can verify transactions for themselves, the simplified method can be fooled by an attacker's fabricated transactions for as long as the attacker can continue to overpower the network. One strategy to protect against this would be to accept alerts from network nodes when they detect an invalid block, prompting the user's software to download the full block and alerted transactions to confirm the inconsistency. Businesses that receive frequent payments will probably still want to run their own nodes for more independent security and quicker verification.

9. Combining and Splitting Value

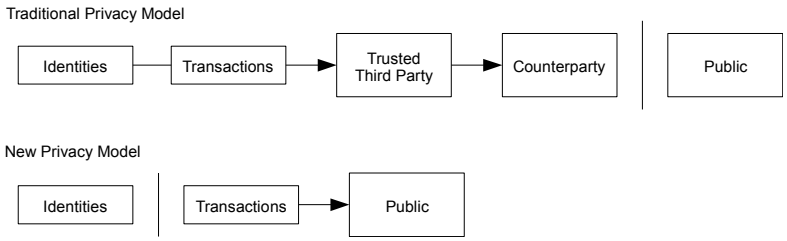
Although it would be possible to handle coins individually, it would be unwieldy to make a separate transaction for every cent in a transfer. To allow value to be split and combined, transactions contain multiple inputs and outputs. Normally there will be either a single input from a larger previous transaction or multiple inputs combining smaller amounts, and at most two outputs: one for the payment, and one returning the change, if any, back to the sender.



It should be noted that fan-out, where a transaction depends on several transactions, and those transactions depend on many more, is not a problem here. There is never the need to extract a complete standalone copy of a transaction's history.

10. Privacy

The traditional banking model achieves a level of privacy by limiting access to information to the parties involved and the trusted third party. The necessity to announce all transactions publicly precludes this method, but privacy can still be maintained by breaking the flow of information in another place: by keeping public keys anonymous. The public can see that someone is sending an amount to someone else, but without information linking the transaction to anyone. This is similar to the level of information released by stock exchanges, where the time and size of individual trades, the "tape", is made public, but without telling who the parties were.



As an additional firewall, a new key pair should be used for each transaction to keep them from being linked to a common owner. Some linking is still unavoidable with multi-input transactions, which necessarily reveal that their inputs were owned by the same owner. The risk is that if the owner of a key is revealed, linking could reveal other transactions that belonged to the same owner.

11. Calculations

We consider the scenario of an attacker trying to generate an alternate chain faster than the honest chain. Even if this is accomplished, it does not throw the system open to arbitrary changes, such as creating value out of thin air or taking money that never belonged to the attacker. Nodes are not going to accept an invalid transaction as payment, and honest nodes will never accept a block containing them. An attacker can only try to change one of his own transactions to take back money he recently spent.

The race between the honest chain and an attacker chain can be characterized as a Binomial Random Walk. The success event is the honest chain being extended by one block, increasing its lead by +1, and the failure event is the attacker's chain being extended by one block, reducing the gap by -1.

The probability of an attacker catching up from a given deficit is analogous to a Gambler's Ruin problem. Suppose a gambler with unlimited credit starts at a deficit and plays potentially an infinite number of trials to try to reach breakeven. We can calculate the probability he ever reaches breakeven, or that an attacker ever catches up with the honest chain, as follows [8]:

p = probability an honest node finds the next block
q = probability the attacker finds the next block
q_z = probability the attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & \text{if } p \leq q \\ (q/p)^z & \text{if } p > q \end{cases}$$

Given our assumption that $p > q$, the probability drops exponentially as the number of blocks the attacker has to catch up with increases. With the odds against him, if he doesn't make a lucky lunge forward early on, his chances become vanishingly small as he falls further behind.

We now consider how long the recipient of a new transaction needs to wait before being sufficiently certain the sender can't change the transaction. We assume the sender is an attacker who wants to make the recipient believe he paid him for a while, then switch it to pay back to himself after some time has passed. The receiver will be alerted when that happens, but the sender hopes it will be too late.

The receiver generates a new key pair and gives the public key to the sender shortly before signing. This prevents the sender from preparing a chain of blocks ahead of time by working on it continuously until he is lucky enough to get far enough ahead, then executing the transaction at that moment. Once the transaction is sent, the dishonest sender starts working in secret on a parallel chain containing an alternate version of his transaction.

The recipient waits until the transaction has been added to a block and z blocks have been linked after it. He doesn't know the exact amount of progress the attacker has made, but assuming the honest blocks took the average expected time per block, the attacker's potential progress will be a Poisson distribution with expected value:

$$\lambda = z\frac{q}{p}$$

To get the probability the attacker could still catch up now, we multiply the Poisson density for each amount of progress he could have made by the probability he could catch up from that point:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{if } k \le z \\ 1 & \text{if } k > z \end{cases}$$

Rearranging to avoid summing the infinite tail of the distribution...

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Converting to C code...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

Running some results, we can see the probability drop off exponentially with z.

```
q=0.1
z=0     P=1.0000000
z=1     P=0.2045873
z=2     P=0.0509779
z=3     P=0.0131722
z=4     P=0.0034552
z=5     P=0.0009137
z=6     P=0.0002428
z=7     P=0.0000647
z=8     P=0.0000173
z=9     P=0.0000046
z=10    P=0.0000012

q=0.3
z=0     P=1.0000000
z=5     P=0.1773523
z=10    P=0.0416605
z=15    P=0.0101008
z=20    P=0.0024804
z=25    P=0.0006132
z=30    P=0.0001522
z=35    P=0.0000379
z=40    P=0.0000095
z=45    P=0.0000024
z=50    P=0.0000006
```

Solving for P less than 0.1%...

```
P < 0.001
q=0.10   z=5
q=0.15   z=8
q=0.20   z=11
q=0.25   z=15
q=0.30   z=24
q=0.35   z=41
q=0.40   z=89
q=0.45   z=340
```

12. Conclusion

We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity. Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis. Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.

References

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," In *Sequences II: Methods in Communication, Security and Computer Science*, pages 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28-35, April 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.

http://ethereum.org/eth Go DEC JAN MAR
140 captures 11 2013 2014 2015
11 Jan 2014 - 22 Feb 2019 About this capture

Ethereum: A Next-Generation Generalized Smart Contract and Decentralized Application Platform

In the last few months, there has been a great amount of interest into the area of using Bitcoin-like blockchains, the mechanism that allows for the entire world to agree on the state of a public ownership database, for more than just money. Commonly cited applications include "colored coins", the idea of using on-blockchain digital assets to represent custom currencies and financial instruments, "smart property", physical objects such as cars which track a colored coin on a blockchain to determine their present legitimate owner, as well as more advanced applications such as decentralized exchange, financial derivatives, peer-to-peer gambling and on-blockchain identity and reputation systems. Perhaps the most ambitious of all is the concept of "autonomous agents" or "decentralized autonomous corporations" - autonomous entities that operate on the blockchain without any central control whatsoever, eschewing all dependence on legal contracts and organizational bylaws in favor of having resources and funds autonomously managed by a self-enforcing smart contract on a cryptographic blockchain.

However, most of these applications are difficult to implement today, simply because the scripting systems of Bitcoin, and even proto-cryptocurrency-2.0 alternatives such as the Bitcoin-based colored coins protocol and so-called "metacoins", are far too limited to allow the kind of arbitrarily complex computation that DACs require. What this project intends to do is take the innovations that such protocols bring, and generalize them - create a fully-fledged, Turing-complete (but heavily fee-regulated) cryptographic ledger that allows participants to encode arbitrarily complex contracts, autonomous agents and relationships that will be mediated entirely by the blockchain. Rather than being limited to a specific set of transaction types, users will be able to use Ethereum as a sort of "Minecraft of crypto-finance" - that is to say, one will be able to implement any feature that one desires simply by coding it in the protocol's internal scripting language. Custom currencies, financial derivatives, identity systems and decentralized organizations will all be easy to do, and it will also be possible to construct transaction types that even the Ethereum developers did not imagine. Altogether, we believe that this design is a solid step toward the realization of "cryptocurrency 2.0"; we hope that Ethereum will be to cryptocurrency what Web 2.0 was to the World Wide Web circa 1995.

Why A New Protocol

When one wants to create a new application, especially one in so delicate an area as cryptography or cryptocurrency, the immediate, and correct, first instinct is to use existing protocols as much as possible. There is no need to create a new currency, or even a new protocol, when the problem can be solved entirely by using existing technologies. Indeed, the puzzle of attempting to solve the problems of smart property, smart contracts and decentralized autonomous corporations on top of Bitcoin is how our interest in cryptocurrency 2.0 technologies originally started. Over the course of our research, however, it became evident that while the Bitcoin protocol is more than adequate for currency, basic multisignature escrow and certain simple versions of smart contracts, there are fundamental limitations that make it non-viable for anything beyond a certain very limited scope of features.

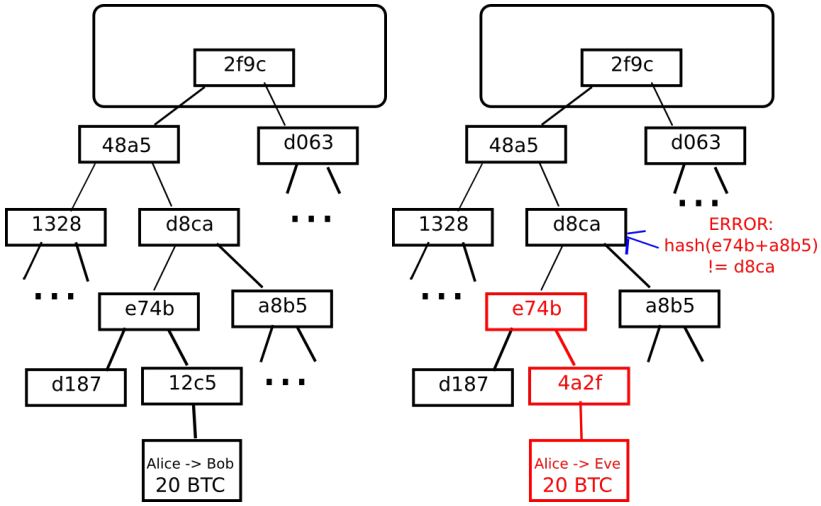
Colored Coins

The first attempt to implement a system for managing smart property and custom currencies and assets on top of a blockchain was built as a sort of overlay protocol on top of Bitcoin itself, with many advocates making a comparison to the way that HTTP serves as a layer on top of TCP in the internet protocol stack. The colored coins protocol is roughly defined as follows:

1. A colored coin issuer determines that a given transaction output $H:i$ (H being the transaction hash and i the output index) represents a certain asset, and publishes a "color definition" specifying this transaction output alongside what it represents (eg. 1 satoshi from $H:i$ = 1 ounce of gold redeemable at amagimetals.com).
2. Others "install" the color definition file in their colored coin clients.
3. When the color is first released, output $H:i$ is the only transaction output to have that color.
4. If a transaction spends inputs with color X, then its outputs will also have color X. For example, if the owner of $H:i$ immediately makes a tranasction to split that output among five addresses, then those transaction outputs will all also have color X. If a transaction has inputs of different colors, then a "color transfer rule" or "color kernel" determines which colors which outputs are (eg. a very naive implementation may say that output 0 has the same color as input 0, output 1 the same color as input 1, etc).
5. When a colored coin client notices that it received a new transaction output, it uses a back-tracing algorithm based on the color kernel to determine the color of the output. Because the rule is deterministic, all clients will agree on what color (or colors) each output has.

However, the protocol has several fundamental flaws:

1. **Difficulty of simplified payment verification** - Bitcoin's Merkle tree construction allows for a protocol known as "simplified payment verification", where a client that does not download the full blockchain can quickly determine the validity of a transaction output by asking other nodes to provide a series of hashes starting from the transaction hash, going up the Merkle tree and culminating at the root hash in the block header. The client will still need to download the block headers to be secure, but the amount of data bandwidth and verification time required drops by a factor of nearly a thousand. With colored coins, this is much harder. The reason is that one cannot determine the color of a transaction output simply by looking up the Merkle tree; rather, one needs to employ the backward scanning algorithm, fetching potentially hundreds of transactions and requesting a Merkle tree validity proof of each one, before a client can be fully satisfied that a transaction has a certain color. After over a year of investigation, including help from ourselves, no solution has been found to this problem.



Simplified payment verification in Bitcoin. Any attempt to change any part of the Merkle tree will eventually lead to an inconsistency somewhere up the chain.

2. **Incompatibility with scripting** - as mentioned above, Bitcoin does have a moderately flexible scripting system, for example allowing users to sign transactions of the form "I release this transaction output to anyone who is willing to pay to me 1 BTC". Other examples include assurance contracts, efficient micropayments and on-blockchain auctions. However, this system is inherently not color-aware; that is to say, one cannot make a transaction of the form "I release this transaction output to anyone who is willing to pay me one gold coin defined by the genesis H:i", because the scripting language has no idea that there even are different colors. One major consequence of this is that, while trust-free swapping of two different colored coins is possible, a full decentralized exchange is not since there is no way to place an order to buy or sell that is enforceable.
3. **Same limitations as Bitcoin** - theoretically, on-blockchain protocols can support advanced derivatives, bets and various kinds of conditional transfers that will be described in more detail later in this paper. Colored coins inherits the limitations of Bitcoin in terms of the impossibility of many such arrangements.

Metacoins

Another concept, once again in the spirit of sitting on top of Bitcoin much like HTTP over TCP, is that of "metacoins". The concept of a metacoin is simple: the metacoin protocol provides for a way of encoding metacoin transaction data into the outputs of a Bitcoin transaction, and a metacoin node works by processing all Bitcoin transactions and evaluating Bitcoin transactions that are valid metacoin transactions in order to determine the current balance sheet at any given time. For example, a simple metacoin protocol might require a transaction to have four outputs: MARKER, FROM, TO and VALUE. MAKRER would be a specific marker address to identify a transaction as a metacoin transaction, FROM would be the address that coins are sent from,TO would be the address that coins are sent to andVALUE would be an address encoding the amount sent. Because the Bitcoin protocol is not metacoin-aware, and thus will not reject invalid metacoin transactions, the metacoin protocol must treat all transactions with the first output going to MARKER as valid and react accordingly. For example, an implementation of the relevant part of this metacoin protocol might look like this:

```
if tx.output[0] != MARKER:
    break
else if balance[tx.output[1]] < decode_value(tx.output[3]):
    break
else if not tx.hasSignature(tx.output[1]):
    break
else:
    balance[tx.output[1]] -= decode_value(tx.output[3]);
    balance[tx.output[2]] += decode_value(tx.output[3]);
```

The advantage of a metacoin protocol is that it can allow for more advanced transaction types, including custom currencies, decentralized exchange, derivatives, etc, that are impossible on top of Bitcoin itself. However, metacoins on top of Bitcoin have one major flaw: simplified payment verification, already difficult with colored coins, is outright impossible on a metacoin. The reason is that while one can use SPV to determine that there is a transaction sending 30 metacoins to address X, that by itself does not mean that address X has 30 metacoins; what if the sender of the transaction did not have 30 metacoins to start with and so the transaction is invalid? Finding out any part of the current

state essentially requires scanning through all transactions going back to the metacoin's original launch to figure out which transactions are valid and which ones are not. This makes it impossible to have a truly secure client without downloading the entire 12 GB Bitcoin blockchain.

Ethereum solves both of these issues by being hosted on its own blockchain, and by storing a distinct "state tree" representing the current balance of each address and a "transaction list" representing the transactions between the current block and the previous block in each block. Ethereum contracts are allowed to have an arbitrarily large memory, and the Turing-complete scripting language makes it possible to encode an entire currency inside of a single contract. The intention of Ethereum is not to replace colored coins and metacoins by offering superior features; rather, Ethereum intends to serve as a superior foundational layer offering a uniquely powerful scripting system on top of which arbitrarily advanced contracts, currencies and other decentralized applications can be built. If existing colored coins and metacoin projects were to move onto Ethereum, they would gain the benefits of Ethereum's simplified payment verification, compatibility with financial derivatives and decentralized exchange, and the ability to work together on a single network. With Ethereum, someone with an idea for a new contract or transaction type that might drastically improve the state of what can be done with cryptocurrency would not need to start their own coin; they could simply implement their idea in Ethereum script code.

Philosophy

The design behind Ethereum is intended to follow the following principles:

1. **Simplicity** - the Ethereum protocol should be as simple as possible, even at the cost of some data storage inefficiency or time inefficiency. An average programmer should ideally be able to follow and implement the entire specification, so as to eventually help minimize the influence that any specific individual or elite group can have on the protocol and furthering the vision of Ethereum as a protocol that is open to all. Optimizations which add complexity should not be included unless they provide very substantial benefit.
2. **Universality** - it is a fundamental part of Ethereum's design philosophy that Ethereum does not have "features". Instead, Ethereum provides an internal Turing-complete scripting language which you can use to construct any smart contract or transaction type that can be mathematically defined. Want to invent your own financial derivative? With Ethereum, you can. Want to make your own currency? Set it up as an Ethereum contract. Want to set up a full-scale Daemon or Skynet? You may need to have a few thousand interlocking contracts, and be sure to feed them generously, to do that, but nothing is stopping you.
3. **Modularity** - different parts of the Ethereum protocol should be designed to be as modular and separable as possible. Over the course of development, it should be easy to make a small protocol modification in one place and have the application stack continue to function without any further modification. Innovations such as Dagger, Patricia trees and RLP should be implemented as separate libraries and made to be feature-complete even if Ethereum does not require certain features so as to make them usable in other protocols as well. Ethereum development should be maximally done so as to benefit the entire cryptocurrency ecosystem, not just itself.
4. **Non-discrimination** - the protocol should not attempt to actively restrict or prevent specific categories of usage, and all regulatory mechanisms in the protocol should be designed to directly regulate the harm, not attempt to oppose specific undesirable applications. You can even run an infinite loop script on top of Ethereum for as long as you are willing to keep paying the per-computational-step transaction fee for it.

Basic Building Blocks

At its core, Ethereum starts off as a fairly regular proof-of-work mined cryptocurrency without many extra complications; in fact, Ethereum is in many ways simpler than the Bitcoin-based cryptocurrencies that we use today. The concept of a transaction having multiple inputs and outputs, for example, is gone, replaced by a more intuitive balance-based model (to prevent transaction replay attacks, as part of each account balance we also store an incrementing nonce). Sequence numbers and lock times are also removed, and all transaction and block data is encoded in a single format. Instead of addresses being the RIPEMD160 hash of the SHA256 hash of the public key prefixed with 04, addresses are simply the last 20 bytes of the SHA3 hash of the public key. Unlike other cryptocurrencies, which aim to have "features", Ethereum intends to take features away, and instead provide its users with near-infinite power through an all-encompassing mechanism called "contracts".

Ethereum Client P2P Protocol

The Ethereum client P2P protocol is a fairly standard cryptocurrency protocol, and can just as easily be used for any other cryptocurrency; the only modification is the introduction of the "Greedy Heaviest Observed Subtree" (GHOST) protocol first introduced by Yonatan Sompolinsky and Aviv Zohar in December 2013; the motivation for and implementation of GHOST will be described in more detail below. The Ethereum client will be entirely reactive; it will not do anything by itself (except for the networking daemon maintaining connections) unless provoked. However, the client will also be more powerful; unlike bitcoind, which only stores a limited amount of data about the blockchain, the Ethereum client will also act as a fully functional backend for a block explorer.

When the client reads a message, it will perform the following steps:

1. Hash the data, and check if the data with that hash has already been received. If so, exit. Otherwise, pass it along to the data parser.
2. If the data parser says that a data item is a valid block, go to step 3. If the data parser says that a data item is a transaction, see if there are enough funds in the sending address for the transaction to go through, and if there are add it to the local transaction list and publish it to the network. If the data parser says that a data item is a message, send the message to the message responder and return the response.
3. Check if the "parent" parameter in the block is already stored in the database. If it is not, exit
4. Check if every block header in the "uncles" parameter in the block has the block's parent as its own parent. If any is not, exit. Note that uncle blocks do not need to be in the database; they just need to have valid proof of work.
5. Call the state updater with arguments (1) the parent of the block, (2) the transaction list of the block, (3) the timestamp of the block and (4) the coinbase of the block and see if the block header outputted by the state updater is exactly the same. If not, exit. If yes, add the block to the database and publish it to the network.
6. Determine TD(block) ("total difficulty") for the new block. TD is defined recursively by TD(genesis_block) = 0 and TD(B) = TD(B.parent) + sum(u.difficulty for u in B.uncles) + B.difficulty. If the new block has higher TD than the current block, set the current block to the new block.

If the node is mining, the node performs the following additional steps upon receiving a block:

1. Determine if the block's parent is in the database. If not, discard it.
2. Determine TD(block) ("total difficulty"). If TD(block) is higher than of any existing block in the database, start mining on that block and clear all uncles. Also, remove all transactions in the new block from the transaction pool if present, and immediately apply the remaining transactions to the new block.
3. If the block's parent is the parent of the highest block, add the block header to the set of uncles and restart the proof of work.

Upon receiving a transaction, the miner should apply the transaction to the current block and then start mining the new block header after the current mining round finishes.

The motivation behind GHOST is that blockchains with fast confirmation times currently suffer from reduced security due to a high stale rate; because blocks take a certain time to propagate through the network, call it t, if miner A mines a block and then miner B happens to mine another block before miner A's block propagates to B, miner B's block will end up wasted and will not contribute to network security. Furthermore, there is a centralization issue: if miner A is a mining pool with 30% hashpower and B has 10% hashpower, A will have a risk of producing stale blocks 70% of the time whereas B will have a risk of producing stale blocks 90% of the time. Thus, if the stale rate is high, A will be substantially more efficient simply by virtue of its size. With these two effects combined, blockchains which produce blocks quickly are very likely to lead to one mining pool having enough percentage of the network to have de facto control over the mining process. GHOST solves the first issue of network security loss by including stale blocks in the calculation of which chain is the "longest". Ethereum only does GHOST down one level for simplicity (ie. stale blocks must be included as uncles in the next block in order to count), but this should have 90%+ of the benefit of GHOST. Also, in Ethereum we give stales 3/4 of their block reward (and the nephew that includes them 1/8 as a reward); this modification is intended to solve the second issue of centralization bias.

Currency and Issuance

The Ethereum network includes its own built-in currency, ether. The main reason for including a currency in the network is twofold. First, like Bitcoin, ether is rewarded to miners to as to incentivize network security. Second, it serves as a mechanism for paying transaction fees for anti-spam purposes. Of the two main alternatives to fees, per-transaction proof of work similar to Hashcash and feeless laissez-faire, the former is wasteful of resources and unfairly punitive against weak computers and smartphones and the latter would lead to the network being almost immediately overwhelmed by an infinitely looping "logic bomb" contract. Ether will have a theoretical hard cap of 2^{128} units (compare $2^{50.9}$ in BTC), although not more than 2^{100} units will be released in the foreseeable future. For convenience and to avoid future argument (see the current mBTC/uBTC/satoshi debate), the denominations will be pre-labelled:

- 1: wei
- 10^3: __
- 10^6: __
- 10^9: koblitz
- 10^12: szabo
- 10^15: finney
- 10^18: ether

This should be taken as an expanded version of the concept of "dollars" and "cents" or "BTC" and "satoshi" that is intended to be future proof; only szabo, finney and ether will likely be used in the foreseeable future. The right to name

the 10^3 and 10^6 units will be left as a high-level secondary reward for the fundraiser subject to pre-approval from ourselves.

The issuance model will be as follows:

- Ether will be sold in a Mastercoin-style fundraiser at the price of 1 ether for 0.0001 BTC. Suppose that X ether gets collected in this way.
- 0.25X ether will be given to the founders.
- 0.25X ether will be given to the Ethereum organization as a reserve pool to pay expenses in ETH such as ETH salaries or bounties for those developers who want part or all of their compensation to be in this form
- 0.5X ether will be mined per year forever after that point (ie. permanent linear inflation)

	After 1 year	After 5 years
Currency units	2X	4X
Fundraiser	50%	25%
Founders	12.5%	6.25%
Reserve	12.5%	6.25%
Miners	25%	62.5%

For example, after five years and assuming no transactions, 25% of the ether will be in the hands of the fundraiser participants, 6.25% in the founder pool, 6.25% paid to the reserve pool, and 62.5% will belong to miners. The linear model reduces the risk of what some see as excessive wealth concentration in Bitcoin, and gives individuals living in present and future eras a fair chance to acquire currency units, while at the same time retaining a strong incentive to invest because the inflation "rate" still tends to zero over time. We also theorize that because coins are always lost over time, and coin loss can be modeled as a percentage of the total supply per year, that the total currency supply in circulation will in fact eventually stabilize at a value equal to the annual issuance divided by the loss rate (eg. 200x annual issuance at a loss rate of 0.5%).

Data Format

All data in Ethereum will be stored in[recursive length prefix encoding](), which serializes arrays of strings of arbitrary length and dimension into strings. For example, ['dog', 'cat'] is serialized (in byte array format) as[130, 67, 100, 111, 103, 67, 99, 97, 116]; the general idea is to encode the data type and length in a single byte followed by the actual data (eg. converted into a byte array, 'dog' becomes [100, 111, 103], so its serialization is[67, 100, 111, 103]. Note that RLP encoding is, as suggested by the name, recursive; when RLP encoding an array, one is really encoding a string which is the concatenation of the RLP encodings of each of the elements. In the event that storing a number is required, the number will be stored in big-endian base 256 format (eg. 32767 in byte array format as [127, 255]).

A full block is stored as:

[block_header, transaction_list , uncle_list]

The block header is:

[parent hash, sha3(rlp_encode(uncle_list)), coinbase address, state_root, sha3(rlp_encode(transaction_list)), difficulty, timestamp, nonce]

Where the data for the proof of work is the RLP encoding of the block WITHOUT the nonceuncle_list and transaction_list are the lists of the uncle block headers and tranactions in the block, respectively.

The state_root is the root of a[Merkle Patricia tree]() containing (key, value) pairs for all addresses where each address is represented as a 20-byte binary string. At each address, the value stored in the Merkle Patricia tree is a string which is the RLP-serialized form of an object of one of the following two forms:

[balance, nonce]

[balance, nonce, contract_root]

The nonce is the number of transactions made from the address, and is incremented every time a transaction is made. The purpose of this is to (1) make each transaction valid only once to prevent replay attacks, and (2) to make it impossible (more precisely, cryptographically infeasible) to construct a contract with the same hash as a pre-existing contract. balance refers to the contract or address's balance, denominated in wei.contract_root is the root of yet another Patricia tree, conaining the contract's memory, if that address is controlled by a contract. Note that in the main Patricia tree all addresses have a length of 20, even if they start with one or more zero bytes, and in the contract subtrees all indices have a length of 32, prepending with zero bytes if necessary.

Mining algorithm

Over the past five years of experience with Bitcoin and alternative cryptocurrencies, one important property for proof of

work functions that has been discovered is that of "memory-hardness" - computing a valid proof of work should require not only a large number of computations, but also a large amount of memory. Currently, two major categories of memory-hard functions, scrypt and Primecoin mining, exist, but both are imperfect; neither require nearly as much memory as an ideal memory-hard function could require, and both suffer from time-memory tradeoff attacks, where the function can be computed with significantly less memory than intended at the cost of sacrificing some computational efficiency. Ethereum instead uses an algorithm called Dagger, a memory-hard proof of work based on moderately connected directed acyclic graphs (DAGs, hence the name), which, while far from optimal, has much stronger memory-hardness properties than anything else in use today: an estimated 50-500 MB of RAM will be required per thread depending on the choice of parameters.

Dagger is more fully specified here:http://wiki.ethereum.org/index.php/Dagger

Transactions

A transaction is stored as:

[nonce, receiving_address, value, [data item 0, data item 1 ... data item n], v, r, s]

nonce is the number of transactions already sent by that address, or an empty string if this is the first transaction(v,r,s) is the raw Electrum-style signature of the transaction without the signature made with the private key corresponding to the sending address, with 27 <= v <= 30. From an Electrum-style signature (65 bytes) it is possible to extract the public key, and thereby the address, directly. Note that all transactions in Ethereum are valid; invalid transactions or transactions with insufficient balance simply have no effect. Transaction fees will be included automatically. If one wishes to voluntarily pay a higher fee, one is always free to do so by constructing a contract which forwards transactions but automatically sends a certain amount or percentage to the miner of the current block.

Difficulty adjustment

Difficulty is adjusted by the formula:

```
D(genesis_block) = 2^36
D(block) =
    if block.timestamp >= block.parent.timestamp + 42: D(block.parent) - floor(D(block.parent) / 1024)
    else:                                              D(block.parent) + floor(D(block.parent) / 1024)
```

This stabilizes around a block time of 60 seconds automatically (note that at a block time of 60 seconds, 50% of blocks come within 42 seconds of the previous; 1 - 1/e = 63.22% come within 60 seconds), and adjusts at a maximum rate of about 0.1% per block (~100% per 12 hours).

Transactions sent to the empty string as an address are a special type of transaction, creating a "contract".

Contracts

Here is where we get to the actually interesting part of the Ethereum protocol. In Ethereum, there are actually two types of entities that can generate and receive transactions: actual people (or bots, as cryptographic protocols cannot distinguish between the two) and contracts. A contract is essentially an automated agent that lives on the Ethereum network, has an Ethereum address and balance, and can send and receive transactions. A contract is "activated" every time someone sends a transaction to it, at which point it runs its code, perhaps modifying its internal state or even sending some transactions, and then shuts down. The "code" for a contract is written in a special-purpose low-level language consisting of a stack, which is not persistent, and 2^256 memory entries, which constitute the contract's permanent state. Note that Ethereum users will not need to code in this low-level stack language; we will provide a simple C-like language with variables, expressions, conditionals and while loops, and provide a compiler down to Ethereum code.

Applications

Here are some examples of what can be done with Ethereum contracts, with all code examples written in a language that will likely be very similar to the C-like high-level language that we will release. The variables tx.sender, tx.value, tx.fee, tx.data and tx.datan are properties of the incoming transaction,contract.memory, contract.address and contract.creator of the contract itself, and block.contract_memory, block.addressbalance, block.number, block.difficulty, block.basefee and block.timestamp properties of the block. block.basefee is the "base fee" which all transaction fees in Ethereum are calculated as a multiple of; for more info see the "fees" section below. All variables expressed as capital letters (eg. A) are constants, to be set by the contract creator when actually releasing the contract. The keywords wei, szabo, finney and ether, representing the denominations of ether, simply act as multipliers of 1, 1 million, etc, respectively.

Sub-currencies

Sub-currencies have many applications, ranging from currencies representing assets such as USD or gold to company stocks and even currencies with only one unit issued to represent collectibles or smart property. Advanced special-purpose financial protocols sitting on top of Ethereum may also wish to organize themselves with an internal currency. Sub-currencies are surprisingly easy to implement in Ethereum; this section describes a fairly simple contract for doing so.

The idea is that if someone wants to send `X` currency units to address `A` in currency contract `C`, they will need to make a transaction of the form `(C, 100 * block.basefee, [A, X])`, and the contract parses the transaction and adjusts balances accordingly. For a transaction to be valid, it must send 100 times the base fee worth of ether to the contract in order to "feed" the contract (as each computational step after the first 16 for any contract costs the contract a small fee and the contract will stop working if its balance drains to zero).

```
if tx.value < 100 * block.basefee:
    stop
if contract.memory[1000]:
    from = tx.sender
    to = tx.data[0]
    value = tx.data[1]
    if to <= 1000:
        stop
    if contract.memory[from] < value:
        stop
    contract.memory[from] = contract.memory[from] - value
    contract.memory[to] = contract.memory[to] + value
else:
    contract.memory[mycreator] = 10000000000000000
    contract.memory[1000] = 1
```

Ethereum sub-currency developers may also wish to add some other more advanced features:

- Include a mechanism by which people can buy currency units in exchange for ether, perhaps auctioning off a set number of units every day.
- Allow transaction fees to be paid in the internal currency, and then refund the ether transaction fee to the sender. This solves one major problem that all other "sub-currency" protocols have had to date: the fact that sub-currency users need to maintain a balance of sub-currency units to use and units in the main currency to pay transaction fees in. Here, a new account would need to be "activated" once with ether, but from that point on it would not need to be recharged.
- Allow for a trust-free decentralized exchange between the currency and ether. Note that trust-free decentralized exchange between any two contracts is theoretically possible in Ethereum even without special support, but special support will allow the process to be done about ten times more cheaply.

Financial derivatives

The underlying key ingredient of a financial derivative is a data feed to provide the price of a particular asset as expressed in another asset (in Ethereum's case, the second asset will usually be ether). There are many ways to implement a data feed; one method, pioneered by the developers of Mastercoin, is to include the data feed in the blockchain. Here is the code:

```
if tx.value < block.basefee:
    stop
if tx.sender != contract.creator:
    stop
memory[data[0]] = data[1]
```

Any other contract will then be able to query index `I` of data store `D` by using `block.contract_memory(D,I)`. A more advanced way to implement a data feed may be to do it off-chain - have the data feed provider sign all values, require anyone attempting to trigger the contract to include the latest signed data, and then use Ethereum's internal scripting functionality to verify the signature. Pretty much any derivative can be made from this, including leveraged trading, options, and even more advanced constructions like collateralized debt obligations (no bailouts here though, so be mindful of black swan risks).

To show an example, let's make a hedging contract. The basic idea is that the contract is created by party `A`, who puts up 4000 ether as a deposit. The contract then lies open for any party to accept it by putting in 1000 ether. Say that 1000 ether is worth $25 at the time the contract is made, according to index `I` of data store `D`. If party `B` accepts it, then after 30 days anyone can send a transaction to make the contract process, sending the same dollar value worth of ether (in our example, $25) back to `B` and the rest to `A`. `B` gains the benefit of being completely insulated against currency volatility risk without having to rely on any issuers. The only risk to `B` is if the value of ether falls by over 80% in 30 days - and even then, if `B` is online `B` can simply quickly hop onto another hedging contract. The benefit to `A` is the

implicit 0.2% fee in the contract, and A can hedge against losses by separately holding USD in another location (or, alternatively, A can be an individual who is optimistic about the future of Ethereum and wants to hold ether at 1.25x leverage, in which case the fee may even be in B's favor).

```
state = contract.memory[1000]
if state == 0:
    if tx.value < 1000 ether:
        stop
    contract.memory[1001] = 998 * block.contract_memory(D,I)
    contract.memory[1002] = block.timestamp
    contract.memory[1003] = tx.sender
else:
    if tx.value < 200 finney:
        stop
    ethervalue = contract.memory[1000] / block.contract_memory(D,I)
    if ethervalue >= 5000 ether:
        send(contract.memory[1003],2000 ether,[])
    else if block.timestamp < contract.memory[1002]:
        send(contract.memory[1003],ethervalue,[])
        send(A,5000 - ethervalue,[])
```

More advanced financial contracts are also possible; complex multi-clause options (eg. "Anyone, hereinafter referred to as X, can claim this contract by putting in 2 USD before Dec 1. X will have a choice on Dec 4 between receiving 1.95 USD on Dec 29 and the right to choose on Dec 11 between 2.20 EUR on Dec 28 and the right to choose on Dec 18 between 1.20 GBP on Dec 30 and paying 1 EUR and getting 3.20 EUR on Dec 29") can be defined simply by storing a state variable just like the contract above but having more clauses in the code, one clause for each possible state. Note that financial contracts of any form do need to be fully collateralized; the Ethereum network controls no enforcement agency and cannot collect debt.

Identity and Reputation Systems

The earliest alternative cryptocurrency of all, Namecoin, attempted to use a Bitcoin-like blockchain to provide a name registration system, where users can register their names in a public database alongside other data. The major cited use case is for a DNS system, mapping domain names like "bitcoin.org" (or, in Namecoin's case, "bitcoin.bit") to an IP address. Other use cases include email authentication and potentially more advanced reputation systems. Here is a simple contract to provide a Namecoin-like name registration system on Ethereum:

```
if tx.value < 25 ether:
    stop
if contract.memory[tx.data[0]]:
    stop
contract.memory[tx.data[0]] = contract.memory[tx.data[1]]
```

One can easily add more complexity to allow users to change mappings, automatically send transactions to the contract and have them forwarded, and even add reputation and web-of-trust systems.

Decentralized Autonomous Organizations

The general concept of a "decentralized autonomous corporation" is that of an entity that has a certain set of shareholders which collect dividends and which, perhpas with a 67% majority, have the right to modify its code. The shareholders would collectively decide on how the corporation should allocate its funds, using either bounties, salaries or even more exotic mechanisms such as an internal currency to reward work, and the funds would be distributed automatically. This essentially replicates the legal trappings of a traditional company but using only cryptographic blockchain technology for enforcement. This is the "corporate" model of a decentralized organization; an alternative, perhaps described as a "decentralized autonomous community", would have all members have an equal share in the decision making and require 67% of existing members to agree to add or remove a member. The requirement that one person can only have one membership would then need to be enforced collectively by the group.

Some "skeleton code" for a DAO might look as follows.

There are three transaction types:

- [0,k] to register a vote in favor of a code change
- [1,k,L,v0,v1...vn] to register a code change at code$_k$ in favor of setting memory starting from location$_L$ to v0, v1 ... vn
- [2,k] to finalize a given code change

Note that the design relies on the randomness of addresses and hashes for data integrity; the contract will likely get corrupted in some fashion after about 2^128 uses, but that is acceptable since nothing close to that volume of usage will exist in the foreseeable future. 2^255 is used as a magic number to store the total number of members, and a membership is stored with a 1 at the member's address. The last three lines of the contract are there to add C as the

first member; from there, it will be C's responsibility to use the democratic code change protocol to add a few other members and code to bootstrap the organization.

```
k = sha3(tx.data[1])
if tx.data[0] == 0:
    if contract.memory[tx.sender] == 0:
        stop
    if contract.memory[k + tx.sender] == 0:
        contract.memory[k + tx.sender] = 1
        contract.memory[k] += 1
else if tx.data[0] == 1:
    if tx.value <= tx.datan * block.basefee * 200:
        stop
    if contract.memory[k] > 0:
        stop
    i = 3
    while i < tx.datan:
        contract.memory[k + i] = tx.data[i]
        i = i + 1
    contract.memory[k] = 1
    contract.memory[k+1] = tx.datan
    contract.memory[k+2] = tx.data[2]
else if tx.data[0] == 2:
    if contract.memory[k] >= contract.memory[2 ^ 255] * 2 / 3:
        if tx.value <= tx.datan * block.basefee * 200:
            stop
        i = 3
        L = contract.memory[k+1]
        loc = contract.memory[k+2]
        while i < L:
            contract.memory[loc+i-3] = tx.data[i]
            i = i + 1
if contract.memory[2 ^ 255 + 1] == 0:
    contract.memory[2 ^ 255 + 1] = 1
    contract.memory[C] = 1
```

This implements the "egalitarian" DAO model; one can easily extend it to a shareholder model by also storing how many shares each owner holds and providing a simple way to transfer shares.

Further Applications

1) **Crop insurace**. One can easily make a financial derivatives contract but using a data feed of the weather instead of any price index. If a farmer in Iowa purchases a derivative that pays out inversely based on the precipitation in Iowa, then if there is a drought the farmer will automatically receive money and if there is enough rain the farmer will be happy because their crops would do well.

2) **Generic insurance**, relying on one of a specified set of third-party judges to adjudicate claims. The contract can even include a complex appeals process if so desired.

3) A **decentrally managed data feed**, using proof-of-stake voting to give an average (or more likely, median) of everyone's opinion on the price of a commodity, the weather or any other relevant data

4) An offer to participate in a cryptographically secure, trust-free**peer-to-peer bet**

5) **SatoshiDice**. The entire gambling site can essentially be replicated on the blockchain using either peer-to-peer bet functionality or a centralized model.

6) A full-scale **on-chain stock market**. Prediction markets are also easy to implement as a trivial consequence.

7) An **on-chain decentralized marketplace**, using the identity and reputation system as a base.

How do contracts work?

A contract making transaction is encoded as follows:

```
[
    nonce,
    '',
    value,
    [
        data item 0,
        data item 1,
        ...
    ],
```

```
    v,
    r,
    s
]
```

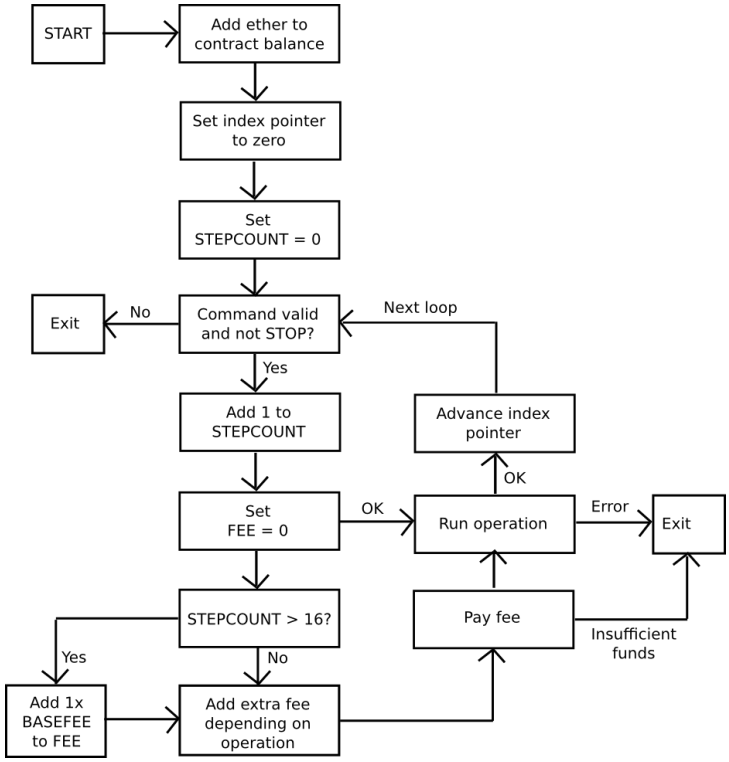
The data items will, in most cases, be script codes (more on this below). Contract creation transaction validation happens as follows:

1. Deserialize the transaction, and extract its sending address from its signature.
2. Calculate the transaction's fee. Check that the balance of the creator is at least the endowment plus the fee. If not, exit, and if yes pay the fee.
3. Take the last 20 bytes of the hash of the transaction making the contract. If a contract with that address already exists, exit. Otherwise, create the contract at that address
4. Copy data item `i` to memory slot `i` for all `i` in `[0 ... n-1]` in the contract.

Whenever a transaction is sent to a contract, the contract executes its scripting code. The precise steps that happen when a contract receives a transaction are as follows:

1. The contract's ether balance increases by the amount sent
2. The index pointer is set to zero, and `STEPCOUNT = 0`
3. Repeat forever:
    - if the command at the index pointer is `STOP`, invalid or greater than 255, exit from the loop
    - set `MINERFEE = 0`, `VOIDFEE = 0`
    - set `STEPCOUNT <- STEPCOUNT + 1`
    - if `STEPCOUNT > 16`, set `MINERFEE <- MINERFEE + STEPFEE`
    - see if the command is `LOAD` or `STORE`. If so, set `MINERFEE <- MINERFEE + DATAFEE`
    - see if the command will fill up a previously zero memory field. If so, set `VOIDFEE <- VOIDFEE + MEMORYFEE`
    - see if the commend will zero a previously used memory field. If so, set `VOIDFEE <- VOIDFEE - MEMORYFEE`
    - see if the command is `EXTRO` or `BALANCE`. If so, set `MINERFEE <- MINERFEE + EXTROFEE`
    - see if the command is a crypto operation. If so, set `MINERFEE <- MINERFEE + CRYPTOFEE`
    - if `MINERFEE + VOIDFEE > CONTRACT.BALANCE`, HALT and exit from the loop
    - else, subtract `MINERFEE + VOIDFEE` from the contract's balance and add `MINERFEE` to a running counter that will be added to the miner's balance once all transactions are parsed. Note that `MINERFEE` may be negative in some cases, in which case the contract balance would actually increase
    - run the command
    - if the command did not exit with an error, update the index pointer and return to the start of the loop. If the contract did exit with an error, break out of the loop.

Contract Script Interpretation



The language maintains a stack which is initialized as empty, and all operations either manipulate the stack, perform special operations such as sending transactions and setting memory, or do both. In the following descriptions, `S[-1]`, `S[-2]`, etc represent the topmost, second topmost, etc items on the stack. The individual operations are defined as follows:

- (00) `STOP` - halts execution
- (01) `ADD` - pops two items and pushes `S[-2] + S[-1] mod 2^256`.
- (02) `MUL` - pops two items and pushes `S[-2] * S[-1] mod 2^256`
- (03) `SUB` - pops two items and pushes `S[-2] - S[-1] mod 2^256`
- (04) `DIV` - pops two items and pushes `floor(S[-2] / S[-1])`. If `S[-1] = 0`, halts execution.
- (05) `SDIV` - pops two items and pushes `floor(S[-2] / S[-1])`, but treating values above `2^255` as negative (ie. `x -> 2^256 - x`). If `S[-1] = 0`, halts execution.
- (06) `MOD` - pops two items and pushes `S[-2] mod S[-1]`. If `S[-1] = 0`, halts execution.
- (07) `SMOD` - pops two items and pushes `S[-2] mod S[-1]`, but treating values above `2^255` as negative (ie. `x -> 2^256 - x`). If `S[-1] = 0`, halts execution.

- (08) EXP - pops two items and pushes S[-2] ^ S[-1] mod 2^256
- (09) NEG - pops one item and pushes 2^256 - S[-1]
- (0a) LT - pops two items and pushes 1 if S[-2] < S[-1] else 0
- (0b) LE - pops two items and pushes 1 if S[-2] <= S[-1] else 0
- (0c) GT - pops two items and pushes 1 if S[-2] > S[-1] else 0
- (0d) GE - pops two items and pushes 1 if S[-2] >= S[-1] else 0
- (0e) EQ - pops two items and pushes 1 if S[-2] == S[-1] else 0
- (of) NOT - pops one item and pushes 1 if S[-1] = 0 else 0
- (10) MYADDRESS - pushes the contract's address as a number
- (11) TXSENDER - pushes the transaction sender's address as a number
- (12) TXVALUE - pushes the transaction value
- (13) TXDATAN - pushes the number of data items
- (14) TXDATA - pops one item and pushes data item S[-1], or zero if index out of range
- (15) BLK_PREVHASH - pushes the hash of the previous block (NOT the current one since that's impossible!)
- (16) BLK_COINBASE - pushes the coinbase of the current block
- (17) BLK_TIMESTAMP - pushes the timestamp of the current block
- (18) BLK_NUMBER - pushes the current block number
- (19) BLK_DIFFICULTY - pushes the difficulty of the current block
- (1a) BASEFEE - pushes the base fee (x as defined in the fee section below)
- (20) SHA256 - pops one item and then ceil(S[-1] / 32) further items. Then takes those items as 32-byte strings, concatenates them in top-to-bottom order, taking out the low-order bytes of the bottommost if necessary, and pushes the SHA256 of the resulting string.
- (21) RIPEMD160 - works just like SHA256 but with the RIPEMD-160 hash
- (22) ECMUL - pops three items. If (S[-2],S[-1]) are a valid point in secp256k1, including both coordinates being less than P, pushes (S[-2],S[-1]) * S[-3], using (0,0) as the point at infinity. Otherwise, pushes (2^256 - 1, 2^256 - 1). Note that there are no restrictions on S[-3].
- (23) ECADD - pops four items and pushes (S[-4],S[-3]) + (S[-2],S[-1]) if both points are valid, otherwise (2^256 - 1,2^256 - 1).
- (24) ECSIGN - pops two items and pushes (v,r,s) as the Electrum-style RFC6979 deterministic signature of message hash S[-1] with private key S[-2] mod N.
- (25) ECRECOVER - pops four items and pushes (x,y) as the public key from the signature (S[-3],S[-2],S[-1]) of message hash S[-4]. If the signature has invalid v,r,s values (ie. v not in [27,28], r not in [0,P], s not in [0,N]), return (2^256 - 1,2^256 - 1).
- (26) ECVALID - pops two items and pushes 1 if (S[-2],S[-1]) is a valid secp256k1 point (including (0,0)) else 0
- (27) SHA3 - works just like SHA256 but with the SHA3 hash, 256 bit version.
- (30) PUSH - pushes the item in memory at the index pointer + 1, and advances the index pointer by 2.
- (31) POP - pops one item.
- (32) DUP - pushes S[-1] to the stack.
- (33) DUPN - reads item in memory at the index pointer + 1, say N, and pushes S[-N] to the stack. Advances the index pointer by 2. If N = 0, exits with an error instead.
- (34) SWAP - pops two items and pushes S[-1] then S[-2]
- (35) SWAPN - reads item in memory at the index pointer + 1, say N, and swaps S[-1] and S[-N] to the stack. Advances the index pointer by 2. If N = 0, exits with an error instead.
- (36) LOAD - pops one item and pushes the item in memory at index S[-1]
- (37) STORE - pops two items and sets the item in memory at index S[-1] to S[-2]
- (40) JMP - pops one item and sets the index pointer to S[-1]
- (41) JMPI - pops two items and sets the index pointer to S[-2] only if S[-1] is nonzero
- (42) IND - pushes the index pointer
- (50) EXTRO - pops two items and pushes memory index S[-2] of contract S[-1]
- (51) BALANCE - pops one item and pushes balance of that address, or zero if the address is invalid
- (60) MKTX - pops three items and initializes a transaction to send S[-2] ether to S[-1] with S[-3] data items. Pops that many data items and adds them in order popped as data items to the transaction. Then sends the transaction. Note that if S[-1] = 0 then this creates a new contract.
- (ff) SUICIDE - pops one item, destroys the contract and clears all memory, sending the entire balance plus the negative fee from clearing memory to the address at S[-1]

Fees

Ethereum will have seven primary fees, of which one applies to transaction senders and six apply to contracts. The fees are:

- TXFEE (100x) - fee for sending a transaction
- NEWCONTRACTFEE (100x) - fee for creating a new contract, not including the memory fee for each item in script code
- STEPFEE (x) - fee for every computational step after than first sixteen in contract execution

- MEMORYFEE (100x) - fee for adding a new item to a contract's memory, including when first creating a contract. The memory fee is the only fee that is not paid to a miner, and is refunded when memory from a contract is removed.
- DATAFEE (20x) - fee for accessing or setting a contract's memory from inside that contract
- EXTROFEE (40x) - fee for accessing memory from another contract inside a contract
- CRYPTOFEE (20x) - fee for using any of the cryptographic operations

One novel innovation in Ethereum is that the fees will be inversely proportional to the square root of the difficulty; that is, `x = floor(10^21 / floor(difficulty ^ 0.5))` . The reason why this is done is twofold:

1. It creates a fully decentralized mechanism for setting transaction fees that ensures that fees do not grow too quickly with the growth of the network. For example, if ether becomes 4x more valuable, then it is expected that network mining power and therefore difficulty will increase by 4x, meaning that ether-denominated transaction fees will decrease by 2x and therefore real-value transaction fees will only increase 2x.
2. If computers get 4x more powerful due to Moore's Law, difficulty will increase 4x and so fees will decrease 2x, reflecting computers' increased ability to handle higher load.

The reason the exponent should be kept below 1 is that (1) Moore's Law may affect computing power faster than the slowest component of the transaction processing pipeline, and (2) if the Ethereum network grows very quickly due to an increased number of computers and not any increase in power to each individual computer, an exponent of 1 may push per-computer load too high.

Conclusion

The Ethereum protocol's design philosophy is in many ways the opposite from that taken by many other cryptocurrencies today. Other cryptocurrencies aim to add complexity and increase the number of "features"; Ethereum, on the other hand, takes features away. The protocol does not "support" multisignature transactions, multiple inputs and outputs, hash codes, lock times or many other features that even Bitcoin provides. Instead, all complexity comes from an all-powerful, Turing-complete assembly language, which can be used to build up literally any feature that is mathematically describable. As a result, we have a cryptocurrency protocol whose codebase is very small, and yet which can do anything that any cryptocurrency will ever be able to do.

References and Further Reading

1. Colored coins whitepaper: https://docs.google.com/a/buterin.com/document/d/1AnkP_cVZTCMLIzw4DvsW6M8Q2JC0lIzrTLuoWu2z1BE/edit
2. Mastercoin whitepaper: https://github.com/mastercoin-MSC/spec
3. Decentralized autonomous corporations, Bitcoin Magazine: http://bitcoinmagazine.com/7050/bootstrapping-a-decentralized-autonomous-corporation-part-i/
4. Smart property: https://en.bitcoin.it/wiki/Smart_Property
5. Smart contracts: https://en.bitcoin.it/wiki/Contracts
6. Simplified payment verification: https://en.bitcoin.it/wiki/Scalability#Simplified*payment*verification
7. Merkle trees: http://en.wikipedia.org/wiki/Merkle_tree
8. Patricia trees: http://en.wikipedia.org/wiki/Patricia_tree
9. Bitcoin whitepaper: http://bitcoin.org/bitcoin.pdf
10. GHOST: http://www.cs.huji.ac.il/~avivz/pubs/13/btc*scalability*full.pdf
11. StorJ and Autonomous Agents, Jeff Garzik: http://garzikrants.blogspot.ca/2013/01/storj-and-bitcoin-autonomous-agents.html
12. Ethereum RLP: http://wiki.ethereum.org/index.php/RLP
13. Ethereum Merkle Patricia trees: http://wiki.ethereum.org/index.php/Patricia_Tree
14. Ethereum Dagger: http://wiki.ethereum.org/index.php/Dagger

ethereum / wi…

Sign up

Code Issues 22 Pull requests 2 Projects 0 Wiki Pulse

Dismiss

Join GitHub today

GitHub is home to over 31 million developers working together to host and review code, manage projects, and build software together.

Sign up

White Paper

André Lage edited this page Feb 18, 2019 · 173 revisions

Jump to bottom

A Next-Generation Smart Contract and Decentralized Application Platform

gitter Docs chat

> An introductory paper to Ethereum, introduced before launch, which is maintained.

Satoshi Nakamoto's development of Bitcoin in 2009 has often been hailed as a radical development in money and currency, being the first example of a digital asset which simultaneously has no backing or intrinsic value and no centralized issuer or controller. However, another - arguably more important - part of the Bitcoin experiment is the underlying blockchain technology as a tool of distributed consensus, and attention is rapidly starting to shift to this other aspect of Bitcoin. Commonly cited alternative applications of blockchain technology include using on-blockchain digital assets to represent custom currencies and financial instruments (colored coins), the ownership of an underlying physical device (smart property), non-fungible assets such as domain names (Namecoin), as well as more complex applications involving having digital assets being directly controlled by a piece of code implementing arbitrary rules (smart contracts) or even blockchain-based decentralized autonomous organizations (DAOs). What Ethereum intends to provide is a blockchain with a built-in fully fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the systems described

above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code.

Contents

Introduction to Bitcoin and Existing

Concepts

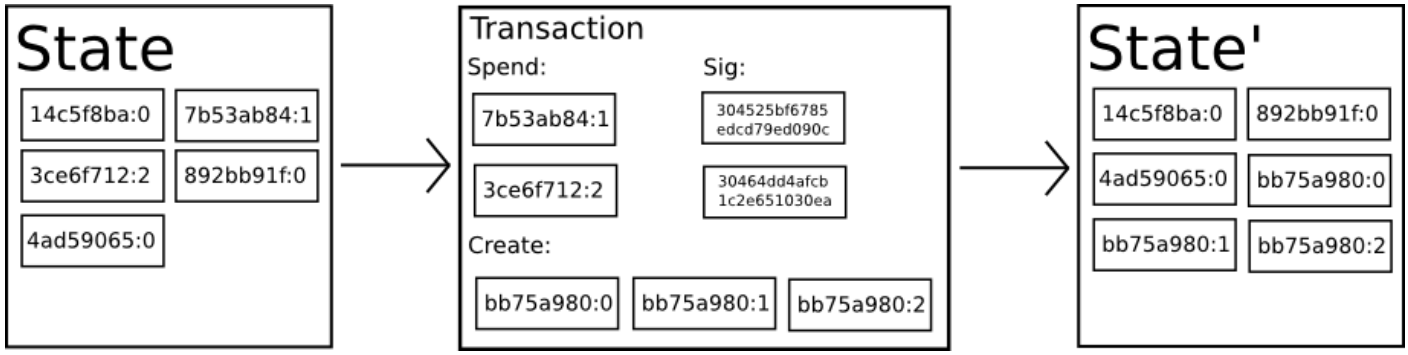
History

The concept of decentralized digital currency, as well as alternative applications like property registries, has been around for decades. The anonymous e-cash protocols of the 1980s and the 1990s, mostly reliant on a cryptographic primitive known as Chaumian blinding, provided a currency with a high degree of privacy, but the protocols largely failed to gain traction because of their reliance on a centralized intermediary. In 1998, Wei Dai's b-money became the first proposal to introduce the idea of creating money through solving computational puzzles as well as decentralized consensus, but the proposal was scant on details as to how decentralized consensus could actually be implemented. In 2005, Hal Finney introduced a concept of reusable proofs of work, a system which uses ideas from b-money together with Adam Back's computationally difficult Hashcash puzzles to create a concept for a cryptocurrency, but once again fell short of the ideal by relying on trusted computing as a backend. In 2009, a decentralized currency was for the first time implemented in practice by Satoshi Nakamoto, combining established primitives for managing ownership through public key cryptography with a consensus algorithm for keeping track of who owns coins, known as "proof of work".

The mechanism behind proof of work was a breakthrough in the space because it simultaneously solved two problems. First, it provided a simple and moderately effective consensus algorithm, allowing nodes in the network to collectively agree on a set of canonical updates to the state of the Bitcoin ledger. Second, it provided a mechanism for allowing free entry into the consensus process, solving the political problem of deciding who gets to influence the consensus, while simultaneously preventing sybil attacks. It does this by substituting a formal barrier to participation, such as the requirement to be registered as a unique entity on a particular list, with an economic barrier - the weight of a single node in the consensus voting process is directly proportional to the computing power that the node brings. Since then, an alternative approach has been proposed called *proof of stake*, calculating the weight of a node as being proportional to its currency holdings and not computational resources; the discussion of the relative merits of the two approaches is beyond the scope of this paper but it should be noted that both approaches can be used to serve as the backbone of a cryptocurrency.

Here is a blog post from Vitalik Buterin, the founder of Ethereum, onEthereum pre-history. Here is another blog post with more history.

Bitcoin As A State Transition System



From a technical standpoint, the ledger of a cryptocurrency such as Bitcoin can be thought of as a state transition system, where there is a "state" consisting of the ownership status of all existing bitcoins and a "state transition function" that takes a state and a transaction and outputs a new state which is the result. In a standard banking system, for example, the state is a balance sheet, a transaction is a request to move $X from A to B, and the state transition function reduces the value in A's account by $X and increases the value in B's account by $X. If A's account has less than $X in the first place, the state transition function returns an error. Hence, one can formally define:

```
APPLY(S,TX) -> S' or ERROR
```

In the banking system defined above:

```
APPLY({ Alice: $50, Bob: $50 },"send $20 from Alice to Bob") = { Alice: $30, Bob
```

But:

```
APPLY({ Alice: $50, Bob: $50 },"send $70 from Alice to Bob") = ERROR
```

The "state" in Bitcoin is the collection of all coins (technically, "unspent transaction outputs" or UTXO) that have been mined and not yet spent, with each UTXO having a denomination and an owner (defined by a 20-byte address which is essentially a cryptographic public key[fn. 1]). A transaction contains one or more inputs, with each input containing a reference to an existing UTXO and a cryptographic signature produced by the private key associated with the owner's address, and one or more outputs, with each output containing a new UTXO to be added to the state.

The state transition function `APPLY(S,TX) -> S'` can be defined roughly as follows:

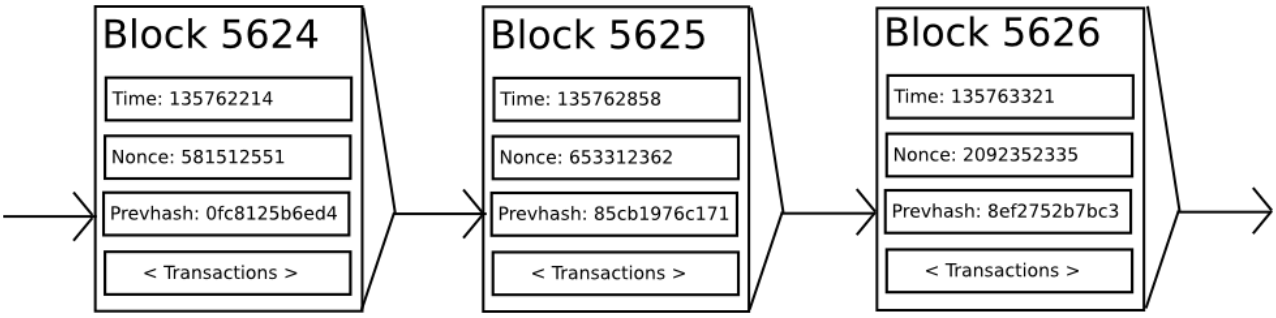
1. For each input in `TX`:
    - If the referenced UTXO is not in `S`, return an error.
    - If the provided signature does not match the owner of the UTXO, return an error.
2. If the sum of the denominations of all input UTXO is less than the sum of the denominations

of all output UTXO, return an error.

3. Return `S'` with all input UTXO removed and all output UTXO added.

The first half of the first step prevents transaction senders from spending coins that do not exist, the second half of the first step prevents transaction senders from spending other people's coins, and the second step enforces conservation of value. In order to use this for payment, the protocol is as follows. Suppose Alice wants to send 11.7 BTC to Bob. First, Alice will look for a set of available UTXO that she owns that totals up to at least 11.7 BTC. Realistically, Alice will not be able to get exactly 11.7 BTC; say that the smallest she can get is 6+4+2=12. She then creates a transaction with those three inputs and two outputs. The first output will be 11.7 BTC with Bob's address as its owner, and the second output will be the remaining 0.3 BTC "change", with the owner being Alice herself.

Mining



If we had access to a trustworthy centralized service, this system would be trivial to implement; it could simply be coded exactly as described, using a centralized server's hard drive to keep track of the state. However, with Bitcoin we are trying to build a decentralized currency system, so we will need to combine the state transition system with a consensus system in order to ensure that everyone agrees on the order of transactions. Bitcoin's decentralized consensus process requires nodes in the network to continuously attempt to produce packages of transactions called "blocks". The network is intended to produce roughly one block every ten minutes, with each block containing a timestamp, a nonce, a reference to (ie. hash of) the previous block and a list of all of the transactions that have taken place since the previous block. Over time, this creates a persistent, ever-growing, "blockchain" that constantly updates to represent the latest state of the Bitcoin ledger.

The algorithm for checking if a block is valid, expressed in this paradigm, is as follows:

1. Check if the previous block referenced by the block exists and is valid.
2. Check that the timestamp of the block is greater than that of the previous block[fn. 2] and less than 2 hours into the future
3. Check that the proof of work on the block is valid.

4. Let `S[0]` be the state at the end of the previous block.
5. Suppose `TX` is the block's transaction list with `n` transactions. For all `i` in `0...n-1`, set `S[i+1] = APPLY(S[i],TX[i])` If any application returns an error, exit and return false.
6. Return true, and register `S[n]` as the state at the end of this block.

Essentially, each transaction in the block must provide a valid state transition from what was the canonical state before the transaction was executed to some new state. Note that the state is not encoded in the block in any way; it is purely an abstraction to be remembered by the validating node and can only be (securely) computed for any block by starting from the genesis state and sequentially applying every transaction in every block. Additionally, note that the order in which the miner includes transactions into the block matters; if there are two transactions A and B in a block such that B spends a UTXO created by A, then the block will be valid if A comes before B but not otherwise.

The one validity condition present in the above list that is not found in other systems is the requirement for "proof of work". The precise condition is that the double-SHA256 hash of every block, treated as a 256-bit number, must be less than a dynamically adjusted target, which as of the time of this writing is approximately 2^{187}. The purpose of this is to make block creation computationally "hard", thereby preventing sybil attackers from remaking the entire blockchain in their favor. Because SHA256 is designed to be a completely unpredictable pseudorandom function, the only way to create a valid block is simply trial and error, repeatedly incrementing the nonce and seeing if the new hash matches.

At the current target of ~2^{187}, the network must make an average of ~2^{69} tries before a valid block is found; in general, the target is recalibrated by the network every 2016 blocks so that on average a new block is produced by some node in the network every ten minutes. In order to compensate miners for this computational work, the miner of every block is entitled to include a transaction giving themselves 12.5 BTC out of nowhere. Additionally, if any transaction has a higher total denomination in its inputs than in its outputs, the difference also goes to the miner as a "transaction fee". Incidentally, this is also the only mechanism by which BTC are issued; the genesis state contained no coins at all.

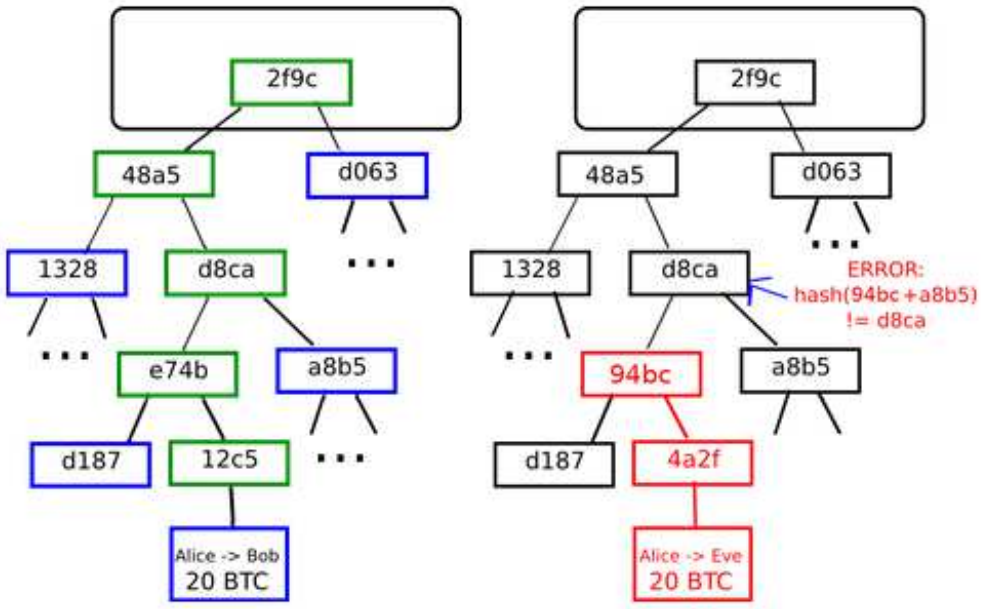
In order to better understand the purpose of mining, let us examine what happens in the event of a malicious attacker. Since Bitcoin's underlying cryptography is known to be secure, the attacker will target the one part of the Bitcoin system that is not protected by cryptography directly: the order of transactions. The attacker's strategy is simple:

1. Send 100 BTC to a merchant in exchange for some product (preferably a rapid-delivery digital good)
2. Wait for the delivery of the product
3. Produce another transaction sending the same 100 BTC to himself
4. Try to convince the network that his transaction to himself was the one that came first.

Once step (1) has taken place, after a few minutes some miner will include the transaction in a

block, say block number 270. After about one hour, five more blocks will have been added to the chain after that block, with each of those blocks indirectly pointing to the transaction and thus "confirming" it. At this point, the merchant will accept the payment as finalized and deliver the product; since we are assuming this is a digital good, delivery is instant. Now, the attacker creates another transaction sending the 100 BTC to himself. If the attacker simply releases it into the wild, the transaction will not be processed; miners will attempt to run `APPLY(S,TX)` and notice that `TX` consumes a UTXO which is no longer in the state. So instead, the attacker creates a "fork" of the blockchain, starting by mining another version of block 270 pointing to the same block 269 as a parent but with the new transaction in place of the old one. Because the block data is different, this requires redoing the proof of work. Furthermore, the attacker's new version of block 270 has a different hash, so the original blocks 271 to 275 do not "point" to it; thus, the original chain and the attacker's new chain are completely separate. The rule is that in a fork the longest blockchain is taken to be the truth, and so legitimate miners will work on the 275 chain while the attacker alone is working on the 270 chain. In order for the attacker to make his blockchain the longest, he would need to have more computational power than the rest of the network combined in order to catch up (hence, "51% attack").

Merkle Trees



Left: it suffices to present only a small number of nodes in a Merkle tree to give a proof of the validity of a branch.

Right: any attempt to change any part of the Merkle tree will eventually lead to an inconsistency somewhere up the chain.

An important scalability feature of Bitcoin is that the block is stored in a multi-level data structure. The "hash" of a block is actually only the hash of the block header, a roughly 200-byte piece of data that contains the timestamp, nonce, previous block hash and the root hash of a data structure called the Merkle tree storing all transactions in the block. A Merkle tree is a type of binary tree, composed of a set of nodes with a large number of leaf nodes at the bottom of the

tree containing the underlying data, a set of intermediate nodes where each node is the hash of its two children, and finally a single root node, also formed from the hash of its two children, representing the "top" of the tree. The purpose of the Merkle tree is to allow the data in a block to be delivered piecemeal: a node can download only the header of a block from one source, the small part of the tree relevant to them from another source, and still be assured that all of the data is correct. The reason why this works is that hashes propagate upward: if a malicious user attempts to swap in a fake transaction into the bottom of a Merkle tree, this change will cause a change in the node above, and then a change in the node above that, finally changing the root of the tree and therefore the hash of the block, causing the protocol to register it as a completely different block (almost certainly with an invalid proof of work).

The Merkle tree protocol is arguably essential to long-term sustainability. A "full node" in the Bitcoin network, one that stores and processes the entirety of every block, takes up about 15 GB of disk space in the Bitcoin network as of April 2014, and is growing by over a gigabyte per month. Currently, this is viable for some desktop computers and not phones, and later on in the future only businesses and hobbyists will be able to participate. A protocol known as "simplified payment verification" (SPV) allows for another class of nodes to exist, called "light nodes", which download the block headers, verify the proof of work on the block headers, and then download only the "branches" associated with transactions that are relevant to them. This allows light nodes to determine with a strong guarantee of security what the status of any Bitcoin transaction, and their current balance, is while downloading only a very small portion of the entire blockchain.

Alternative Blockchain Applications

The idea of taking the underlying blockchain idea and applying it to other concepts also has a long history. In 1998, Nick Szabo came out with the concept of secure property titles with owner authority, a document describing how "new advances in replicated database technology" will allow for a blockchain-based system for storing a registry of who owns what land, creating an elaborate framework including concepts such as homesteading, adverse possession and Georgian land tax. However, there was unfortunately no effective replicated database system available at the time, and so the protocol was never implemented in practice. After 2009, however, once Bitcoin's decentralized consensus was developed a number of alternative applications rapidly began to emerge.

- **Namecoin** - created in 2010, Namecoin is best described as a decentralized name registration database. In decentralized protocols like Tor, Bitcoin and BitMessage, there needs to be some way of identifying accounts so that other people can interact with them, but in all existing solutions the only kind of identifier available is a pseudorandom hash like `1LW79wp5ZBqaHW1jL5TCiBCrhQYtHagUWy`. Ideally, one would like to be able to have an account with a name like "george". However, the problem is that if one person can create an account named "george" then someone else can use the same process to register "george" for themselves as well and impersonate them. The only solution is a first-to-file paradigm,

where the first registerer succeeds and the second fails - a problem perfectly suited for the Bitcoin consensus protocol. Namecoin is the oldest, and most successful, implementation of a name registration system using such an idea.

- **Colored coins** - the purpose of colored coins is to serve as a protocol to allow people to create their own digital currencies - or, in the important trivial case of a currency with one unit, digital tokens, on the Bitcoin blockchain. In the colored coins protocol, one "issues" a new currency by publicly assigning a color to a specific Bitcoin UTXO, and the protocol recursively defines the color of other UTXO to be the same as the color of the inputs that the transaction creating them spent (some special rules apply in the case of mixed-color inputs). This allows users to maintain wallets containing only UTXO of a specific color and send them around much like regular bitcoins, backtracking through the blockchain to determine the color of any UTXO that they receive.
- **Metacoins** - the idea behind a metacoin is to have a protocol that lives on top of Bitcoin, using Bitcoin transactions to store metacoin transactions but having a different state transition function, `APPLY'`. Because the metacoin protocol cannot prevent invalid metacoin transactions from appearing in the Bitcoin blockchain, a rule is added that if `APPLY'(S,TX)` returns an error, the protocol defaults to `APPLY'(S,TX) = S`. This provides an easy mechanism for creating an arbitrary cryptocurrency protocol, potentially with advanced features that cannot be implemented inside of Bitcoin itself, but with a very low development cost since the complexities of mining and networking are already handled by the Bitcoin protocol. Metacoins have been used to implement some classes of financial contracts, name registration and decentralized exchange.

Thus, in general, there are two approaches toward building a consensus protocol: building an independent network, and building a protocol on top of Bitcoin. The former approach, while reasonably successful in the case of applications like Namecoin, is difficult to implement; each individual implementation needs to bootstrap an independent blockchain, as well as building and testing all of the necessary state transition and networking code. Additionally, we predict that the set of applications for decentralized consensus technology will follow a power law distribution where the vast majority of applications would be too small to warrant their own blockchain, and we note that there exist large classes of decentralized applications, particularly decentralized autonomous organizations, that need to interact with each other.

The Bitcoin-based approach, on the other hand, has the flaw that it does not inherit the simplified payment verification features of Bitcoin. SPV works for Bitcoin because it can use blockchain depth as a proxy for validity; at some point, once the ancestors of a transaction go far enough back, it is safe to say that they were legitimately part of the state. Blockchain-based meta-protocols, on the other hand, cannot force the blockchain not to include transactions that are not valid within the context of their own protocols. Hence, a fully secure SPV meta-protocol implementation would need to backward scan all the way to the beginning of the Bitcoin blockchain to determine whether or not certain transactions are valid. Currently, all "light" implementations of Bitcoin-based meta-protocols rely on a trusted server to provide the data, arguably a highly suboptimal result especially when one of the primary purposes of a

cryptocurrency is to eliminate the need for trust.

Scripting

Even without any extensions, the Bitcoin protocol actually does facilitate a weak version of a concept of "smart contracts". UTXO in Bitcoin can be owned not just by a public key, but also by a more complicated script expressed in a simple stack-based programming language. In this paradigm, a transaction spending that UTXO must provide data that satisfies the script. Indeed, even the basic public key ownership mechanism is implemented via a script: the script takes an elliptic curve signature as input, verifies it against the transaction and the address that owns the UTXO, and returns 1 if the verification is successful and 0 otherwise. Other, more complicated, scripts exist for various additional use cases. For example, one can construct a script that requires signatures from two out of a given three private keys to validate ("multisig"), a setup useful for corporate accounts, secure savings accounts and some merchant escrow situations. Scripts can also be used to pay bounties for solutions to computational problems, and one can even construct a script that says something like "this Bitcoin UTXO is yours if you can provide an SPV proof that you sent a Dogecoin transaction of this denomination to me", essentially allowing decentralized cross-cryptocurrency exchange.

However, the scripting language as implemented in Bitcoin has several important limitations:

- **Lack of Turing-completeness** - that is to say, while there is a large subset of computation that the Bitcoin scripting language supports, it does not nearly support everything. The main category that is missing is loops. This is done to avoid infinite loops during transaction verification; theoretically it is a surmountable obstacle for script programmers, since any loop can be simulated by simply repeating the underlying code many times with an if statement, but it does lead to scripts that are very space-inefficient. For example, implementing an alternative elliptic curve signature algorithm would likely require 256 repeated multiplication rounds all individually included in the code.
- **Value-blindness** - there is no way for a UTXO script to provide fine-grained control over the amount that can be withdrawn. For example, one powerful use case of an oracle contract would be a hedging contract, where A and B put in $1000 worth of BTC and after 30 days the script sends $1000 worth of BTC to A and the rest to B. This would require an oracle to determine the value of 1 BTC in USD, but even then it is a massive improvement in terms of trust and infrastructure requirement over the fully centralized solutions that are available now. However, because UTXO are all-or-nothing, the only way to achieve this is through the very inefficient hack of having many UTXO of varying denominations (eg. one UTXO of 2^k for every k up to 30) and having O pick which UTXO to send to A and which to B.
- **Lack of state** - a UTXO can either be spent or unspent; there is no opportunity for multi-stage contracts or scripts which keep any other internal state beyond that. This makes it hard to make multi-stage options contracts, decentralized exchange offers or two-stage cryptographic commitment protocols (necessary for secure computational bounties). It also

means that UTXO can only be used to build simple, one-off contracts and not more complex "stateful" contracts such as decentralized organizations, and makes meta-protocols difficult to implement. Binary state combined with value-blindness also mean that another important application, withdrawal limits, is impossible.

- **Blockchain-blindness** - UTXO are blind to blockchain data such as the nonce, the timestamp and previous block hash. This severely limits applications in gambling, and several other categories, by depriving the scripting language of a potentially valuable source of randomness.

Thus, we see three approaches to building advanced applications on top of cryptocurrency: building a new blockchain, using scripting on top of Bitcoin, and building a meta-protocol on top of Bitcoin. Building a new blockchain allows for unlimited freedom in building a feature set, but at the cost of development time, bootstrapping effort and security. Using scripting is easy to implement and standardize, but is very limited in its capabilities, and meta-protocols, while easy, suffer from faults in scalability. With Ethereum, we intend to build an alternative framework that provides even larger gains in ease of development as well as even stronger light client properties, while at the same time allowing applications to share an economic environment and blockchain security.

Ethereum

The intent of Ethereum is to create an alternative protocol for building decentralized applications, providing a different set of tradeoffs that we believe will be very useful for a large class of decentralized applications, with particular emphasis on situations where rapid development time, security for small and rarely used applications, and the ability of different applications to very efficiently interact, are important. Ethereum does this by building what is essentially the ultimate abstract foundational layer: a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions. A bare-bones version of Namecoin can be written in two lines of code, and other protocols like currencies and reputation systems can be built in under twenty. Smart contracts, cryptographic "boxes" that contain value and only unlock it if certain conditions are met, can also be built on top of the platform, with vastly more power than that offered by Bitcoin scripting because of the added powers of Turing-completeness, value-awareness, blockchain-awareness and state.

Philosophy

The design behind Ethereum is intended to follow the following principles:

1. **Simplicity**: the Ethereum protocol should be as simple as possible, even at the cost of some data storage or time inefficiency.[fn. 3] An average programmer should ideally be able

to follow and implement the entire specification,[fn. 4] so as to fully realize the unprecedented democratizing potential that cryptocurrency brings and further the vision of Ethereum as a protocol that is open to all. Any optimization which adds complexity should not be included unless that optimization provides very substantial benefit.

2. **Universality**: a fundamental part of Ethereum's design philosophy is that Ethereum does not have "features".[fn. 5] Instead, Ethereum provides an internal Turing-complete scripting language, which a programmer can use to construct any smart contract or transaction type that can be mathematically defined. Want to invent your own financial derivative? With Ethereum, you can. Want to make your own currency? Set it up as an Ethereum contract. Want to set up a full-scale Daemon or Skynet? You may need to have a few thousand interlocking contracts, and be sure to feed them generously, to do that, but nothing is stopping you with Ethereum at your fingertips.
3. **Modularity**: the parts of the Ethereum protocol should be designed to be as modular and separable as possible. Over the course of development, our goal is to create a program where if one was to make a small protocol modification in one place, the application stack would continue to function without any further modification. Innovations such as Ethash (see the Yellow Paper Appendix or wiki article), modified Patricia trees (Yellow Paper, wiki) and RLP (YP, wiki) should be, and are, implemented as separate, feature-complete libraries. This is so that even though they are used in Ethereum, even if Ethereum does not require certain features, such features are still usable in other protocols as well. Ethereum development should be maximally done so as to benefit the entire cryptocurrency ecosystem, not just itself.
4. **Agility**: details of the Ethereum protocol are not set in stone. Although we will be extremely judicious about making modifications to high-level constructs, for instance with the sharding roadmap, abstracting execution, with only data availability enshrined in consensus. Computational tests later on in the development process may lead us to discover that certain modifications, e.g. to the protocol architecture or to the Ethereum Virtual Machine (EVM), will substantially improve scalability or security. If any such opportunities are found, we will exploit them.
5. **Non-discrimination** and **non-censorship**: the protocol should not attempt to actively restrict or prevent specific categories of usage. All regulatory mechanisms in the protocol should be designed to directly regulate the harm and not attempt to oppose specific undesirable applications. A programmer can even run an infinite loop script on top of Ethereum for as long as they are willing to keep paying the per-computational-step transaction fee.

Ethereum Accounts

In Ethereum, the state is made up of objects called "accounts", with each account having a 20-byte address and state transitions being direct transfers of value and information between accounts. An Ethereum account contains four fields:

- The **nonce**, a counter used to make sure each transaction can only be processed once
- The account's current **ether balance**
- The account's **contract code**, if present
- The account's **storage** (empty by default)

"Ether" is the main internal crypto-fuel of Ethereum, and is used to pay transaction fees. In general, there are two types of accounts: **externally owned accounts**, controlled by private keys, and **contract accounts**, controlled by their contract code. An externally owned account has no code, and one can send messages from an externally owned account by creating and signing a transaction; in a contract account, every time the contract account receives a message its code activates, allowing it to read and write to internal storage and send other messages or create contracts in turn.

Note that "contracts" in Ethereum should not be seen as something that should be "fulfilled" or "complied with"; rather, they are more like "autonomous agents" that live inside of the Ethereum execution environment, always executing a specific piece of code when "poked" by a message or transaction, and having direct control over their own ether balance and their own key/value store to keep track of persistent variables.

Messages and Transactions

The term "transaction" is used in Ethereum to refer to the signed data package that stores a message to be sent from an externally owned account. Transactions contain:

- The recipient of the message
- A signature identifying the sender
- The amount of ether to transfer from the sender to the recipient
- An optional data field
- A `STARTGAS` value, representing the maximum number of computational steps the transaction execution is allowed to take
- A `GASPRICE` value, representing the fee the sender pays per computational step

The first three are standard fields expected in any cryptocurrency. The data field has no function by default, but the virtual machine has an opcode which a contract can use to access the data; as an example use case, if a contract is functioning as an on-blockchain domain registration service, then it may wish to interpret the data being passed to it as containing two "fields", the first field being a domain to register and the second field being the IP address to register it to. The contract would read these values from the message data and appropriately place them in storage.

The `STARTGAS` and `GASPRICE` fields are crucial for Ethereum's anti-denial of service model. In order to prevent accidental or hostile infinite loops or other computational wastage in code, each transaction is required to set a limit to how many computational steps of code execution it can

use. The fundamental unit of computation is "gas"; usually, a computational step costs 1 gas, but some operations cost higher amounts of gas because they are more computationally expensive, or increase the amount of data that must be stored as part of the state. There is also a fee of 5 gas for every byte in the transaction data. The intent of the fee system is to require an attacker to pay proportionately for every resource that they consume, including computation, bandwidth and storage; hence, any transaction that leads to the network consuming a greater amount of any of these resources must have a gas fee roughly proportional to the increment.

Messages

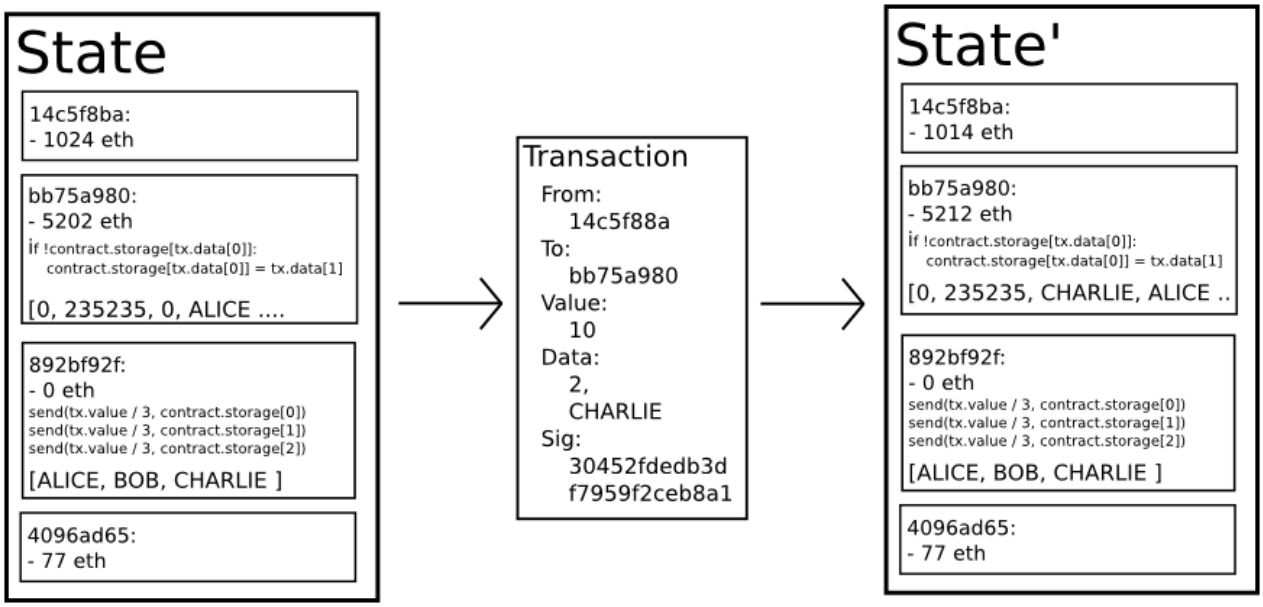
Contracts have the ability to send "messages" to other contracts. Messages are virtual objects that are never serialized and exist only in the Ethereum execution environment. A message contains:

- The sender of the message (implicit)
- The recipient of the message
- The amount of ether to transfer alongside the message
- An optional data field
- A `STARTGAS` value

Essentially, a message is like a transaction, except it is produced by a contract and not an external actor. A message is produced when a contract currently executing code executes the `CALL` opcode, which produces and executes a message. Like a transaction, a message leads to the recipient account running its code. Thus, contracts can have relationships with other contracts in exactly the same way that external actors can.

Note that the gas allowance assigned by a transaction or contract applies to the total gas consumed by that transaction and all sub-executions. For example, if an external actor A sends a transaction to B with 1000 gas, and B consumes 600 gas before sending a message to C, and the internal execution of C consumes 300 gas before returning, then B can spend another 100 gas before running out of gas.

Ethereum State Transition Function



The Ethereum state transition function, `APPLY(S,TX) -> S'` can be defined as follows:

1. Check if the transaction is well-formed (ie. has the right number of values), the signature is valid, and the nonce matches the nonce in the sender's account. If not, return an error.
2. Calculate the transaction fee as `STARTGAS * GASPRICE`, and determine the sending address from the signature. Subtract the fee from the sender's account balance and increment the sender's nonce. If there is not enough balance to spend, return an error.
3. Initialize `GAS = STARTGAS`, and take off a certain quantity of gas per byte to pay for the bytes in the transaction.
4. Transfer the transaction value from the sender's account to the receiving account. If the receiving account does not yet exist, create it. If the receiving account is a contract, run the contract's code either to completion or until the execution runs out of gas.
5. If the value transfer failed because the sender did not have enough money, or the code execution ran out of gas, revert all state changes except the payment of the fees, and add the fees to the miner's account.
6. Otherwise, refund the fees for all remaining gas to the sender, and send the fees paid for gas consumed to the miner.

For example, suppose that the contract's code is:

```
if !self.storage[calldataload(0)]:
    self.storage[calldataload(0)] = calldataload(32)
```

Note that in reality the contract code is written in the low-level EVM code; this example is written in Serpent, one of our high-level languages, for clarity, and can be compiled down to EVM code. Suppose that the contract's storage starts off empty, and a transaction is sent with 10 ether

value, 2000 gas, 0.001 ether gasprice, and 64 bytes of data, with bytes 0-31 representing the number `2` and bytes 32-63 representing the string `CHARLIE`.[fn. 6] The process for the state transition function in this case is as follows:

1. Check that the transaction is valid and well formed.
2. Check that the transaction sender has at least 2000 * 0.001 = 2 ether. If it is, then subtract 2 ether from the sender's account.
3. Initialize gas = 2000; assuming the transaction is 170 bytes long and the byte-fee is 5, subtract 850 so that there is 1150 gas left.
4. Subtract 10 more ether from the sender's account, and add it to the contract's account.
5. Run the code. In this case, this is simple: it checks if the contract's storage at index `2` is used, notices that it is not, and so it sets the storage at index `2` to the value `CHARLIE`. Suppose this takes 187 gas, so the remaining amount of gas is 1150 - 187 = 963
6. Add 963 * 0.001 = 0.963 ether back to the sender's account, and return the resulting state.

If there was no contract at the receiving end of the transaction, then the total transaction fee would simply be equal to the provided `GASPRICE` multiplied by the length of the transaction in bytes, and the data sent alongside the transaction would be irrelevant.

Note that messages work equivalently to transactions in terms of reverts: if a message execution runs out of gas, then that message's execution, and all other executions triggered by that execution, revert, but parent executions do not need to revert. This means that it is "safe" for a contract to call another contract, as if A calls B with G gas then A's execution is guaranteed to lose at most G gas. Finally, note that there is an opcode, `CREATE`, that creates a contract; its execution mechanics are generally similar to `CALL`, with the exception that the output of the execution determines the code of a newly created contract.

Code Execution

The code in Ethereum contracts is written in a low-level, stack-based bytecode language, referred to as "Ethereum virtual machine code" or "EVM code". The code consists of a series of bytes, where each byte represents an operation. In general, code execution is an infinite loop that consists of repeatedly carrying out the operation at the current program counter (which begins at zero) and then incrementing the program counter by one, until the end of the code is reached or an error or `STOP` or `RETURN` instruction is detected. The operations have access to three types of space in which to store data:

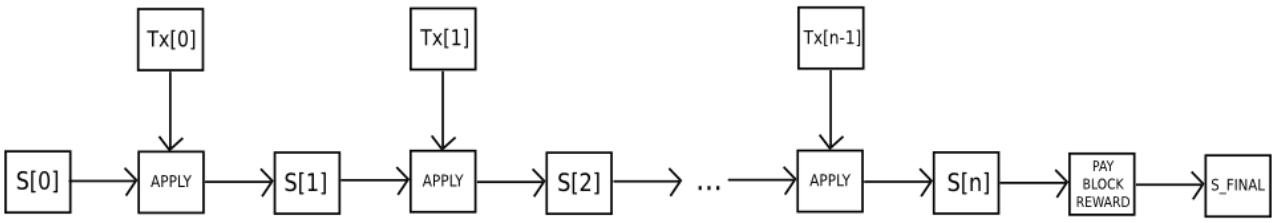
- The **stack**, a last-in-first-out container to which values can be pushed and popped
- **Memory**, an infinitely expandable byte array
- The contract's long-term **storage**, a key/value store. Unlike stack and memory, which reset after computation ends, storage persists for the long term.

The code can also access the value, sender and data of the incoming message, as well as block

header data, and the code can also return a byte array of data as an output.

The formal execution model of EVM code is surprisingly simple. While the Ethereum virtual machine is running, its full computational state can be defined by the tuple `(block_state, transaction, message, code, memory, stack, pc, gas)`, where `block_state` is the global state containing all accounts and includes balances and storage. At the start of every round of execution, the current instruction is found by taking the `pc`-th byte of `code` (or 0 if `pc >= len(code)`), and each instruction has its own definition in terms of how it affects the tuple. For example, `ADD` pops two items off the stack and pushes their sum, reduces `gas` by 1 and increments `pc` by 1, and `SSTORE` pops the top two items off the stack and inserts the second item into the contract's storage at the index specified by the first item. Although there are many ways to optimize Ethereum virtual machine execution via just-in-time compilation, a basic implementation of Ethereum can be done in a few hundred lines of code.

Blockchain and Mining



The Ethereum blockchain is in many ways similar to the Bitcoin blockchain, although it does have some differences. The main difference between Ethereum and Bitcoin with regard to the blockchain architecture is that, unlike Bitcoin(which only contains a copy of the transaction list), Ethereum blocks contain a copy of both the transaction list and the most recent state. Aside from that, two other values, the block number and the difficulty, are also stored in the block. The basic block validation algorithm in Ethereum is as follows:

1. Check if the previous block referenced exists and is valid.
2. Check that the timestamp of the block is greater than that of the referenced previous block and less than 15 minutes into the future
3. Check that the block number, difficulty, transaction root, uncle root and gas limit (various low-level Ethereum-specific concepts) are valid.
4. Check that the proof of work on the block is valid.
5. Let `S[0]` be the state at the end of the previous block.
6. Let `TX` be the block's transaction list, with `n` transactions. For all `i` in `0...n-1`, set `S[i+1] = APPLY(S[i],TX[i])`. If any application returns an error, or if the total gas consumed in the block up until this point exceeds the `GASLIMIT`, return an error.
7. Let `S_FINAL` be `S[n]`, but adding the block reward paid to the miner.

8. Check if the Merkle tree root of the state `S_FINAL` is equal to the final state root provided in the block header. If it is, the block is valid; otherwise, it is not valid.

The approach may seem highly inefficient at first glance, because it needs to store the entire state with each block, but in reality efficiency should be comparable to that of Bitcoin. The reason is that the state is stored in the tree structure, and after every block only a small part of the tree needs to be changed. Thus, in general, between two adjacent blocks the vast majority of the tree should be the same, and therefore the data can be stored once and referenced twice using pointers (ie. hashes of subtrees). A special kind of tree known as a "Patricia tree" is used to accomplish this, including a modification to the Merkle tree concept that allows for nodes to be inserted and deleted, and not just changed, efficiently. Additionally, because all of the state information is part of the last block, there is no need to store the entire blockchain history - a strategy which, if it could be applied to Bitcoin, can be calculated to provide 5-20x savings in space.

A commonly asked question is "where" contract code is executed, in terms of physical hardware. This has a simple answer: the process of executing contract code is part of the definition of the state transition function, which is part of the block validation algorithm, so if a transaction is added into block `B` the code execution spawned by that transaction will be executed by all nodes, now and in the future, that download and validate block `B`.

Applications

In general, there are three types of applications on top of Ethereum. The first category is financial applications, providing users with more powerful ways of managing and entering into contracts using their money. This includes sub-currencies, financial derivatives, hedging contracts, savings wallets, wills, and ultimately even some classes of full-scale employment contracts. The second category is semi-financial applications, where money is involved but there is also a heavy non-monetary side to what is being done; a perfect example is self-enforcing bounties for solutions to computational problems. Finally, there are applications such as online voting and decentralized governance that are not financial at all.

Token Systems

On-blockchain token systems have many applications ranging from sub-currencies representing assets such as USD or gold to company stocks, individual tokens representing smart property, secure unforgeable coupons, and even token systems with no ties to conventional value at all, used as point systems for incentivization. Token systems are surprisingly easy to implement in Ethereum. The key point to understand is that a currency, or token system, fundamentally is a database with one operation: subtract X units from A and give X units to B, with the provision that (1) A had at least X units before the transaction and (2) the transaction is approved by A. All that it takes to implement a token system is to implement this logic into a contract.

The basic code for implementing a token system in Serpent looks as follows:

```
def send(to, value):
    if self.storage[msg.sender] >= value:
        self.storage[msg.sender] = self.storage[msg.sender] - value
        self.storage[to] = self.storage[to] + value
```

This is essentially a literal implementation of the "banking system" state transition function described further above in this document. A few extra lines of code need to be added to provide for the initial step of distributing the currency units in the first place and a few other edge cases, and ideally a function would be added to let other contracts query for the balance of an address. But that's all there is to it. Theoretically, Ethereum-based token systems acting as sub-currencies can potentially include another important feature that on-chain Bitcoin-based meta-currencies lack: the ability to pay transaction fees directly in that currency. The way this would be implemented is that the contract would maintain an ether balance with which it would refund ether used to pay fees to the sender, and it would refill this balance by collecting the internal currency units that it takes in fees and reselling them in a constant running auction. Users would thus need to "activate" their accounts with ether, but once the ether is there it would be reusable because the contract would refund it each time.

Financial derivatives and Stable-Value Currencies

Financial derivatives are the most common application of a "smart contract", and one of the simplest to implement in code. The main challenge in implementing financial contracts is that the majority of them require reference to an external price ticker; for example, a very desirable application is a smart contract that hedges against the volatility of ether (or another cryptocurrency) with respect to the US dollar, but doing this requires the contract to know what the value of ETH/USD is. The simplest way to do this is through a "data feed" contract maintained by a specific party (eg. NASDAQ) designed so that that party has the ability to update the contract as needed, and providing an interface that allows other contracts to send a message to that contract and get back a response that provides the price.

Given that critical ingredient, the hedging contract would look as follows:

1. Wait for party A to input 1000 ether.
2. Wait for party B to input 1000 ether.
3. Record the USD value of 1000 ether, calculated by querying the data feed contract, in storage, say this is $x.
4. After 30 days, allow A or B to "reactivate" the contract in order to send $x worth of ether (calculated by querying the data feed contract again to get the new price) to A and the rest to B.

Such a contract would have significant potential in crypto-commerce. One of the main problems

cited about cryptocurrency is the fact that it's volatile; although many users and merchants may want the security and convenience of dealing with cryptographic assets, they may not wish to face that prospect of losing 23% of the value of their funds in a single day. Up until now, the most commonly proposed solution has been issuer-backed assets; the idea is that an issuer creates a sub-currency in which they have the right to issue and revoke units, and provide one unit of the currency to anyone who provides them (offline) with one unit of a specified underlying asset (eg. gold, USD). The issuer then promises to provide one unit of the underlying asset to anyone who sends back one unit of the crypto-asset. This mechanism allows any non-cryptographic asset to be "uplifted" into a cryptographic asset, provided that the issuer can be trusted.

In practice, however, issuers are not always trustworthy, and in some cases the banking infrastructure is too weak, or too hostile, for such services to exist. Financial derivatives provide an alternative. Here, instead of a single issuer providing the funds to back up an asset, a decentralized market of speculators, betting that the price of a cryptographic reference asset (eg. ETH) will go up, plays that role. Unlike issuers, speculators have no option to default on their side of the bargain because the hedging contract holds their funds in escrow. Note that this approach is not fully decentralized, because a trusted source is still needed to provide the price ticker, although arguably even still this is a massive improvement in terms of reducing infrastructure requirements (unlike being an issuer, issuing a price feed requires no licenses and can likely be categorized as free speech) and reducing the potential for fraud.

Identity and Reputation Systems

The earliest alternative cryptocurrency of all, Namecoin, attempted to use a Bitcoin-like blockchain to provide a name registration system, where users can register their names in a public database alongside other data. The major cited use case is for a DNS system, mapping domain names like "bitcoin.org" (or, in Namecoin's case, "bitcoin.bit") to an IP address. Other use cases include email authentication and potentially more advanced reputation systems. Here is the basic contract to provide a Namecoin-like name registration system on Ethereum:

```
def register(name, value):
    if !self.storage[name]:
        self.storage[name] = value
```

The contract is very simple; all it is a database inside the Ethereum network that can be added to, but not modified or removed from. Anyone can register a name with some value, and that registration then sticks forever. A more sophisticated name registration contract will also have a "function clause" allowing other contracts to query it, as well as a mechanism for the "owner" (ie. the first registerer) of a name to change the data or transfer ownership. One can even add reputation and web-of-trust functionality on top.

Decentralized File Storage

Over the past few years, there have emerged a number of popular online file storage startups, the most prominent being Dropbox, seeking to allow users to upload a backup of their hard drive and have the service store the backup and allow the user to access it in exchange for a monthly fee. However, at this point the file storage market is at times relatively inefficient; a cursory look at various existing solutions shows that, particularly at the "uncanny valley" 20-200 GB level at which neither free quotas nor enterprise-level discounts kick in, monthly prices for mainstream file storage costs are such that you are paying for more than the cost of the entire hard drive in a single month. Ethereum contracts can allow for the development of a decentralized file storage ecosystem, where individual users can earn small quantities of money by renting out their own hard drives and unused space can be used to further drive down the costs of file storage.

The key underpinning piece of such a device would be what we have termed the "decentralized Dropbox contract". This contract works as follows. First, one splits the desired data up into blocks, encrypting each block for privacy, and builds a Merkle tree out of it. One then makes a contract with the rule that, every N blocks, the contract would pick a random index in the Merkle tree (using the previous block hash, accessible from contract code, as a source of randomness), and give X ether to the first entity to supply a transaction with a simplified payment verification-like proof of ownership of the block at that particular index in the tree. When a user wants to re-download their file, they can use a micropayment channel protocol (eg. pay 1 szabo per 32 kilobytes) to recover the file; the most fee-efficient approach is for the payer not to publish the transaction until the end, instead replacing the transaction with a slightly more lucrative one with the same nonce after every 32 kilobytes.

An important feature of the protocol is that, although it may seem like one is trusting many random nodes not to decide to forget the file, one can reduce that risk down to near-zero by splitting the file into many pieces via secret sharing, and watching the contracts to see each piece is still in some node's possession. If a contract is still paying out money, that provides a cryptographic proof that someone out there is still storing the file.

Decentralized Autonomous Organizations

The general concept of a "decentralized autonomous organization" is that of a virtual entity that has a certain set of members or shareholders which, perhaps with a 67% majority, have the right to spend the entity's funds and modify its code. The members would collectively decide on how the organization should allocate its funds. Methods for allocating a DAO's funds could range from bounties, salaries to even more exotic mechanisms such as an internal currency to reward work. This essentially replicates the legal trappings of a traditional company or nonprofit but using only cryptographic blockchain technology for enforcement. So far much of the talk around DAOs has been around the "capitalist" model of a "decentralized autonomous corporation" (DAC) with dividend-receiving shareholders and tradable shares; an alternative, perhaps described as a "decentralized autonomous community", would have all members have an equal

share in the decision making and require 67% of existing members to agree to add or remove a member. The requirement that one person can only have one membership would then need to be enforced collectively by the group.

A general outline for how to code a DAO is as follows. The simplest design is simply a piece of self-modifying code that changes if two thirds of members agree on a change. Although code is theoretically immutable, one can easily get around this and have de-facto mutability by having chunks of the code in separate contracts, and having the address of which contracts to call stored in the modifiable storage. In a simple implementation of such a DAO contract, there would be three transaction types, distinguished by the data provided in the transaction:

- `[0,i,K,V]` to register a proposal with index `i` to change the address at storage index `K` to value `V`
- `[1,i]` to register a vote in favor of proposal `i`
- `[2,i]` to finalize proposal `i` if enough votes have been made

The contract would then have clauses for each of these. It would maintain a record of all open storage changes, along with a list of who voted for them. It would also have a list of all members. When any storage change gets to two thirds of members voting for it, a finalizing transaction could execute the change. A more sophisticated skeleton would also have built-in voting ability for features like sending a transaction, adding members and removing members, and may even provide for Liquid Democracy-style vote delegation (ie. anyone can assign someone to vote for them, and assignment is transitive so if A assigns B and B assigns C then C determines A's vote). This design would allow the DAO to grow organically as a decentralized community, allowing people to eventually delegate the task of filtering out who is a member to specialists, although unlike in the "current system" specialists can easily pop in and out of existence over time as individual community members change their alignments.

An alternative model is for a decentralized corporation, where any account can have zero or more shares, and two thirds of the shares are required to make a decision. A complete skeleton would involve asset management functionality, the ability to make an offer to buy or sell shares, and the ability to accept offers (preferably with an order-matching mechanism inside the contract). Delegation would also exist Liquid Democracy-style, generalizing the concept of a "board of directors".

Further Applications

1. Savings wallets. Suppose that Alice wants to keep her funds safe, but is worried that she will lose or someone will hack her private key. She puts ether into a contract with Bob, a bank, as follows:

- Alice alone can withdraw a maximum of 1% of the funds per day.
- Bob alone can withdraw a maximum of 1% of the funds per day, but Alice has the ability to make a transaction with her key shutting off this ability.

- Alice and Bob together can withdraw anything.

Normally, 1% per day is enough for Alice, and if Alice wants to withdraw more she can contact Bob for help. If Alice's key gets hacked, she runs to Bob to move the funds to a new contract. If she loses her key, Bob will get the funds out eventually. If Bob turns out to be malicious, then she can turn off his ability to withdraw.

2. Crop insurance. One can easily make a financial derivatives contract but using a data feed of the weather instead of any price index. If a farmer in Iowa purchases a derivative that pays out inversely based on the precipitation in Iowa, then if there is a drought, the farmer will automatically receive money and if there is enough rain the farmer will be happy because their crops would do well. This can be expanded to natural disaster insurance generally.

3. A decentralized data feed. For financial contracts for difference, it may actually be possible to decentralize the data feed via a protocol called SchellingCoin. SchellingCoin basically works as follows: N parties all put into the system the value of a given datum (eg. the ETH/USD price), the values are sorted, and everyone between the 25th and 75th percentile gets one token as a reward. Everyone has the incentive to provide the answer that everyone else will provide, and the only value that a large number of players can realistically agree on is the obvious default: the truth. This creates a decentralized protocol that can theoretically provide any number of values, including the ETH/USD price, the temperature in Berlin or even the result of a particular hard computation.

4. Smart multisignature escrow. Bitcoin allows multisignature transaction contracts where, for example, three out of a given five keys can spend the funds. Ethereum allows for more granularity; for example, four out of five can spend everything, three out of five can spend up to 10% per day, and two out of five can spend up to 0.5% per day. Additionally, Ethereum multisig is asynchronous - two parties can register their signatures on the blockchain at different times and the last signature will automatically send the transaction.

5. Cloud computing. The EVM technology can also be used to create a verifiable computing environment, allowing users to ask others to carry out computations and then optionally ask for proofs that computations at certain randomly selected checkpoints were done correctly. This allows for the creation of a cloud computing market where any user can participate with their desktop, laptop or specialized server, and spot-checking together with security deposits can be used to ensure that the system is trustworthy (ie. nodes cannot profitably cheat). Although such a system may not be suitable for all tasks; tasks that require a high level of inter-process communication, for example, cannot easily be done on a large cloud of nodes. Other tasks, however, are much easier to parallelize; projects like SETI@home, folding@home and genetic algorithms can easily be implemented on top of such a platform.

6. Peer-to-peer gambling. Any number of peer-to-peer gambling protocols, such as Frank Stajano and Richard Clayton's Cyberdice, can be implemented on the Ethereum blockchain. The simplest gambling protocol is actually simply a contract for difference on the next block hash, and more advanced protocols can be built up from there, creating gambling services with near-

zero fees that have no ability to cheat.

7. Prediction markets. Provided an oracle or SchellingCoin, prediction markets are also easy to implement, and prediction markets together with SchellingCoin may prove to be the first mainstream application of futarchy as a governance protocol for decentralized organizations.

8. On-chain decentralized marketplaces, using the identity and reputation system as a base.

Miscellanea And Concerns

Modified GHOST Implementation

The "Greedy Heaviest Observed Subtree" (GHOST) protocol is an innovation first introduced by Yonatan Sompolinsky and Aviv Zohar in December 2013. The motivation behind GHOST is that blockchains with fast confirmation times currently suffer from reduced security due to a high stale rate - because blocks take a certain time to propagate through the network, if miner A mines a block and then miner B happens to mine another block before miner A's block propagates to B, miner B's block will end up wasted and will not contribute to network security. Furthermore, there is a centralization issue: if miner A is a mining pool with 30% hashpower and B has 10% hashpower, A will have a risk of producing a stale block 70% of the time (since the other 30% of the time A produced the last block and so will get mining data immediately) whereas B will have a risk of producing a stale block 90% of the time. Thus, if the block interval is short enough for the stale rate to be high, A will be substantially more efficient simply by virtue of its size. With these two effects combined, blockchains which produce blocks quickly are very likely to lead to one mining pool having a large enough percentage of the network hashpower to have de facto control over the mining process.

As described by Sompolinsky and Zohar, GHOST solves the first issue of network security loss by including stale blocks in the calculation of which chain is the "longest"; that is to say, not just the parent and further ancestors of a block, but also the stale descendants of the block's ancestor (in Ethereum jargon, "uncles") are added to the calculation of which block has the largest total proof of work backing it. To solve the second issue of centralization bias, we go beyond the protocol described by Sompolinsky and Zohar, and also provide block rewards to stales: a stale block receives 87.5% of its base reward, and the nephew that includes the stale block receives the remaining 12.5%. Transaction fees, however, are not awarded to uncles.

Ethereum implements a simplified version of GHOST which only goes down seven levels. Specifically, it is defined as follows:

- A block must specify a parent, and it must specify 0 or more uncles
- An uncle included in block `B` must have the following properties:
  - It must be a direct child of the `k`-th generation ancestor of `B`, where `2 <= k <= 7`.
  - It cannot be an ancestor of `B`

    - An uncle must be a valid block header, but does not need to be a previously verified or even valid block
    - An uncle must be different from all uncles included in previous blocks and all other uncles included in the same block (non-double-inclusion)
- For every uncle `U` in block `B`, the miner of `B` gets an additional 3.125% added to its coinbase reward and the miner of U gets 93.75% of a standard coinbase reward.

This limited version of GHOST, with uncles includable only up to 7 generations, was used for two reasons. First, unlimited GHOST would include too many complications into the calculation of which uncles for a given block are valid. Second, unlimited GHOST with compensation as used in Ethereum removes the incentive for a miner to mine on the main chain and not the chain of a public attacker.

Fees

Because every transaction published into the blockchain imposes on the network the cost of needing to download and verify it, there is a need for some regulatory mechanism, typically involving transaction fees, to prevent abuse. The default approach, used in Bitcoin, is to have purely voluntary fees, relying on miners to act as the gatekeepers and set dynamic minimums. This approach has been received very favorably in the Bitcoin community particularly because it is "market-based", allowing supply and demand between miners and transaction senders determine the price. The problem with this line of reasoning is, however, that transaction processing is not a market; although it is intuitively attractive to construe transaction processing as a service that the miner is offering to the sender, in reality every transaction that a miner includes will need to be processed by every node in the network, so the vast majority of the cost of transaction processing is borne by third parties and not the miner that is making the decision of whether or not to include it. Hence, tragedy-of-the-commons problems are very likely to occur.

However, as it turns out this flaw in the market-based mechanism, when given a particular inaccurate simplifying assumption, magically cancels itself out. The argument is as follows. Suppose that:

1. A transaction leads to `k` operations, offering the reward `kR` to any miner that includes it where `R` is set by the sender and `k` and `R` are (roughly) visible to the miner beforehand.
2. An operation has a processing cost of `C` to any node (ie. all nodes have equal efficiency)
3. There are `N` mining nodes, each with exactly equal processing power (ie. `1/N` of total)
4. No non-mining full nodes exist.

A miner would be willing to process a transaction if the expected reward is greater than the cost. Thus, the expected reward is `kR/N` since the miner has a `1/N` chance of processing the next block, and the processing cost for the miner is simply `kC`. Hence, miners will include transactions where `kR/N > kC`, or `R > NC`. Note that `R` is the per-operation fee provided by the sender, and is thus a lower bound on the benefit that the sender derives from the

transaction, and `NC` is the cost to the entire network together of processing an operation. Hence, miners have the incentive to include only those transactions for which the total utilitarian benefit exceeds the cost.

However, there are several important deviations from those assumptions in reality:

1. The miner does pay a higher cost to process the transaction than the other verifying nodes, since the extra verification time delays block propagation and thus increases the chance the block will become a stale.
2. There do exist non-mining full nodes.
3. The mining power distribution may end up radically inegalitarian in practice.
4. Speculators, political enemies and crazies whose utility function includes causing harm to the network do exist, and they can cleverly set up contracts where their cost is much lower than the cost paid by other verifying nodes.

(1) provides a tendency for the miner to include fewer transactions, and (2) increases `NC`; hence, these two effects at least partially cancel each other out.[How?] (3) and (4) are the major issue; to solve them we simply institute a floating cap: no block can have more operations than `BLK_LIMIT_FACTOR` times the long-term exponential moving average. Specifically:

```
blk.oplimit = floor((blk.parent.oplimit * (EMAFACTOR - 1) + floor(parent.opcount
```

`BLK_LIMIT_FACTOR` and `EMA_FACTOR` are constants that will be set to 65536 and 1.5 for the time being, but will likely be changed after further analysis.

There is another factor disincentivizing large block sizes in Bitcoin: blocks that are large will take longer to propagate, and thus have a higher probability of becoming stales. In Ethereum, highly gas-consuming blocks can also take longer to propagate both because they are physically larger and because they take longer to process the transaction state transitions to validate. This delay disincentive is a significant consideration in Bitcoin, but less so in Ethereum because of the GHOST protocol; hence, relying on regulated block limits provides a more stable baseline.

Computation And Turing-Completeness

An important note is that the Ethereum virtual machine is Turing-complete; this means that EVM code can encode any computation that can be conceivably carried out, including infinite loops. EVM code allows looping in two ways. First, there is a `JUMP` instruction that allows the program to jump back to a previous spot in the code, and a `JUMPI` instruction to do conditional jumping, allowing for statements like `while x < 27: x = x * 2`. Second, contracts can call other contracts, potentially allowing for looping through recursion. This naturally leads to a problem: can malicious users essentially shut miners and full nodes down by forcing them to enter into an infinite loop? The issue arises because of a problem in computer science known as the halting

problem: there is no way to tell, in the general case, whether or not a given program will ever halt.

As described in the state transition section, our solution works by requiring a transaction to set a maximum number of computational steps that it is allowed to take, and if execution takes longer computation is reverted but fees are still paid. Messages work in the same way. To show the motivation behind our solution, consider the following examples:

- An attacker creates a contract which runs an infinite loop, and then sends a transaction activating that loop to the miner. The miner will process the transaction, running the infinite loop, and wait for it to run out of gas. Even though the execution runs out of gas and stops halfway through, the transaction is still valid and the miner still claims the fee from the attacker for each computational step.
- An attacker creates a very long infinite loop with the intent of forcing the miner to keep computing for such a long time that by the time computation finishes a few more blocks will have come out and it will not be possible for the miner to include the transaction to claim the fee. However, the attacker will be required to submit a value for `STARTGAS` limiting the number of computational steps that execution can take, so the miner will know ahead of time that the computation will take an excessively large number of steps.
- An attacker sees a contract with code of some form like `send(A,contract.storage[A]); contract.storage[A] = 0`, and sends a transaction with just enough gas to run the first step but not the second (ie. making a withdrawal but not letting the balance go down). The contract author does not need to worry about protecting against such attacks, because if execution stops halfway through the changes they get reverted.
- A financial contract works by taking the median of nine proprietary data feeds in order to minimize risk. An attacker takes over one of the data feeds, which is designed to be modifiable via the variable-address-call mechanism described in the section on DAOs, and converts it to run an infinite loop, thereby attempting to force any attempts to claim funds from the financial contract to run out of gas. However, the financial contract can set a gas limit on the message to prevent this problem.

The alternative to Turing-completeness is Turing-incompleteness, where `JUMP` and `JUMPI` do not exist and only one copy of each contract is allowed to exist in the call stack at any given time. With this system, the fee system described and the uncertainties around the effectiveness of our solution might not be necessary, as the cost of executing a contract would be bounded above by its size. Additionally, Turing-incompleteness is not even that big a limitation; out of all the contract examples we have conceived internally, so far only one required a loop, and even that loop could be removed by making 26 repetitions of a one-line piece of code. Given the serious implications of Turing-completeness, and the limited benefit, why not simply have a Turing-incomplete language? In reality, however, Turing-incompleteness is far from a neat solution to the problem. To see why, consider the following contracts:

```
C0: call(C1); call(C1);
C1: call(C2); call(C2);
C2: call(C3); call(C3);
...
C49: call(C50); call(C50);
C50: (run one step of a program and record the change in storage)
```

Now, send a transaction to A. Thus, in 51 transactions, we have a contract that takes up 2^{50} computational steps. Miners could try to detect such logic bombs ahead of time by maintaining a value alongside each contract specifying the maximum number of computational steps that it can take, and calculating this for contracts calling other contracts recursively, but that would require miners to forbid contracts that create other contracts (since the creation and execution of all 26 contracts above could easily be rolled into a single contract). Another problematic point is that the address field of a message is a variable, so in general it may not even be possible to tell which other contracts a given contract will call ahead of time. Hence, all in all, we have a surprising conclusion: Turing-completeness is surprisingly easy to manage, and the lack of Turing-completeness is equally surprisingly difficult to manage unless the exact same controls are in place - but in that case why not just let the protocol be Turing-complete?

Currency And Issuance

The Ethereum network includes its own built-in currency, ether, which serves the dual purpose of providing a primary liquidity layer to allow for efficient exchange between various types of digital assets and, more importantly, of providing a mechanism for paying transaction fees. For convenience and to avoid future argument (see the current mBTC/uBTC/satoshi debate in Bitcoin), the denominations will be pre-labelled:

- 1: wei
- 10^{12}: szabo
- 10^{15}: finney
- 10^{18}: ether

This should be taken as an expanded version of the concept of "dollars" and "cents" or "BTC" and "satoshi". In the near future, we expect "ether" to be used for ordinary transactions, "finney" for microtransactions and "szabo" and "wei" for technical discussions around fees and protocol implementation; the remaining denominations may become useful later and should not be included in clients at this point.

The issuance model will be as follows:

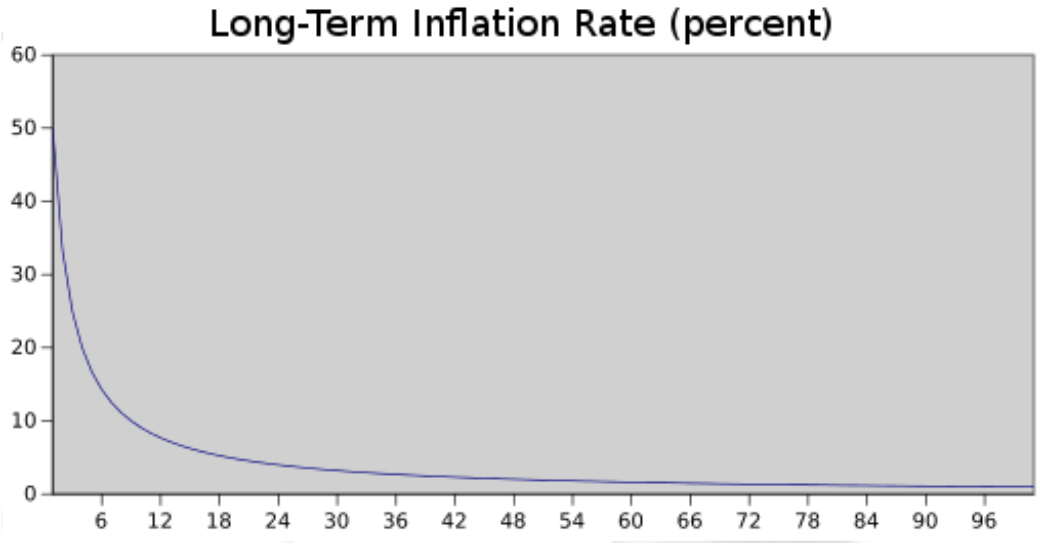
- Ether will be released in a currency sale at the price of 1000-2000 ether per BTC, a mechanism intended to fund the Ethereum organization and pay for development that has been used with success by other platforms such as Mastercoin and NXT. Earlier buyers will

benefit from larger discounts. The BTC received from the sale will be used entirely to pay salaries and bounties to developers and invested into various for-profit and non-profit projects in the Ethereum and cryptocurrency ecosystem.

- 0.099x the total amount sold (60102216 ETH) will be allocated to the organization to compensate early contributors and pay ETH-denominated expenses before the genesis block.
- 0.099x the total amount sold will be maintained as a long-term reserve.
- 0.26x the total amount sold will be allocated to miners per year forever after that point.

Group	At launch	After 1 year	After 5 years
Currency units	1.198X	1.458X	2.498X
Purchasers	83.5%	68.6%	40.0%
Reserve spent pre-sale	8.26%	6.79%	3.96%
Reserve used post-sale	8.26%	6.79%	3.96%
Miners	0%	17.8%	52.0%

Long-Term Supply Growth Rate (percent)



Despite the linear currency issuance, just like with Bitcoin over time the supply growth rate nevertheless tends to zero

The two main choices in the above model are (1) the existence and size of an endowment pool, and (2) the existence of a permanently growing linear supply, as opposed to a capped supply as in Bitcoin. The justification of the endowment pool is as follows. If the endowment pool did not exist, and the linear issuance reduced to 0.217x to provide the same inflation rate, then the total quantity of ether would be 16.5% less and so each unit would be 19.8% more valuable. Hence, in the equilibrium 19.8% more ether would be purchased in the sale, so each unit would once again be exactly as valuable as before. The organization would also then have 1.198x as much

BTC, which can be considered to be split into two slices: the original BTC, and the additional 0.198x. Hence, this situation is *exactly equivalent* to the endowment, but with one important difference: the organization holds purely BTC, and so is not incentivized to support the value of the ether unit.

The permanent linear supply growth model reduces the risk of what some see as excessive wealth concentration in Bitcoin, and gives individuals living in present and future eras a fair chance to acquire currency units, while at the same time retaining a strong incentive to obtain and hold ether because the "supply growth rate" as a percentage still tends to zero over time. We also theorize that because coins are always lost over time due to carelessness, death, etc, and coin loss can be modeled as a percentage of the total supply per year, that the total currency supply in circulation will in fact eventually stabilize at a value equal to the annual issuance divided by the loss rate (eg. at a loss rate of 1%, once the supply reaches 26X then 0.26X will be mined and 0.26X lost every year, creating an equilibrium).

Note that in the future, it is likely that Ethereum will switch to a proof-of-stake model for security, reducing the issuance requirement to somewhere between zero and 0.05X per year. In the event that the Ethereum organization loses funding or for any other reason disappears, we leave open a "social contract": anyone has the right to create a future candidate version of Ethereum, with the only condition being that the quantity of ether must be at most equal to `60102216 * (1.198 + 0.26 * n)` where `n` is the number of years after the genesis block. Creators are free to crowd-sell or otherwise assign some or all of the difference between the PoS-driven supply expansion and the maximum allowable supply expansion to pay for development. Candidate upgrades that do not comply with the social contract may justifiably be forked into compliant versions.

Mining Centralization

The Bitcoin mining algorithm works by having miners compute SHA256 on slightly modified versions of the block header millions of times over and over again, until eventually one node comes up with a version whose hash is less than the target (currently around 2^{192}). However, this mining algorithm is vulnerable to two forms of centralization. First, the mining ecosystem has come to be dominated by ASICs (application-specific integrated circuits), computer chips designed for, and therefore thousands of times more efficient at, the specific task of Bitcoin mining. This means that Bitcoin mining is no longer a highly decentralized and egalitarian pursuit, requiring millions of dollars of capital to effectively participate in. Second, most Bitcoin miners do not actually perform block validation locally; instead, they rely on a centralized mining pool to provide the block headers. This problem is arguably worse: as of the time of this writing, the top three mining pools indirectly control roughly 50% of processing power in the Bitcoin network, although this is mitigated by the fact that miners can switch to other mining pools if a pool or coalition attempts a 51% attack.

The current intent at Ethereum is to use a mining algorithm where miners are required to fetch random data from the state, compute some randomly selected transactions from the last N

blocks in the blockchain, and return the hash of the result. This has two important benefits. First, Ethereum contracts can include any kind of computation, so an Ethereum ASIC would essentially be an ASIC for general computation - ie. a better CPU. Second, mining requires access to the entire blockchain, forcing miners to store the entire blockchain and at least be capable of verifying every transaction. This removes the need for centralized mining pools; although mining pools can still serve the legitimate role of evening out the randomness of reward distribution, this function can be served equally well by peer-to-peer pools with no central control.

This model is untested, and there may be difficulties along the way in avoiding certain clever optimizations when using contract execution as a mining algorithm. However, one notably interesting feature of this algorithm is that it allows anyone to "poison the well", by introducing a large number of contracts into the blockchain specifically designed to stymie certain ASICs. The economic incentives exist for ASIC manufacturers to use such a trick to attack each other. Thus, the solution that we are developing is ultimately an adaptive economic human solution rather than purely a technical one.

Scalability

One common concern about Ethereum is the issue of scalability. Like Bitcoin, Ethereum suffers from the flaw that every transaction needs to be processed by every node in the network. With Bitcoin, the size of the current blockchain rests at about 15 GB, growing by about 1 MB per hour. If the Bitcoin network were to process Visa's 2000 transactions per second, it would grow by 1 MB per three seconds (1 GB per hour, 8 TB per year). Ethereum is likely to suffer a similar growth pattern, worsened by the fact that there will be many applications on top of the Ethereum blockchain instead of just a currency as is the case with Bitcoin, but ameliorated by the fact that Ethereum full nodes need to store just the state instead of the entire blockchain history.

The problem with such a large blockchain size is centralization risk. If the blockchain size increases to, say, 100 TB, then the likely scenario would be that only a very small number of large businesses would run full nodes, with all regular users using light SPV nodes. In such a situation, there arises the potential concern that the full nodes could band together and all agree to cheat in some profitable fashion (eg. change the block reward, give themselves BTC). Light nodes would have no way of detecting this immediately. Of course, at least one honest full node would likely exist, and after a few hours information about the fraud would trickle out through channels like Reddit, but at that point it would be too late: it would be up to the ordinary users to organize an effort to blacklist the given blocks, a massive and likely infeasible coordination problem on a similar scale as that of pulling off a successful 51% attack. In the case of Bitcoin, this is currently a problem, but there exists a blockchain modification suggested by Peter Todd which will alleviate this issue.

In the near term, Ethereum will use two additional strategies to cope with this problem. First, because of the blockchain-based mining algorithms, at least every miner will be forced to be a full node, creating a lower bound on the number of full nodes. Second and more importantly,

however, we will include an intermediate state tree root in the blockchain after processing each transaction. Even if block validation is centralized, as long as one honest verifying node exists, the centralization problem can be circumvented via a verification protocol. If a miner publishes an invalid block, that block must either be badly formatted, or the state `S[n]` is incorrect. Since `S[0]` is known to be correct, there must be some first state `S[i]` that is incorrect where `S[i-1]` is correct. The verifying node would provide the index `i`, along with a "proof of invalidity" consisting of the subset of Patricia tree nodes needing to process `APPLY(S[i-1],TX[i]) -> S[i]`. Nodes would be able to use those Patricia nodes to run that part of the computation, and see that the `S[i]` generated does not match the `S[i]` provided.

Another, more sophisticated, attack would involve the malicious miners publishing incomplete blocks, so the full information does not even exist to determine whether or not blocks are valid. The solution to this is a challenge-response protocol: verification nodes issue "challenges" in the form of target transaction indices, and upon receiving a node a light node treats the block as untrusted until another node, whether the miner or another verifier, provides a subset of Patricia nodes as a proof of validity.

Conclusion

The Ethereum protocol was originally conceived as an upgraded version of a cryptocurrency, providing advanced features such as on-blockchain escrow, withdrawal limits, financial contracts, gambling markets and the like via a highly generalized programming language. The Ethereum protocol would not "support" any of the applications directly, but the existence of a Turing-complete programming language means that arbitrary contracts can theoretically be created for any transaction type or application. What is more interesting about Ethereum, however, is that the Ethereum protocol moves far beyond just currency. Protocols around decentralized file storage, decentralized computation and decentralized prediction markets, among dozens of other such concepts, have the potential to substantially increase the efficiency of the computational industry, and provide a massive boost to other peer-to-peer protocols by adding for the first time an economic layer. Finally, there is also a substantial array of applications that have nothing to do with money at all.

The concept of an arbitrary state transition function as implemented by the Ethereum protocol provides for a platform with unique potential; rather than being a closed-ended, single-purpose protocol intended for a specific array of applications in data storage, gambling or finance, Ethereum is open-ended by design, and we believe that it is extremely well-suited to serving as a foundational layer for a very large number of both financial and non-financial protocols in the years to come.

Notes and Further Reading

Notes

1. A sophisticated reader may notice that in fact a Bitcoin address is the hash of the elliptic curve public key, and not the public key itself. However, it is in fact perfectly legitimate cryptographic terminology to refer to the pubkey hash as a public key itself. This is because Bitcoin's cryptography can be considered to be a custom digital signature algorithm, where the public key consists of the hash of the ECC pubkey, the signature consists of the ECC pubkey concatenated with the ECC signature, and the verification algorithm involves checking the ECC pubkey in the signature against the ECC pubkey hash provided as a public key and then verifying the ECC signature against the ECC pubkey.
2. Technically, the median of the 11 previous blocks.
3. The Ethereum protocol should be as simple as practical, but it may be necessary to have quite a high level of complexity, for instance to scale, to internalize costs of storage, bandwidth and I/O, for security, privacy, transparency, etc. Where complexity is necessary, documentation should be as clear, concise and up-to-date as possible, so that someone completely unschooled in Ethereum can learn it and become an expert.
4. See the Yellow Paper for the Ethereum Virtual Machine (which is useful as a specification and as a reference for building an Ethereum client from scratch), while also there are many topics in the Ethereum wiki, such as sharding developmment, core development, dapp development, research, Casper R&D, and networking protocols. For research and possible future implementation there is ethresear.ch.
5. Another way of expressing this is abstraction. The latest roadmap is planning to abstract execution, allowing execution engines to not necessarily have to follow one canonical specification, but for instance it could be tailored for a specific application, as well as a shard. (This heterogeneity of execution engines is not explicitly stated in the roadmap. There is also heterogeneous sharding, which Vlad Zamfir conceptualized.)
6. Internally, 2 and "CHARLIE" are both numbers, with the latter being in big-endian base 256 representation. Numbers can be at least 0 and at most 2^{256}-1.

Further Reading

1. Intrinsic value: http://bitcoinmagazine.com/8640/an-exploration-of-intrinsic-value-what-it-is-why-bitcoin-doesnt-have-it-and-why-bitcoin-does-have-it/
2. Smart property: https://en.bitcoin.it/wiki/Smart_Property
3. Smart contracts: https://en.bitcoin.it/wiki/Contracts
4. B-money: http://www.weidai.com/bmoney.txt
5. Reusable proofs of work: http://www.finney.org/~hal/rpow/
6. Secure property titles with owner authority: http://szabo.best.vwh.net/securetitle.html
7. Bitcoin whitepaper: http://bitcoin.org/bitcoin.pdf
8. Namecoin: https://namecoin.org/

9. Zooko's triangle: http://en.wikipedia.org/wiki/Zooko's_triangle
10. Colored coins whitepaper: https://docs.google.com/a/buterin.com/document/d/1AnkP_cVZTCMLIzw4DvsW6M8Q2JC0lIzrTLuoWu2z1BE/edit
11. Mastercoin whitepaper: https://github.com/mastercoin-MSC/spec
12. Decentralized autonomous corporations, Bitcoin Magazine: http://bitcoinmagazine.com/7050/bootstrapping-a-decentralized-autonomous-corporation-part-i/
13. Simplified payment verification: https://en.bitcoin.it/wiki/Scalability#Simplifiedpaymentverification
14. Merkle trees: http://en.wikipedia.org/wiki/Merkle_tree
15. Patricia trees: http://en.wikipedia.org/wiki/Patricia_tree
16. GHOST: https://eprint.iacr.org/2013/881.pdf
17. StorJ and Autonomous Agents, Jeff Garzik: http://garzikrants.blogspot.ca/2013/01/storj-and-bitcoin-autonomous-agents.html
18. Mike Hearn on Smart Property at Turing Festival: http://www.youtube.com/watch?v=Pu4PAMFPo5Y
19. Ethereum RLP: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-RLP
20. Ethereum Merkle Patricia trees: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-Patricia-Tree
21. Peter Todd on Merkle sum trees: http://sourceforge.net/p/bitcoin/mailman/message/31709140/

For history of the white paper, seehttps://github.com/ethereum/wiki/blob/old-before-deleting-all-files-go-to-wiki-wiki-instead/old-whitepaper-for-historical-reference.md#historical-sources-of-the-white-paper

| Deutsch | English | Español | Français | □□□ | Italiano | Portuguese | Română | فارسی | العربية | □□(□□)

| □□(□□) | □□□

▸ **Pages** 218

Contents

- Networking
- Ethereum Technologies
- Ethash/Dashimoto
- Whisper

Basics

- Home
- Ethereum Whitepaper
- Ethereum Introduction
- Uses: DAOs and dapps
- Getting Ether
- Releases
- FAQs
- Design Rationale
- EVM intro: Ethereum Yellow Paper, Beige Paper and Py-EVM.
- Wiki for (old) website (still a good introduction)
- Glossary

R&D

- Sharding introduction & R&D Compendium, FAQs & roadmap
- Casper Proof-of-Stake compendium and FAQs.
- Alternative blockchains, randomness, economics, and other research topics
- Hard Problems of Cryptocurrency
- Governance

Ethereum Virtual Machine (EVM)

Ethereum clients, tools, wallets, dapp browsers and other projects

ĐApp Development

Infrastructure

- Chain Spec Format
- Inter-exchange Client Address Protocol
- URL Hint Protocol
- NatSpec Determination
- Network Status
- Raspberry Pi
- Mining
- Licensing

- Consortium Chain Development

Networking

- libp2p (preferred over devp2p for future use, if not yet as of Feb 2019)
- Ethereum Wire Protocol
- devp2p Specifications
- devp2p Whitepaper (old)

Ethereum Technologies

- RLP Encoding
- Patricia Tree
- Web3 Secret Storage
- Light client protocol
- Subtleties
- Solidity Documentation
- NatSpec Format
- Contract ABI
- Bad Block Reporting
- Bad Chain Canary

Ethash/Dashimoto

- Ethash
- Ethash Yellow Paper appendix
- Ethash C API
- Ethash DAG

Whisper

- Whisper Proposal
- Whisper Overview
- PoC-1 Wire protocol
- PoC-2 Wire protocol
- PoC-2 Whitepaper

Clone this wiki locally

```
https://github.com/ethereum/wiki.wiki.git
```

© 2019 GitHub, Inc.
Terms
Privacy

Security
Status
Help

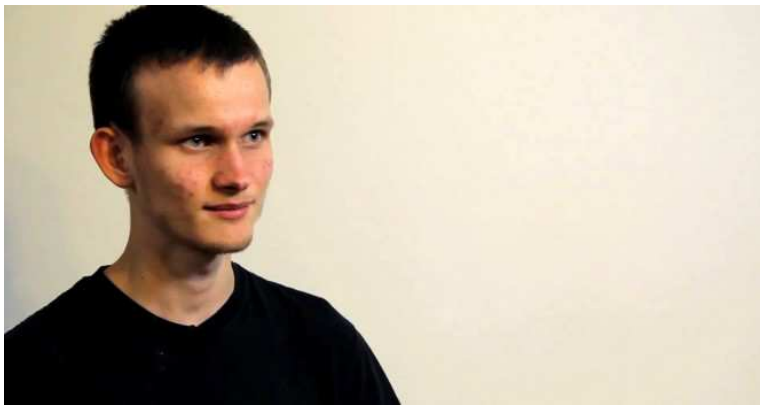
Contact GitHub
Pricing
API
Training
Blog
About

INTERNET ARCHIVE WayBackMachine http://www.wired.com/2014/01/ethereum/ Go FEB MAR APR 18 2015 2016 2017
169 captures 04 Apr 2014 - 20 Feb 2019 About this capture
BUSINESS CULTURE DESIGN GEAR SCIENCE SECURITY TRANSPORTATION

SHARE
SHARE
TWEET
PIN
COMMENT
EMAIL

KLINT FINLEY BUSINESS 01.27.14 6:30 AM

OUT IN THE OPEN: TEENAGE HACKER TRANSFORMS WEB INTO ONE GIANT BITCOIN NETWORK



Vitalik Buterin. Image: Vitalik Buterin

MOST PEOPLE THINK of bitcoin as a form of money, if they think of bitcoin at all. But 19-year-old hacker Vitalik Buterin sees it as something more — much more. He sees it as a new way of building just about any internet application.

The bitcoin digital currency is driven by open source software that runs across thousands of machines around the globe. Borrowing code from this rather clever piece of software, independent hackers have already built applications such as the Twitter-style social network Twister, the encrypted e-mail alternative Bitmessage, and the unseizable domain name system Namecoin. But Buterin believes that many other applications can benefit from the genius of the bitcoin software, and that's why he's joining

LATEST NEWS

MOVIES
New *Warcraft* Teaser: Prepare for War!
42 MINS

ABSURD CREATURES
Absurd Creatures: This Bird Impales Its Victims on Thorns. How Metal Is
3 HOURS

forces with several other hackers to create something called Ethereum.

Buterin believes so many other applications can benefit from the genius of the bitcoin software, and that's why he has joined forces with several other hackers to create something called Ethereum.

He envisions Ethereum as an online service that lets you build practically anything in the image of bitcoin and run it across a worldwide network of machines. At its core, bitcoin is a way of reliably storing and moving digital objects or pieces of information. Today, it stores and moves money, but Buterin believes the same basic system could give rise to a new breed of social networks, data storage systems and securities markets — all operated without the help of a central authority.

Born in Russia and raised in Canada, Buterin was interested in mathematics and computer science from an early age. But when he first stumbled on to bitcoin in 2011, it didn't grab him. "I ignored it," he says. "I thought it had no intrinsic value, so it had to fail."

But, over the next few weeks, he grew curious about this unusual creation. He received his first bitcoins as payment for articles written for a site called Bitcoin Weekly, where he was paid five bitcoins per article, the equivalent of $3.75 at the time. "It was my first ever real job, and it paid around $1.30 per hour," he says. He kept writing about the digital currency in the pages of Bitcoin Magazine and other pubs. Then, in 2013, just as he was about to lose interest in the thing, the price of bitcoin skyrocketed.

Deciding that bitcoin was going to be a much bigger deal than most people realized, he dropped out of university and started traveling the world, jumping from bitcoin meetup to bitcoin meetup and contributing to various open source projects. Ethereum is the result of all those conversations and software experiments.

A BITCOIN FOR EVERYTHING

Ethereum won't use the peer-to-peer network that bitcoin runs on, nor will it use the same software. Instead, Buterin and his team are building a completely new system that will

run atop its own network. But the project borrows heavily from the ideas behind the bitcoin software.

All bitcoin transactions, for instance, are stored in a massive public ledger called the "blockchain." This is a type of encrypted database, and you can use it to power other applications — as we've seen with Twister and BitMessage. Ethereum will feed still more applications through something similar to the blockchain, and it will offer a stripped-down version of the Python programming language — known as Ethereum Script — that's specifically designed for building these blockchain-based applications.

As with bitcoin, the network that underpins Ethereum will be powered by machines donated by the people of the world, and to encourage donations, the system will allow these machines to collect fees from developers who build and run an applications atop the network. In similar fashion, bitcoin shares its money with those who run the machines driving its network.

The system could potentially drive everything from Dropbox-style storage systems to custom digital currencies. According to Buterin, it will be particularly well suited to something called "smart contracts." A simple example is a betting system. Two people could place bets on, say, the outcome of the Super Bowl, entrusting a certain amount of digital currency to system. The system would then check the final score of the game via the web and distribute the funds appropriately. No bookie needed.

But Buterin also envisions far more complex smart contracts, including joint savings accounts, financial exchange markets, or even trust funds. Theoretically, these contracts would be more trustworthy because — if the software is properly designed — no one could cheat. Many bitcoin geeks even believe that smart contracts could lead to the creation "autonomous corporations" — entire companies run by bots instead of humans.

The Ethereum team plans to have a test network up and running soon, and on February 1, it will launch a crowdfunding campaign to finance its further development. The code will be open source, another reflection of the bitcoin way.

A WEB OF OUR OWN

Ethereum isn't alone in its lofty ambitions. There are several projects trying to add smart contracts and other new tools to bitcoin. Some, like QixCoin and Bitcloud, are building their own networks. Others, like Colored Coins and Mastercoin, are piggybacking on the existing bitcoin network. Buterin has contributed to both Colored Coins and Mastercoin, but ultimately, he decided that it would make more sense to create an entirely new system.

"I saw really smart people whacking their heads against the wall at Colored Coins, and eventually, I realized people are having such a hard time not because the problem is hard," he says. "The problem is easy. People are having a hard time because bitcoin is a bad protocol to build this stuff onto."

'The problem is easy. People are having a hard time because bitcoin is a bad protocol to build this stuff onto'

Although applications that run on the bitcoin network have the advantage of using existing infrastructure — and they benefit from the scrutiny that security experts give the system — they're limited by the design of the host software. For example, although bitcoin offers its own scripting language, the language is deliberately limited to ensure security. The Ethereum team believe these limitations made sense in the early days of bitcoin, when the ideas surrounding the currency were new and unproven. But they say that since bitcoin now seems reasonably stable and secure, it's time to experiment with ways of making it more flexible.

But, certainly, Ethereum faces several challenges. Many are worried that the Ethereum blockchain will quickly grow to an unwieldy size if it gains widespread use. Buterin thinks the team can tackle this problem, but we won't know for sure until the network is in action. Security is another big concern. These are just two reasons why the team is launching a test network before rolling out the real thing. "It won't be the official network and transactions won't carry over, and it will likely be quite buggy and fail a lot at first," Buterin says.

In other words, it's very early days for this new-age software system. But Ethereum and other next-gen crypto-platforms paint a very attractive picture of our online

future, one where users are in control, not governments or big companies. Building this future is an enormous task, but Vitalik Buterin wouldn't have it any other way.

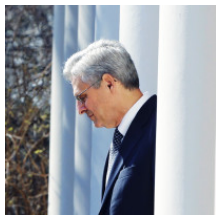
#BITCOIN #INDIE WEB #OUT IN THE OPEN

VIEW COMMENTS

SPONSORED STORIES POWERED BY OUTBRAIN

ENTERTAINMENT
Hollywood Is Not OK With You Watching New Movies at Home
6 HOURS

WE RECOMMEND
POWERED BY OUTBRAIN



POLITICS
Is Judge Garland a Liberal? Wikipedia Editors Battle to Save Facts From Politics
21 HOURS

GAME OF CLOUDS
The Epic Story of Dropbox's Exodus From the Amazon Cloud Empire
03.14.16

ARTIFICIAL

MONEY

INTELLIGENCE

In Two Moves, AlphaGo and Lee Sedol Redefined the Future

2 DAYS

MONEY

CROWDFUNDING FOR THE PUBLIC GOOD IS EVIL

2 DAYS

GET OUR NEWSLETTER

WIRED's biggest stories, delivered to your inbox.

Enter your email

SUBMIT

WE'RE ON PINTEREST

See what's inspiring us.

FOLLOW

Use of this site constitutes acceptance of our user agreement (effective 3/21/12) and privacy policy (effective 3/21/12). Affiliate link policy. Your California privacy rights. The material on this site may not be reproduced, distributed, transmitted, cached or otherwise used, except with the prior written permission of Condé Nast.

ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

BYZANTIUM VERSION 69351d5 - 2018-12-10

DR. GAVIN WOOD
FOUNDER, ETHEREUM & PARITY
GAVIN@PARITY.IO

Abstract. The blockchain paradigm when coupled with cryptographically-secured transactions has demonstrated its utility through a number of projects, with Bitcoin being one of the most notable ones. Each such project can be seen as a simple application on a decentralised, but singleton, compute resource. We can call this paradigm a transactional singleton machine with shared-state.

Ethereum implements this paradigm in a generalised manner. Furthermore it provides a plurality of such resources, each with a distinct state and operating code but able to interact through a message-passing framework with others. We discuss its design, implementation issues, the opportunities it provides and the future hurdles we envisage.

1. Introduction

With ubiquitous internet connections in most places of the world, global information transmission has become incredibly cheap. Technology-rooted movements like Bitcoin have demonstrated through the power of the default, consensus mechanisms, and voluntary respect of the social contract, that it is possible to use the internet to make a decentralised value-transfer system that can be shared across the world and virtually free to use. This system can be said to be a very specialised version of a cryptographically secure, transaction-based state machine. Follow-up systems such as Namecoin adapted this original "currency application" of the technology into other applications albeit rather simplistic ones.

Ethereum is a project which attempts to build the generalised technology; technology on which all transaction-based state machine concepts may be built. Moreover it aims to provide to the end-developer a tightly integrated end-to-end system for building software on a hitherto unexplored compute paradigm in the mainstream: a trustful object messaging compute framework.

1.1. Driving Factors.

There are many goals of this project; one key goal is to facilitate transactions between consenting individuals who would otherwise have no means to trust one another. This may be due to geographical separation, interfacing difficulty, or perhaps the incompatibility, incompetence, unwillingness, expense, uncertainty, inconvenience, or corruption of existing legal systems. By specifying a state-change system through a rich and unambiguous language, and furthermore architecting a system such that we can reasonably expect that an agreement will be thus enforced autonomously, we can provide a means to this end.

Dealings in this proposed system would have several attributes not often found in the real world. The incorruptibility of judgement, often difficult to find, comes naturally from a disinterested algorithmic interpreter. Transparency, or being able to see exactly how a state or judgement came about through the transaction log and rules or instructional codes, never happens perfectly in human-based systems since natural language is necessarily vague, information is often lacking, and plain old prejudices are difficult to shake.

Overall, we wish to provide a system such that users can be guaranteed that no matter with which other individuals, systems or organisations they interact, they can do so with absolute confidence in the possible outcomes and how those outcomes might come about.

1.2. Previous Work.

Buterin [2013a] first proposed the kernel of this work in late November, 2013. Though now evolved in many ways, the key functionality of a blockchain with a Turing-complete language and an effectively unlimited inter-transaction storage capability remains unchanged.

Dwork and Naor [1992] provided the first work into the usage of a cryptographic proof of computational expenditure ("proof-of-work") as a means of transmitting a value signal over the Internet. The value-signal was utilised here as a spam deterrence mechanism rather than any kind of currency, but critically demonstrated the potential for a basic data channel to carry a *strong economic signal*, allowing a receiver to make a physical assertion without having to rely upon *trust*. Back [2002] later produced a system in a similar vein.

The first example of utilising the proof-of-work as a strong economic signal to secure a currency was by Vishnumurthy et al. [2003]. In this instance, the token was used to keep peer-to-peer file trading in check, providing "consumers" with the ability to make micro-payments to "suppliers" for their services. The security model afforded by the proof-of-work was augmented with digital signatures and a ledger in order to ensure that the historical record couldn't be corrupted and that malicious actors could not spoof payment or unjustly complain about service delivery. Five years later, Nakamoto [2008] introduced another such proof-of-work-secured value token, somewhat wider in scope. The fruits of this project, Bitcoin, became the first widely adopted global decentralised transaction ledger.

Other projects built on Bitcoin's success; the alt-coins introduced numerous other currencies through alteration to the protocol. Some of the best known are Litecoin and Primecoin, discussed by Sprankel [2013]. Other projects sought to take the core value content mechanism of the protocol and repurpose it; Aron [2012] discusses, for example,

the Namecoin project which aims to provide a decentralised name-resolution system.

Other projects still aim to build upon the Bitcoin network itself, leveraging the large amount of value placed in the system and the vast amount of computation that goes into the consensus mechanism. The Mastercoin project, first proposed by Willett [2013], aims to build a richer protocol involving many additional high-level features on top of the Bitcoin protocol through utilisation of a number of auxiliary parts to the core protocol. The Coloured Coins project, proposed by Rosenfeld et al. [2012], takes a similar but more simplified strategy, embellishing the rules of a transaction in order to break the fungibility of Bitcoin's base currency and allow the creation and tracking of tokens through a special "chroma-wallet"-protocol-aware piece of software.

Additional work has been done in the area with discarding the decentralisation foundation; Ripple, discussed by Boutellier and Heinzen [2014], has sought to create a "federated" system for currency exchange, effectively creating a new financial clearing system. It has demonstrated that high efficiency gains can be made if the decentralisation premise is discarded.

Early work on smart contracts has been done by Szabo [1997] and Miller [1997]. Around the 1990s it became clear that algorithmic enforcement of agreements could become a significant force in human cooperation. Though no specific system was proposed to implement such a system, it was proposed that the future of law would be heavily affected by such systems. In this light, Ethereum may be seen as a general implementation of such a *crypto-law* system.

For a list of terms used in this paper, refer to Appendix A.

2. The Blockchain Paradigm

Ethereum, taken as a whole, can be viewed as a transaction-based state machine: we begin with a genesis state and incrementally execute transactions to morph it into some final state. It is this final state which we accept as the canonical "version" of the world of Ethereum. The state can include such information as account balances, reputations, trust arrangements, data pertaining to information of the physical world; in short, anything that can currently be represented by a computer is admissible. Transactions thus represent a valid arc between two states; the 'valid' part is important—there exist far more invalid state changes than valid state changes. Invalid state changes might, e.g., be things such as reducing an account balance without an equal and opposite increase elsewhere. A valid state transition is one which comes about through a transaction. Formally:

$$\boldsymbol{\sigma}_{t+1} \equiv \Upsilon(\boldsymbol{\sigma}_t, T) \tag{1}$$

where Υ is the Ethereum state transition function. In Ethereum, Υ, together with $\boldsymbol{\sigma}$ are considerably more powerful than any existing comparable system; Υ allows components to carry out arbitrary computation, while $\boldsymbol{\sigma}$ allows components to store arbitrary state between transactions.

Transactions are collated into blocks; blocks are chained together using a cryptographic hash as a means of reference. Blocks function as a journal, recording a series of transactions together with the previous block and an identifier for the final state (though do not store the final state itself—that would be far too big). They also punctuate the transaction series with incentives for nodes to *mine*. This incentivisation takes place as a state-transition function, adding value to a nominated account.

Mining is the process of dedicating effort (working) to bolster one series of transactions (a block) over any other potential competitor block. It is achieved thanks to a cryptographically secure proof. This scheme is known as a proof-of-work and is discussed in detail in section 11.5.

Formally, we expand to:

$$\boldsymbol{\sigma}_{t+1} \equiv \Pi(\boldsymbol{\sigma}_t, B) \tag{2}$$

$$B \equiv (..., (T_0, T_1, ...), ...) \tag{3}$$

$$\Pi(\boldsymbol{\sigma}, B) \equiv \Omega(B, \Upsilon(\Upsilon(\boldsymbol{\sigma}, T_0), T_1)...) \tag{4}$$

Where Ω is the block-finalisation state transition function (a function that rewards a nominated party); B is this block, which includes a series of transactions amongst some other components; and Π is the block-level state-transition function.

This is the basis of the blockchain paradigm, a model that forms the backbone of not only Ethereum, but all decentralised consensus-based transaction systems to date.

2.1. Value.

In order to incentivise computation within the network, there needs to be an agreed method for transmitting value. To address this issue, Ethereum has an intrinsic currency, Ether, known also as ETH and sometimes referred to by the Old English Đ. The smallest subdenomination of Ether, and thus the one in which all integer values of the currency are counted, is the Wei. One Ether is defined as being 10^{18} Wei. There exist other subdenominations of Ether:

Multiplier	Name
10^0	Wei
10^{12}	Szabo
10^{15}	Finney
10^{18}	Ether

Throughout the present work, any reference to value, in the context of Ether, currency, a balance or a payment, should be assumed to be counted in Wei.

2.2. Which History?

Since the system is decentralised and all parties have an opportunity to create a new block on some older pre-existing block, the resultant structure is necessarily a tree of blocks. In order to form a consensus as to which path, from root (the genesis block) to leaf (the block containing the most recent transactions) through this tree structure, known as the blockchain, there must be an agreed-upon scheme. If there is ever a disagreement between nodes as to which root-to-leaf path down the block tree is the 'best' blockchain, then a *fork* occurs.

This would mean that past a given point in time (block), multiple states of the system may coexist: some nodes believing one block to contain the canonical transactions, other nodes believing some other block to be canonical, potentially containing radically different or incompatible transactions. This is to be avoided at all costs as the uncertainty that would ensue would likely kill all confidence in the entire system.

The scheme we use in order to generate consensus is a simplified version of the GHOST protocol introduced by Sompolinsky and Zohar [2013]. This process is described in detail in section 10.

Sometimes, a path follows a new protocol from a particular height. This document describes one version of the protocol. In order to follow back the history of a path, one must reference multiple versions of this document.

3. Conventions

We use a number of typographical conventions for the formal notation, some of which are quite particular to the present work:

The two sets of highly structured, 'top-level', state values, are denoted with bold lowercase Greek letters. They fall into those of world-state, which are denoted $\boldsymbol{\sigma}$ (or a variant thereupon) and those of machine-state, $\boldsymbol{\mu}$.

Functions operating on highly structured values are denoted with an upper-case Greek letter, e.g. Υ, the Ethereum state transition function.

For most functions, an uppercase letter is used, e.g. C, the general cost function. These may be subscripted to denote specialised variants, e.g. C_{SSTORE}, the cost function for the SSTORE operation. For specialised and possibly externally defined functions, we may format as typewriter text, e.g. the Keccak-256 hash function (as per the winning entry to the SHA-3 contest by Bertoni et al. [2017], rather than later releases), is denoted KEC (and generally referred to as plain Keccak). Also KEC512 is referring to the Keccak 512 hash function.

Tuples are typically denoted with an upper-case letter, e.g. T, is used to denote an Ethereum transaction. This symbol may, if accordingly defined, be subscripted to refer to an individual component, e.g. T_{n}, denotes the nonce of said transaction. The form of the subscript is used to denote its type; e.g. uppercase subscripts refer to tuples with subscriptable components.

Scalars and fixed-size byte sequences (or, synonymously, arrays) are denoted with a normal lower-case letter, e.g. n is used in the document to denote a transaction nonce. Those with a particularly special meaning may be Greek, e.g. δ, the number of items required on the stack for a given operation.

Arbitrary-length sequences are typically denoted as a bold lower-case letter, e.g. $\mathbf{o}$ is used to denote the byte sequence given as the output data of a message call. For particularly important values, a bold uppercase letter may be used.

Throughout, we assume scalars are non-negative integers and thus belong to the set $\mathbb{N}$. The set of all byte sequences is $\mathbb{B}$, formally defined in Appendix B. If such a set of sequences is restricted to those of a particular length, it is denoted with a subscript, thus the set of all byte sequences of length 32 is named $\mathbb{B}_{32}$ and the set of all non-negative integers smaller than 2^{256} is named $\mathbb{N}_{256}$. This is formally defined in section 4.3.

Square brackets are used to index into and reference individual components or subsequences of sequences, e.g. $\boldsymbol{\mu}_{\mathbf{s}}[0]$ denotes the first item on the machine's stack. For subsequences, ellipses are used to specify the intended range, to include elements at both limits, e.g. $\boldsymbol{\mu}_{\mathbf{m}}[0..31]$ denotes the first 32 items of the machine's memory.

In the case of the global state $\boldsymbol{\sigma}$, which is a sequence of accounts, themselves tuples, the square brackets are used to reference an individual account.

When considering variants of existing values, we follow the rule that within a given scope for definition, if we assume that the unmodified 'input' value be denoted by the placeholder $\square$ then the modified and utilisable value is denoted as $\square'$, and intermediate values would be $\square^*$, $\square^{**}$ &c. On very particular occasions, in order to maximise readability and only if unambiguous in meaning, we may use alpha-numeric subscripts to denote intermediate values, especially those of particular note.

When considering the use of existing functions, given a function f, the function f^* denotes a similar, element-wise version of the function mapping instead between sequences. It is formally defined in section 4.3.

We define a number of useful functions throughout. One of the more common is ℓ, which evaluates to the last item in the given sequence:

$$\ell(\mathbf{x}) \equiv \mathbf{x}[\|\mathbf{x}\| - 1] \tag{5}$$

4. Blocks, State and Transactions

Having introduced the basic concepts behind Ethereum, we will discuss the meaning of a transaction, a block and the state in more detail.

4.1. World State.

The world state (*state*), is a mapping between addresses (160-bit identifiers) and account states (a data structure serialised as RLP, see Appendix B). Though not stored on the blockchain, it is assumed that the implementation will maintain this mapping in a modified Merkle Patricia tree (*trie*, see Appendix D). The trie requires a simple database backend that maintains a mapping of bytearrays to bytearrays; we name this underlying database the state database. This has a number of benefits; firstly the root node of this structure is cryptographically dependent on all internal data and as such its hash can be used as a secure identity for the entire system state. Secondly, being an immutable data structure, it allows any previous state (whose root hash is known) to be recalled by simply altering the root hash accordingly. Since we store all such root hashes in the blockchain, we are able to trivially revert to old states.

The account state, $\boldsymbol{\sigma}[a]$, comprises the following four fields:

- **nonce:** A scalar value equal to the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account. For account of address a in state $\boldsymbol{\sigma}$, this would be formally denoted $\boldsymbol{\sigma}[a]_{\text{n}}$.
- **balance:** A scalar value equal to the number of Wei owned by this address. Formally denoted $\boldsymbol{\sigma}[a]_{\text{b}}$.
- **storageRoot:** A 256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account (a mapping between 256-bit integer values), encoded into the trie as a mapping from the Keccak 256-bit hash of the 256-bit integer keys to the RLP-encoded 256-bit integer values. The hash is formally denoted $\boldsymbol{\sigma}[a]_{\text{s}}$.
- **codeHash:** The hash of the EVM code of this account—this is the code that gets executed should this address receive a message call; it is immutable and thus, unlike all other fields, cannot be changed after construction. All such code fragments are contained in the state database under their corresponding hashes for later retrieval. This hash is

formally denoted $\boldsymbol{\sigma}[a]_{\mathrm{c}}$, and thus the code may be denoted as $\mathbf{b}$, given that $\mathtt{KEC}(\mathbf{b}) = \boldsymbol{\sigma}[a]_{\mathrm{c}}$.

Since we typically wish to refer not to the trie's root hash but to the underlying set of key/value pairs stored within, we define a convenient equivalence:

$$\mathtt{TRIE}\big(L_I^*(\boldsymbol{\sigma}[a]_{\mathbf{s}})\big) \equiv \boldsymbol{\sigma}[a]_{\mathrm{s}} \tag{6}$$

The collapse function for the set of key/value pairs in the trie, L_I^*, is defined as the element-wise transformation of the base function L_I, given as:

$$L_I\big((k, v)\big) \equiv \big(\mathtt{KEC}(k), \mathtt{RLP}(v)\big) \tag{7}$$

where:

$$k \in \mathbb{B}_{32} \quad \wedge \quad v \in \mathbb{N} \tag{8}$$

It shall be understood that $\boldsymbol{\sigma}[a]_{\mathbf{s}}$ is not a 'physical' member of the account and does not contribute to its later serialisation.

If the **codeHash** field is the Keccak-256 hash of the empty string, i.e. $\boldsymbol{\sigma}[a]_{\mathrm{c}} = \mathtt{KEC}\big(()\big)$, then the node represents a simple account, sometimes referred to as a "non-contract" account.

Thus we may define a world-state collapse function L_S:

$$L_S(\boldsymbol{\sigma}) \equiv \{p(a) : \boldsymbol{\sigma}[a] \neq \varnothing\} \tag{9}$$

where

$$p(a) \equiv \big(\mathtt{KEC}(a), \mathtt{RLP}\big((\boldsymbol{\sigma}[a]_{\mathrm{n}}, \boldsymbol{\sigma}[a]_{\mathrm{b}}, \boldsymbol{\sigma}[a]_{\mathrm{s}}, \boldsymbol{\sigma}[a]_{\mathrm{c}})\big)\big) \tag{10}$$

This function, L_S, is used alongside the trie function to provide a short identity (hash) of the world state. We assume:

$$\forall a : \boldsymbol{\sigma}[a] = \varnothing \ \vee \ (a \in \mathbb{B}_{20} \ \wedge \ v(\boldsymbol{\sigma}[a])) \tag{11}$$

where v is the account validity function:

$$v(x) \equiv x_{\mathrm{n}} \in \mathbb{N}_{256} \wedge x_{\mathrm{b}} \in \mathbb{N}_{256} \wedge x_{\mathrm{s}} \in \mathbb{B}_{32} \wedge x_{\mathrm{c}} \in \mathbb{B}_{32} \tag{12}$$

An account is *empty* when it has no code, zero nonce and zero balance:

$$\mathtt{EMPTY}(\boldsymbol{\sigma}, a) \quad \equiv \quad \boldsymbol{\sigma}[a]_{\mathrm{c}} = \mathtt{KEC}\big(()\big) \wedge \boldsymbol{\sigma}[a]_{\mathrm{n}} = 0 \wedge \boldsymbol{\sigma}[a]_{\mathrm{b}} = 0 \tag{13}$$

Even callable precompiled contracts can have an empty account state. This is because their account states do not usually contain the code describing its behavior.

An account is *dead* when its account state is non-existent or empty:

$$\mathtt{DEAD}(\boldsymbol{\sigma}, a) \quad \equiv \quad \boldsymbol{\sigma}[a] = \varnothing \vee \mathtt{EMPTY}(\boldsymbol{\sigma}, a) \tag{14}$$

4.2. **The Transaction.** A transaction (formally, T) is a single cryptographically-signed instruction constructed by an actor externally to the scope of Ethereum. While it is assumed that the ultimate external actor will be human in nature, software tools will be used in its construction and dissemination[1]. There are two types of transactions: those which result in message calls and those which result in the creation of new accounts with associated code (known informally as 'contract creation'). Both types specify a number of common fields:

nonce: A scalar value equal to the number of transactions sent by the sender; formally T_{n}.

gasPrice: A scalar value equal to the number of Wei to be paid per unit of *gas* for all computation costs incurred as a result of the execution of this transaction; formally T_{p}.

gasLimit: A scalar value equal to the maximum amount of gas that should be used in executing this transaction. This is paid up-front, before any computation is done and may not be increased later; formally T_{g}.

to: The 160-bit address of the message call's recipient or, for a contract creation transaction, $\varnothing$, used here to denote the only member of $\mathbb{B}_0$; formally T_{t}.

value: A scalar value equal to the number of Wei to be transferred to the message call's recipient or, in the case of contract creation, as an endowment to the newly created account; formally T_{v}.

v, r, s: Values corresponding to the signature of the transaction and used to determine the sender of the transaction; formally T_{w}, T_{r} and T_{s}. This is expanded in Appendix F.

Additionally, a contract creation transaction contains:

init: An unlimited size byte array specifying the EVM-code for the account initialisation procedure, formally $T_{\mathbf{i}}$.

init is an EVM-code fragment; it returns the **body**, a second fragment of code that executes each time the account receives a message call (either through a transaction or due to the internal execution of code). **init** is executed only once at account creation and gets discarded immediately thereafter.

In contrast, a message call transaction contains:

data: An unlimited size byte array specifying the input data of the message call, formally $T_{\mathbf{d}}$.

Appendix F specifies the function, S, which maps transactions to the sender, and happens through the ECDSA of the SECP-256k1 curve, using the hash of the transaction (excepting the latter three signature fields) as the datum to sign. For the present we simply assert that the sender of a given transaction T can be represented with $S(T)$.

$$L_T(T) \equiv \begin{cases} (T_{\mathrm{n}}, T_{\mathrm{p}}, T_{\mathrm{g}}, T_{\mathrm{t}}, T_{\mathrm{v}}, T_{\mathbf{i}}, T_{\mathrm{w}}, T_{\mathrm{r}}, T_{\mathrm{s}}) & \text{if } T_{\mathrm{t}} = \varnothing \\ (T_{\mathrm{n}}, T_{\mathrm{p}}, T_{\mathrm{g}}, T_{\mathrm{t}}, T_{\mathrm{v}}, T_{\mathbf{d}}, T_{\mathrm{w}}, T_{\mathrm{r}}, T_{\mathrm{s}}) & \text{otherwise} \end{cases} \tag{15}$$

Here, we assume all components are interpreted by the RLP as integer values, with the exception of the arbitrary length byte arrays $T_{\mathbf{i}}$ and $T_{\mathbf{d}}$.

$$\begin{aligned} &T_{\mathrm{n}} \in \mathbb{N}_{256} \ \wedge \ T_{\mathrm{v}} \in \mathbb{N}_{256} \ \wedge \ T_{\mathrm{p}} \in \mathbb{N}_{256} \ \wedge \\ &T_{\mathrm{g}} \in \mathbb{N}_{256} \ \wedge \ T_{\mathrm{w}} \in \mathbb{N}_{5} \ \wedge \ T_{\mathrm{r}} \in \mathbb{N}_{256} \ \wedge \\ &T_{\mathrm{s}} \in \mathbb{N}_{256} \ \wedge \ T_{\mathbf{d}} \in \mathbb{B} \ \wedge \ T_{\mathbf{i}} \in \mathbb{B} \end{aligned} \tag{16}$$

where

$$\mathbb{N}_{\mathrm{n}} = \{P : P \in \mathbb{N} \wedge P < 2^n\} \tag{17}$$

The address hash $T_{\mathbf{t}}$ is slightly different: it is either a 20-byte address hash or, in the case of being a contract-creation transaction (and thus formally equal to $\varnothing$), it is

[1]Notably, such 'tools' could ultimately become so causally removed from their human-based initiation—or humans may become so causally-neutral—that there could be a point at which they rightly be considered autonomous agents. e.g. contracts may offer bounties to humans for being sent transactions to initiate their execution.

the RLP empty byte sequence and thus the member of $\mathbb{B}_0$:

$$T_{\mathbf{t}} \in \begin{cases} \mathbb{B}_{20} & \text{if} \quad T_{\mathrm{t}} \neq \varnothing \\ \mathbb{B}_0 & \text{otherwise} \end{cases} \tag{18}$$

4.3. **The Block.** The block in Ethereum is the collection of relevant pieces of information (known as the block *header*), H, together with information corresponding to the comprised transactions, $\mathbf{T}$, and a set of other block headers $\mathbf{U}$ that are known to have a parent equal to the present block's parent's parent (such blocks are known as *ommers*[2]). The block header contains several pieces of information:

parentHash: The Keccak 256-bit hash of the parent block's header, in its entirety; formally H_{p}.

ommersHash: The Keccak 256-bit hash of the ommers list portion of this block; formally H_{o}.

beneficiary: The 160-bit address to which all fees collected from the successful mining of this block be transferred; formally H_{c}.

stateRoot: The Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied; formally H_{r}.

transactionsRoot: The Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list portion of the block; formally H_{t}.

receiptsRoot: The Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list portion of the block; formally H_{e}.

logsBloom: The Bloom filter composed from indexable information (logger address and log topics) contained in each log entry from the receipt of each transaction in the transactions list; formally H_{b}.

difficulty: A scalar value corresponding to the difficulty level of this block. This can be calculated from the previous block's difficulty level and the timestamp; formally H_{d}.

number: A scalar value equal to the number of ancestor blocks. The genesis block has a number of zero; formally H_{i}.

gasLimit: A scalar value equal to the current limit of gas expenditure per block; formally H_{l}.

gasUsed: A scalar value equal to the total gas used in transactions in this block; formally H_{g}.

timestamp: A scalar value equal to the reasonable output of Unix's time() at this block's inception; formally H_{s}.

extraData: An arbitrary byte array containing data relevant to this block. This must be 32 bytes or fewer; formally H_{x}.

mixHash: A 256-bit hash which, combined with the nonce, proves that a sufficient amount of computation has been carried out on this block; formally H_{m}.

nonce: A 64-bit value which, combined with the mixhash, proves that a sufficient amount of computation has been carried out on this block; formally H_{n}.

The other two components in the block are simply a list of ommer block headers (of the same format as above), $B_{\mathbf{U}}$ and a series of the transactions, $B_{\mathbf{T}}$. Formally, we can refer to a block B:

$$B \equiv (B_H, B_{\mathbf{T}}, B_{\mathbf{U}}) \tag{19}$$

4.3.1. *Transaction Receipt.* In order to encode information about a transaction concerning which it may be useful to form a zero-knowledge proof, or index and search, we encode a receipt of each transaction containing certain information from its execution. Each receipt, denoted $B_{\mathbf{R}}[i]$ for the ith transaction, is placed in an index-keyed trie and the root recorded in the header as H_{e}.

The transaction receipt, R, is a tuple of four items comprising: the cumulative gas used in the block containing the transaction receipt as of immediately after the transaction has happened, R_{u}, the set of logs created through execution of the transaction, $R_{\mathbf{l}}$ and the Bloom filter composed from information in those logs, R_{b} and the status code of the transaction, R_{z}:

$$R \equiv (R_{\mathrm{u}}, R_{\mathrm{b}}, R_{\mathbf{l}}, R_{\mathrm{z}}) \tag{20}$$

The function L_R trivially prepares a transaction receipt for being transformed into an RLP-serialised byte array:

$$L_R(R) \equiv (0 \in \mathbb{B}_{256}, R_{\mathrm{u}}, R_{\mathrm{b}}, R_{\mathbf{l}}) \tag{21}$$

where $0 \in \mathbb{B}_{256}$ replaces the pre-transaction state root that existed in previous versions of the protocol.

We assert that the status code R_{z} is a non-negative integer.

$$R_{\mathrm{z}} \in \mathbb{N} \tag{22}$$

We assert that R_{u}, the cumulative gas used, is a non-negative integer and that the logs Bloom, R_{b}, is a hash of size 2048 bits (256 bytes):

$$R_{\mathrm{u}} \in \mathbb{N} \quad \wedge \quad R_{\mathrm{b}} \in \mathbb{B}_{256} \tag{23}$$

The sequence $R_{\mathbf{l}}$ is a series of log entries, $(O_0, O_1, ...)$. A log entry, O, is a tuple of the logger's address, O_a, a possibly empty series of 32-byte log topics, $O_{\mathbf{t}}$ and some number of bytes of data, $O_{\mathbf{d}}$:

$$O \equiv (O_{\mathrm{a}}, (O_{\mathbf{t}0}, O_{\mathbf{t}1}, ...), O_{\mathbf{d}}) \tag{24}$$

$$O_{\mathrm{a}} \in \mathbb{B}_{20} \quad \wedge \quad \forall_{x \in O_{\mathbf{t}}} : x \in \mathbb{B}_{32} \quad \wedge \quad O_{\mathbf{d}} \in \mathbb{B} \tag{25}$$

We define the Bloom filter function, M, to reduce a log entry into a single 256-byte hash:

$$M(O) \equiv \bigvee_{x \in \{O_{\mathrm{a}}\} \cup O_{\mathbf{t}}} \left(M_{3:2048}(x) \right) \tag{26}$$

where $M_{3:2048}$ is a specialised Bloom filter that sets three bits out of 2048, given an arbitrary byte sequence. It does this through taking the low-order 11 bits of each of

[2] *ommer* is a gender-neutral term to mean "sibling of parent"; see `https://nonbinary.miraheze.org/wiki/Gender_neutral_language#Aunt.2FUncle`

[3] 11 bits = 2^{2048}, and the low-order 11 bits is the modulo 2048 of the operand, which is in this case is "each of the first three pairs of bytes in a Keccak-256 hash of the byte sequence."

the first three pairs of bytes in a Keccak-256 hash of the byte sequence.[3] Formally:

$$M_{3:2048}(\mathbf{x} : \mathbf{x} \in \mathbb{B}) \equiv \mathbf{y} : \mathbf{y} \in \mathbb{B}_{256} \quad \text{where:} \tag{27}$$

$$\mathbf{y} = (0, 0, ..., 0) \quad \text{except:} \tag{28}$$

$$\forall_{i \in \{0,2,4\}} : \mathcal{B}_{m(\mathbf{x},i)}(\mathbf{y}) = 1 \tag{29}$$

$$m(\mathbf{x}, i) \equiv \texttt{KEC}(\mathbf{x})[i, i+1] \bmod 2048 \tag{30}$$

where $\mathcal{B}$ is the bit reference function such that $\mathcal{B}_{\mathrm{j}}(\mathbf{x})$ equals the bit of index j (indexed from 0) in the byte array $\mathbf{x}$.

4.3.2. *Holistic Validity.* We can assert a block's validity if and only if it satisfies several conditions: it must be internally consistent with the ommer and transaction block hashes and the given transactions $B_{\mathbf{T}}$ (as specified in sec 11), when executed in order on the base state $\boldsymbol{\sigma}$ (derived from the final state of the parent block), result in a new state of the identity H_{r}:

$$\begin{aligned} H_{\mathrm{r}} &\equiv \texttt{TRIE}(L_S(\Pi(\boldsymbol{\sigma}, B))) \quad \wedge \\ H_{\mathrm{o}} &\equiv \texttt{KEC}(\texttt{RLP}(L^*_H(B_{\mathbf{U}}))) \quad \wedge \\ H_{\mathrm{t}} &\equiv \texttt{TRIE}(\{\forall i < \|B_{\mathbf{T}}\|, i \in \mathbb{N} : \\ &\qquad p(i, L_T(B_{\mathbf{T}}[i]))\}) \quad \wedge \\ H_{\mathrm{e}} &\equiv \texttt{TRIE}(\{\forall i < \|B_{\mathbf{R}}\|, i \in \mathbb{N} : \\ &\qquad p(i, L_R(B_{\mathbf{R}}[i]))\}) \quad \wedge \\ H_{\mathrm{b}} &\equiv \bigvee_{\mathbf{r} \in B_{\mathbf{R}}} (\mathbf{r}_{\mathrm{b}}) \end{aligned} \tag{31}$$

where $p(k, v)$ is simply the pairwise RLP transformation, in this case, the first being the index of the transaction in the block and the second being the transaction receipt:

$$p(k, v) \equiv \big(\texttt{RLP}(k), \texttt{RLP}(v)\big) \tag{32}$$

Furthermore:

$$\texttt{TRIE}(L_S(\boldsymbol{\sigma})) = P(B_H)_{H_{\mathrm{r}}} \tag{33}$$

Thus $\texttt{TRIE}(L_S(\boldsymbol{\sigma}))$ is the root node hash of the Merkle Patricia tree structure containing the key-value pairs of the state $\boldsymbol{\sigma}$ with values encoded using RLP, and $P(B_H)$ is the parent block of B, defined directly.

The values stemming from the computation of transactions, specifically the transaction receipts, $B_{\mathbf{R}}$, and that defined through the transaction's state-accumulation function, Π, are formalised later in section 11.4.

4.3.3. *Serialisation.* The function L_B and L_H are the preparation functions for a block and block header respectively. Much like the transaction receipt preparation function L_R, we assert the types and order of the structure for when the RLP transformation is required:

$$\begin{aligned} L_H(H) \equiv (& H_{\mathrm{p}}, H_{\mathrm{o}}, H_{\mathrm{c}}, H_{\mathrm{r}}, H_{\mathrm{t}}, H_{\mathrm{e}}, H_{\mathrm{b}}, H_{\mathrm{d}}, \\ & H_{\mathrm{i}}, H_{\mathrm{l}}, H_{\mathrm{g}}, H_{\mathrm{s}}, H_{\mathrm{x}}, H_{\mathrm{m}}, H_{\mathrm{n}}) \end{aligned} \tag{34}$$

$$L_B(B) \equiv \big(L_H(B_H), L^*_T(B_{\mathbf{T}}), L^*_H(B_{\mathbf{U}})\big) \tag{35}$$

With L^*_T and L^*_H being element-wise sequence transformations, thus:

$$f^*\big((x_0, x_1, ...)\big) \equiv \big(f(x_0), f(x_1), ...\big) \quad \text{for any function } f \tag{36}$$

The component types are defined thus:

$$\begin{aligned} & H_{\mathrm{p}} \in \mathbb{B}_{32} \wedge H_{\mathrm{o}} \in \mathbb{B}_{32} \wedge H_{\mathrm{c}} \in \mathbb{B}_{20} \wedge \\ & H_{\mathrm{r}} \in \mathbb{B}_{32} \wedge H_{\mathrm{t}} \in \mathbb{B}_{32} \wedge H_{\mathrm{e}} \in \mathbb{B}_{32} \wedge \\ & H_{\mathrm{b}} \in \mathbb{B}_{256} \wedge H_{\mathrm{d}} \in \mathbb{N} \wedge H_{\mathrm{i}} \in \mathbb{N} \wedge \\ & H_{\mathrm{l}} \in \mathbb{N} \wedge H_{\mathrm{g}} \in \mathbb{N} \wedge H_{\mathrm{s}} \in \mathbb{N}_{256} \wedge \\ & H_{\mathrm{x}} \in \mathbb{B} \wedge H_{\mathrm{m}} \in \mathbb{B}_{32} \wedge H_{\mathrm{n}} \in \mathbb{B}_{8} \end{aligned} \tag{37}$$

where

$$\mathbb{B}_{\mathrm{n}} = \{B : B \in \mathbb{B} \wedge \|B\| = n\} \tag{38}$$

We now have a rigorous specification for the construction of a formal block structure. The RLP function RLP (see Appendix B) provides the canonical method for transforming this structure into a sequence of bytes ready for transmission over the wire or storage locally.

4.3.4. *Block Header Validity.* We define $P(B_H)$ to be the parent block of B, formally:

$$P(H) \equiv B' : \texttt{KEC}(\texttt{RLP}(B'_H)) = H_{\mathrm{p}} \tag{39}$$

The block number is the parent's block number incremented by one:

$$H_{\mathrm{i}} \equiv P(H)_{H_{\mathrm{i}}} + 1 \tag{40}$$

The canonical difficulty of a block of header H is defined as $D(H)$:

$$D(H) \equiv \begin{cases} D_0 & \text{if} \quad H_{\mathrm{i}} = 0 \\ \max\big(D_0, P(H)_{H_{\mathrm{d}}} + x \times \varsigma_2 + \epsilon\big) & \text{otherwise} \end{cases} \tag{41}$$

where:

$$D_0 \equiv 131072 \tag{42}$$

$$x \equiv \left\lfloor \frac{P(H)_{H_{\mathrm{d}}}}{2048} \right\rfloor \tag{43}$$

$$\varsigma_2 \equiv \max\left(y - \left\lfloor \frac{H_{\mathrm{s}} - P(H)_{H_{\mathrm{s}}}}{9} \right\rfloor, -99\right) \tag{44}$$

$$y \equiv \begin{cases} 1 & \text{if} \|P(H)_{\mathbf{U}}\| = 0 \\ 2 & \text{otherwise} \end{cases}$$

$$\epsilon \equiv \left\lfloor 2^{\left\lfloor H'_{\mathrm{i}} \div 100000 \right\rfloor - 2} \right\rfloor \tag{45}$$

$$H'_{\mathrm{i}} \equiv \max(H_{\mathrm{i}} - 3000000, 0) \tag{46}$$

Note that D_0 is the difficulty of the genesis block. The *Homestead* difficulty parameter, ς_2, is used to affect a dynamic homeostasis of time between blocks, as the time between blocks varies, as discussed below, as implemented in EIP-2 by Buterin [2015]. In the Homestead release, the exponential difficulty symbol, ϵ causes the difficulty to slowly increase (every 100,000 blocks) at an exponential rate, and thus increasing the block time difference, and putting time pressure on transitioning to proof-of-stake. This effect, known as the "difficulty bomb", or "ice age", was explained in EIP-649 by Schoedon and Buterin [2017] and delayed and implemented earlier in EIP-2. ς_2 was also modified in EIP-100 with the use of x, the adjustment factor above, and the denominator 9, in order to target the mean block time including uncle blocks by Buterin [2016]. Finally, in the Byzantium release, with EIP-649, the ice age was delayed by creating a fake block number, H'_{i}, which is obtained by subtracting three million from the actual block number, which in other words reduced ϵ and the time difference between blocks, in order to allow

more time to develop proof-of-stake and preventing the network from "freezing" up.

The canonical gas limit H_l of a block of header H must fulfil the relation:

$$\begin{aligned} & H_\mathrm{l} < P(H)_{H_\mathrm{l}} + \left\lfloor \frac{P(H)_{H_\mathrm{l}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} > P(H)_{H_\mathrm{l}} - \left\lfloor \frac{P(H)_{H_\mathrm{l}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} \geqslant 5000 \end{aligned} \tag{47}$$

H_s is the timestamp (in Unix's time()) of block H and must fulfil the relation:

$$H_\mathrm{s} > P(H)_{H_\mathrm{s}} \tag{48}$$

This mechanism enforces a homeostasis in terms of the time between blocks; a smaller period between the last two blocks results in an increase in the difficulty level and thus additional computation required, lengthening the likely next period. Conversely, if the period is too large, the difficulty, and expected time to the next block, is reduced.

The nonce, H_n, must satisfy the relations:

$$n \leqslant \frac{2^{256}}{H_\mathrm{d}} \quad \wedge \quad m = H_\mathrm{m} \tag{49}$$

with $(n, m) = \mathtt{PoW}(H_{\not\mathrm{n}}, H_\mathrm{n}, \mathbf{d})$.

Where $H_{\not\mathrm{n}}$ is the new block's header H, but *without* the nonce and mix-hash components, $\mathbf{d}$ being the current DAG, a large data set needed to compute the mix-hash, and $\mathtt{PoW}$ is the proof-of-work function (see section 11.5): this evaluates to an array with the first item being the mix-hash, to prove that a correct DAG has been used, and the second item being a pseudo-random number cryptographically dependent on H and $\mathbf{d}$. Given an approximately uniform distribution in the range $[0, 2^{64})$, the expected time to find a solution is proportional to the difficulty, H_d.

This is the foundation of the security of the blockchain and is the fundamental reason why a malicious node cannot propagate newly created blocks that would otherwise overwrite ("rewrite") history. Because the nonce must satisfy this requirement, and because its satisfaction depends on the contents of the block and in turn its composed transactions, creating new, valid, blocks is difficult and, over time, requires approximately the total compute power of the trustworthy portion of the mining peers.

Thus we are able to define the block header validity function $V(H)$:

$$\begin{aligned} V(H) \equiv \; & n \leqslant \frac{2^{256}}{H_\mathrm{d}} \wedge m = H_\mathrm{m} \quad \wedge \\ & H_\mathrm{d} = D(H) \quad \wedge \\ & H_\mathrm{g} \leq H_\mathrm{l} \quad \wedge \\ & H_\mathrm{l} < P(H)_{H_\mathrm{l}} + \left\lfloor \frac{P(H)_{H_\mathrm{l}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} > P(H)_{H_\mathrm{l}} - \left\lfloor \frac{P(H)_{H_\mathrm{l}}}{1024} \right\rfloor \quad \wedge \\ & H_\mathrm{l} \geqslant 5000 \quad \wedge \\ & H_\mathrm{s} > P(H)_{H_\mathrm{s}} \quad \wedge \\ & H_\mathrm{i} = P(H)_{H_\mathrm{i}} + 1 \quad \wedge \\ & \|H_\mathrm{x}\| \leq 32 \end{aligned} \tag{50}$$

where $(n, m) = \mathtt{PoW}(H_{\not\mathrm{n}}, H_\mathrm{n}, \mathbf{d})$

Noting additionally that **extraData** must be at most 32 bytes.

5. Gas and Payment

In order to avoid issues of network abuse and to sidestep the inevitable questions stemming from Turing completeness, all programmable computation in Ethereum is subject to fees. The fee schedule is specified in units of *gas* (see Appendix G for the fees associated with various computation). Thus any given fragment of programmable computation (this includes creating contracts, making message calls, utilising and accessing account storage and executing operations on the virtual machine) has a universally agreed cost in terms of gas.

Every transaction has a specific amount of gas associated with it: **gasLimit**. This is the amount of gas which is implicitly purchased from the sender's account balance. The purchase happens at the according **gasPrice**, also specified in the transaction. The transaction is considered invalid if the account balance cannot support such a purchase. It is named **gasLimit** since any unused gas at the end of the transaction is refunded (at the same rate of purchase) to the sender's account. Gas does not exist outside of the execution of a transaction. Thus for accounts with trusted code associated, a relatively high gas limit may be set and left alone.

In general, Ether used to purchase gas that is not refunded is delivered to the *beneficiary* address, the address of an account typically under the control of the miner. Transactors are free to specify any **gasPrice** that they wish, however miners are free to ignore transactions as they choose. A higher gas price on a transaction will therefore cost the sender more in terms of Ether and deliver a greater value to the miner and thus will more likely be selected for inclusion by more miners. Miners, in general, will choose to advertise the minimum gas price for which they will execute transactions and transactors will be free to canvas these prices in determining what gas price to offer. Since there will be a (weighted) distribution of minimum acceptable gas prices, transactors will necessarily have a trade-off to make between lowering the gas price and maximising the chance that their transaction will be mined in a timely manner.

6. Transaction Execution

The execution of a transaction is the most complex part of the Ethereum protocol: it defines the state transition function Υ. It is assumed that any transactions executed first pass the initial tests of intrinsic validity. These include:

(1) The transaction is well-formed RLP, with no additional trailing bytes;
(2) the transaction signature is valid;
(3) the transaction nonce is valid (equivalent to the sender account's current nonce);
(4) the gas limit is no smaller than the intrinsic gas, g_0, used by the transaction; and
(5) the sender account balance contains at least the cost, v_0, required in up-front payment.

Formally, we consider the function Υ, with T being a transaction and $\boldsymbol{\sigma}$ the state:

$$\boldsymbol{\sigma}' = \Upsilon(\boldsymbol{\sigma}, T) \tag{51}$$

Thus $\boldsymbol{\sigma}'$ is the post-transactional state. We also define Υ^g to evaluate to the amount of gas used in the execution of a transaction, Υ^l to evaluate to the transaction's accrued

log items and Υ^{z} to evaluate to the status code resulting from the transaction. These will be formally defined later.

6.1. **Substate.** Throughout transaction execution, we accrue certain information that is acted upon immediately following the transaction. We call this *transaction substate*, and represent it as A, which is a tuple:

$$A \equiv (A_{\mathbf{s}}, A_{\mathbf{l}}, A_{\mathbf{t}}, A_{\mathrm{r}}) \tag{52}$$

The tuple contents include $A_{\mathbf{s}}$, the self-destruct set: a set of accounts that will be discarded following the transaction's completion. $A_{\mathbf{l}}$ is the log series: this is a series of archived and indexable 'checkpoints' in VM code execution that allow for contract-calls to be easily tracked by onlookers external to the Ethereum world (such as decentralised application front-ends). $A_{\mathbf{t}}$ is the set of touched accounts, of which the empty ones are deleted at the end of a transaction. Finally there is A_{r}, the refund balance, increased through using the SSTORE instruction in order to reset contract storage to zero from some non-zero value. Though not immediately refunded, it is allowed to partially offset the total execution costs.

We define the empty substate A^0 to have no self-destructs, no logs, no touched accounts and a zero refund balance:

$$A^0 \equiv (\varnothing, (), \varnothing, 0) \tag{53}$$

6.2. **Execution.** We define intrinsic gas g_0, the amount of gas this transaction requires to be paid prior to execution, as follows:

$$g_0 \equiv \sum_{i \in T_{\mathbf{i}}, T_{\mathbf{d}}} \begin{cases} G_{\mathrm{txdatazero}} & \text{if} \quad i = 0 \\ G_{\mathrm{txdatanonzero}} & \text{otherwise} \end{cases} \tag{54}$$

$$+ \begin{cases} G_{\mathrm{txcreate}} & \text{if} \quad T_{\mathrm{t}} = \varnothing \\ 0 & \text{otherwise} \end{cases} \tag{55}$$

$$+ G_{\mathrm{transaction}} \tag{56}$$

where $T_{\mathbf{i}}, T_{\mathbf{d}}$ means the series of bytes of the transaction's associated data and initialisation EVM-code, depending on whether the transaction is for contract-creation or message-call. G_{txcreate} is added if the transaction is contract-creating, but not if a result of EVM-code. G is fully defined in Appendix G.

The up-front cost v_0 is calculated as:

$$v_0 \equiv T_{\mathrm{g}} T_{\mathrm{p}} + T_{\mathrm{v}} \tag{57}$$

The validity is determined as:

$$\begin{aligned} S(T) &\neq \varnothing \quad \wedge \\ \boldsymbol{\sigma}[S(T)] &\neq \varnothing \quad \wedge \\ T_{\mathrm{n}} &= \boldsymbol{\sigma}[S(T)]_{\mathrm{n}} \quad \wedge \\ g_0 &\leqslant T_{\mathrm{g}} \quad \wedge \\ v_0 &\leqslant \boldsymbol{\sigma}[S(T)]_{\mathrm{b}} \quad \wedge \\ T_{\mathrm{g}} &\leqslant B_{H\mathrm{l}} - \ell(B_{\mathbf{R}})_{\mathrm{u}} \end{aligned} \tag{58}$$

Note the final condition; the sum of the transaction's gas limit, T_{g}, and the gas utilised in this block prior, given by $\ell(B_{\mathbf{R}})_{\mathrm{u}}$, must be no greater than the block's **gasLimit**, $B_{H\mathrm{l}}$.

The execution of a valid transaction begins with an irrevocable change made to the state: the nonce of the account of the sender, $S(T)$, is incremented by one and the balance is reduced by part of the up-front cost, $T_{\mathrm{g}} T_{\mathrm{p}}$. The gas available for the proceeding computation, g, is defined as $T_{\mathrm{g}} - g_0$. The computation, whether contract creation or a message call, results in an eventual state (which may legally be equivalent to the current state), the change to which is deterministic and never invalid: there can be no invalid transactions from this point.

We define the checkpoint state $\boldsymbol{\sigma}_0$:

$$\boldsymbol{\sigma}_0 \equiv \boldsymbol{\sigma} \quad \text{except:} \tag{59}$$

$$\boldsymbol{\sigma}_0[S(T)]_{\mathrm{b}} \equiv \boldsymbol{\sigma}[S(T)]_{\mathrm{b}} - T_{\mathrm{g}} T_{\mathrm{p}} \tag{60}$$

$$\boldsymbol{\sigma}_0[S(T)]_{\mathrm{n}} \equiv \boldsymbol{\sigma}[S(T)]_{\mathrm{n}} + 1 \tag{61}$$

Evaluating $\boldsymbol{\sigma}_P$ from $\boldsymbol{\sigma}_0$ depends on the transaction type; either contract creation or message call; we define the tuple of post-execution provisional state $\boldsymbol{\sigma}_P$, remaining gas g', substate A and status code z:

$$(\boldsymbol{\sigma}_P, g', A, z) \equiv \begin{cases} \Lambda_4(\boldsymbol{\sigma}_0, S(T), T_{\mathrm{o}}, g, \\ \quad T_{\mathrm{p}}, T_{\mathrm{v}}, T_{\mathbf{i}}, 0, \top) & \text{if} \quad T_{\mathrm{t}} = \varnothing \\ \Theta_4(\boldsymbol{\sigma}_0, S(T), T_{\mathrm{o}}, T_{\mathrm{t}}, T_{\mathrm{t}}, \\ \quad g, T_{\mathrm{p}}, T_{\mathrm{v}}, T_{\mathbf{d}}, 0, \top) & \text{otherwise} \end{cases} \tag{62}$$

where g is the amount of gas remaining after deducting the basic amount required to pay for the existence of the transaction:

$$g \equiv T_{\mathrm{g}} - g_0 \tag{63}$$

and T_{o} is the original transactor, which can differ from the sender in the case of a message call or contract creation not directly triggered by a transaction but coming from the execution of EVM-code.

Note we use Θ_4 and Λ_4 to denote the fact that only the first four components of the functions' values are taken; the final represents the message-call's output value (a byte array) and is unused in the context of transaction evaluation.

After the message call or contract creation is processed, the refund counter has to be incremented for the accounts that were self-destructed throughout its invocation.

$$A'_r \equiv A_r + \sum_{i \in A_s} R_{selfdestruct} \tag{64}$$

Then the state is finalised by determining the amount to be refunded, g^* from the remaining gas, g', plus some allowance from the refund counter, to the sender at the original rate.

$$g^* \equiv g' + \min\left\{ \left\lfloor \frac{T_{\mathrm{g}} - g'}{2} \right\rfloor, A'_{\mathrm{r}} \right\} \tag{65}$$

The total refundable amount is the legitimately remaining gas g', added to A_{r}, with the latter component being capped up to a maximum of half (rounded down) of the total amount used $T_{\mathrm{g}} - g'$. Therefore, g^* is the total gas that remains after the transaction has been executed.

The Ether for the gas is given to the miner, whose address is specified as the beneficiary of the present block B. So we define the pre-final state $\boldsymbol{\sigma}^*$ in terms of the provisional state $\boldsymbol{\sigma}_P$:

$$\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma}_P \quad \text{except} \tag{66}$$

$$\boldsymbol{\sigma}^*[S(T)]_{\mathrm{b}} \equiv \boldsymbol{\sigma}_P[S(T)]_{\mathrm{b}} + g^* T_{\mathrm{p}} \tag{67}$$

$$\boldsymbol{\sigma}^*[m]_{\mathrm{b}} \equiv \boldsymbol{\sigma}_P[m]_{\mathrm{b}} + (T_{\mathrm{g}} - g^*) T_{\mathrm{p}} \tag{68}$$

$$m \equiv B_{H_{\mathrm{c}}} \tag{69}$$

The final state, $\boldsymbol{\sigma}'$, is reached after deleting all accounts that either appear in the self-destruct list or are touched

and empty:

$$\boldsymbol{\sigma}' \equiv \boldsymbol{\sigma}^* \quad \text{except} \tag{70}$$

$$\forall i \in A_{\mathbf{s}} : \boldsymbol{\sigma}'[i] = \varnothing \tag{71}$$

$$\forall i \in A_{\mathbf{t}} : \boldsymbol{\sigma}'[i] = \varnothing \quad \text{if} \quad \texttt{DEAD}(\boldsymbol{\sigma}^*, i) \tag{72}$$

And finally, we specify Υ^{g}, the total gas used in this transaction Υ^{l}, the logs created by this transaction and Υ^{z}, the status code of this transaction:

$$\Upsilon^{\mathrm{g}}(\boldsymbol{\sigma}, T) \equiv T_{\mathrm{g}} - g^* \tag{73}$$

$$\Upsilon^{\mathrm{l}}(\boldsymbol{\sigma}, T) \equiv A_{\mathrm{l}} \tag{74}$$

$$\Upsilon^{\mathrm{z}}(\boldsymbol{\sigma}, T) \equiv z \tag{75}$$

These are used to help define the transaction receipt and are also used later for state and nonce validation.

7. Contract Creation

There are a number of intrinsic parameters used when creating an account: sender (s), original transactor (o), available gas (g), gas price (p), endowment (v) together with an arbitrary length byte array, $\mathbf{i}$, the initialisation EVM code, the present depth of the message-call/contract-creation stack (e) and finally the permission to make modifications to the state (w).

We define the creation function formally as the function Λ, which evaluates from these values, together with the state $\boldsymbol{\sigma}$ to the tuple containing the new state, remaining gas, accrued transaction substate and an error message $(\boldsymbol{\sigma}', g', A, \mathbf{o})$, as in section 6:

$$(\boldsymbol{\sigma}', g', A, z, \mathbf{o}) \equiv \Lambda(\boldsymbol{\sigma}, s, o, g, p, v, \mathbf{i}, e, w) \tag{76}$$

The address of the new account is defined as being the rightmost 160 bits of the Keccak hash of the RLP encoding of the structure containing only the sender and the account nonce. Thus we define the resultant address for the new account a:

$$a \equiv \mathcal{B}_{96..255}\Big(\texttt{KEC}\Big(\texttt{RLP}\big(\ (s, \boldsymbol{\sigma}[s]_{\mathrm{n}} - 1)\ \big)\Big)\Big) \tag{77}$$

where KEC is the Keccak 256-bit hash function, RLP is the RLP encoding function, $\mathcal{B}_{a..b}(X)$ evaluates to a binary value containing the bits of indices in the range $[a, b]$ of the binary data X, and $\boldsymbol{\sigma}[x]$ is the address state of x, or $\varnothing$ if none exists. Note we use one fewer than the sender's nonce value; we assert that we have incremented the sender account's nonce prior to this call, and so the value used is the sender's nonce at the beginning of the responsible transaction or VM operation.

The account's nonce is initially defined as one, the balance as the value passed, the storage as empty and the code hash as the Keccak 256-bit hash of the empty string; the sender's balance is also reduced by the value passed. Thus the mutated state becomes $\boldsymbol{\sigma}^*$:

$$\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma} \quad \text{except:} \tag{78}$$

$$\boldsymbol{\sigma}^*[a] = \big(1, v + v', \texttt{TRIE}(\varnothing), \texttt{KEC}(())\big) \tag{79}$$

$$\boldsymbol{\sigma}^*[s] = \begin{cases} \varnothing & \text{if } \boldsymbol{\sigma}[s] = \varnothing \ \wedge \ v = 0 \\ \mathbf{a}^* & \text{otherwise} \end{cases} \tag{80}$$

$$\mathbf{a}^* \equiv (\boldsymbol{\sigma}[s]_{\mathrm{n}}, \boldsymbol{\sigma}[s]_{\mathrm{b}} - v, \boldsymbol{\sigma}[s]_{\mathbf{s}}, \boldsymbol{\sigma}[s]_{\mathrm{c}}) \tag{81}$$

where v' is the account's pre-existing value, in the event it was previously in existence:

$$v' \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\sigma}[a] = \varnothing \\ \boldsymbol{\sigma}[a]_{\mathrm{b}} & \text{otherwise} \end{cases} \tag{82}$$

Finally, the account is initialised through the execution of the initialising EVM code $\mathbf{i}$ according to the execution model (see section 9). Code execution can effect several events that are not internal to the execution state: the account's storage can be altered, further accounts can be created and further message calls can be made. As such, the code execution function Ξ evaluates to a tuple of the resultant state $\boldsymbol{\sigma}^{**}$, available gas remaining g^{**}, the accrued substate A and the body code of the account $\mathbf{o}$.

$$(\boldsymbol{\sigma}^{**}, g^{**}, A, \mathbf{o}) \equiv \Xi(\boldsymbol{\sigma}^*, g, I, \{s, a\}) \tag{83}$$

where I contains the parameters of the execution environment, that is:

$$I_{\mathrm{a}} \equiv a \tag{84}$$

$$I_{\mathrm{o}} \equiv o \tag{85}$$

$$I_{\mathrm{p}} \equiv p \tag{86}$$

$$I_{\mathbf{d}} \equiv () \tag{87}$$

$$I_{\mathrm{s}} \equiv s \tag{88}$$

$$I_{\mathrm{v}} \equiv v \tag{89}$$

$$I_{\mathbf{b}} \equiv \mathbf{i} \tag{90}$$

$$I_{\mathrm{e}} \equiv e \tag{91}$$

$$I_{\mathrm{w}} \equiv w \tag{92}$$

$I_{\mathbf{d}}$ evaluates to the empty tuple as there is no input data to this call. I_H has no special treatment and is determined from the blockchain.

Code execution depletes gas, and gas may not go below zero, thus execution may exit before the code has come to a natural halting state. In this (and several other) exceptional cases we say an out-of-gas (OOG) exception has occurred: The evaluated state is defined as being the empty set, $\varnothing$, and the entire create operation should have no effect on the state, effectively leaving it as it was immediately prior to attempting the creation.

If the initialization code completes successfully, a final contract-creation cost is paid, the code-deposit cost, c, proportional to the size of the created contract's code:

$$c \equiv G_{codedeposit} \times |\mathbf{o}| \tag{93}$$

If there is not enough gas remaining to pay this, i.e. $g^{**} < c$, then we also declare an out-of-gas exception.

The gas remaining will be zero in any such exceptional condition, i.e. if the creation was conducted as the reception of a transaction, then this doesn't affect payment of the intrinsic cost of contract creation; it is paid regardless. However, the value of the transaction is not transferred to the aborted contract's address when we are out-of-gas.

If such an exception does not occur, then the remaining gas is refunded to the originator and the now-altered state is allowed to persist. Thus formally, we may specify the resultant state, gas, substate and status code as $(\boldsymbol{\sigma}', g', A, z)$ where:

(94)

$$g' \equiv \begin{cases} 0 & \text{if} \quad F \\ g^{**} - c & \text{otherwise} \end{cases}$$

(95)

$$\boldsymbol{\sigma}' \equiv \begin{cases} \boldsymbol{\sigma} & \text{if} \quad F \\ \boldsymbol{\sigma}^{**} \quad \text{except:} & \\ \quad \boldsymbol{\sigma}'[a] = \varnothing & \text{if} \quad \texttt{DEAD}(\boldsymbol{\sigma}^{**}, a) \\ \boldsymbol{\sigma}^{**} \quad \text{except:} & \\ \quad \boldsymbol{\sigma}'[a]_{\mathrm{c}} = \texttt{KEC}(\mathbf{o}) & \text{otherwise} \end{cases}$$

(96)

$$z \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\sigma}^{**} = \varnothing \vee g^{**} < c \\ 1 & \text{otherwise} \end{cases}$$

where

(97)

$$F \equiv \left((\boldsymbol{\sigma}^{**} = \varnothing \ \wedge \ \mathbf{o} = \varnothing) \vee \ g^{**} < c \ \vee \ |\mathbf{o}| > 24576\right)$$

The exception in the determination of $\boldsymbol{\sigma}'$ dictates that $\mathbf{o}$, the resultant byte sequence from the execution of the initialisation code, specifies the final body code for the newly-created account.

Note that intention is that the result is either a successfully created new contract with its endowment, or no new contract with no transfer of value.

7.1. **Subtleties.** Note that while the initialisation code is executing, the newly created address exists but with no intrinsic body code[4]. Thus any message call received by it during this time causes no code to be executed. If the initialisation execution ends with a SELFDESTRUCT instruction, the matter is moot since the account will be deleted before the transaction is completed. For a normal STOP code, or if the code returned is otherwise empty, then the state is left with a zombie account, and any remaining balance will be locked into the account forever.

8. Message Call

In the case of executing a message call, several parameters are required: sender (s), transaction originator (o), recipient (r), the account whose code is to be executed (c, usually the same as recipient), available gas (g), value (v) and gas price (p) together with an arbitrary length byte array, $\mathbf{d}$, the input data of the call, the present depth of the message-call/contract-creation stack (e) and finally the permission to make modifications to the state (w).

Aside from evaluating to a new state and transaction substate, message calls also have an extra component—the output data denoted by the byte array $\mathbf{o}$. This is ignored when executing transactions, however message calls can be initiated due to VM-code execution and in this case this information is used.

(98) $$(\boldsymbol{\sigma}', g', A, z, \mathbf{o}) \equiv \Theta(\boldsymbol{\sigma}, s, o, r, c, g, p, v, \tilde{v}, \mathbf{d}, e, w)$$

Note that we need to differentiate between the value that is to be transferred, v, from the value apparent in the execution context, $\tilde{v}$, for the DELEGATECALL instruction.

We define $\boldsymbol{\sigma}_1$, the first transitional state as the original state but with the value transferred from sender to recipient:

(99) $$\boldsymbol{\sigma}_1[r]_{\mathrm{b}} \equiv \boldsymbol{\sigma}[r]_{\mathrm{b}} + v \quad \wedge \quad \boldsymbol{\sigma}_1[s]_{\mathrm{b}} \equiv \boldsymbol{\sigma}[s]_{\mathrm{b}} - v$$

unless $s = r$.

Throughout the present work, it is assumed that if $\boldsymbol{\sigma}_1[r]$ was originally undefined, it will be created as an account with no code or state and zero balance and nonce. Thus the previous equation should be taken to mean:

(100) $$\boldsymbol{\sigma}_1 \equiv \boldsymbol{\sigma}'_1 \quad \text{except:}$$

(101) $$\boldsymbol{\sigma}_1[s] \equiv \begin{cases} \varnothing & \text{if } \boldsymbol{\sigma}'_1[s] = \varnothing \ \wedge \ v = 0 \\ \mathbf{a}_1 & \text{otherwise} \end{cases}$$

(102) $$\mathbf{a}_1 \equiv \left(\boldsymbol{\sigma}'_1[s]_{\mathrm{n}}, \boldsymbol{\sigma}'_1[s]_{\mathrm{b}} - v, \boldsymbol{\sigma}'_1[s]_{\mathbf{s}}, \boldsymbol{\sigma}'_1[s]_{\mathrm{c}}\right)$$

(103) $$\text{and} \quad \boldsymbol{\sigma}'_1 \equiv \boldsymbol{\sigma} \quad \text{except:}$$

(104)

$$\begin{cases} \boldsymbol{\sigma}'_1[r] \equiv (0, v, \texttt{TRIE}(\varnothing), \texttt{KEC}(())) & \text{if} \quad \boldsymbol{\sigma}[r] = \varnothing \wedge v \neq 0 \\ \boldsymbol{\sigma}'_1[r] \equiv \varnothing & \text{if} \quad \boldsymbol{\sigma}[r] = \varnothing \wedge v = 0 \\ \boldsymbol{\sigma}'_1[r] \equiv \mathbf{a}'_1 & \text{otherwise} \end{cases}$$

(105) $$\mathbf{a}'_1 \equiv (\boldsymbol{\sigma}[r]_{\mathrm{n}}, \boldsymbol{\sigma}[r]_{\mathrm{b}} + v, \boldsymbol{\sigma}[r]_{\mathbf{s}}, \boldsymbol{\sigma}[r]_{\mathrm{c}})$$

The account's associated code (identified as the fragment whose Keccak hash is $\boldsymbol{\sigma}[c]_{\mathrm{c}}$) is executed according to the execution model (see section 9). Just as with contract creation, if the execution halts in an exceptional fashion (i.e. due to an exhausted gas supply, stack underflow, invalid jump destination or invalid instruction), then no gas is refunded to the caller and the state is reverted to the point immediately prior to balance transfer (i.e. $\boldsymbol{\sigma}$).

(106) $$\boldsymbol{\sigma}' \equiv \begin{cases} \boldsymbol{\sigma} & \text{if} \quad \boldsymbol{\sigma}^{**} = \varnothing \\ \boldsymbol{\sigma}^{**} & \text{otherwise} \end{cases}$$

(107) $$g' \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\sigma}^{**} = \varnothing \ \wedge \\ & \quad \mathbf{o} = \varnothing \\ g^{**} & \text{otherwise} \end{cases}$$

$$z \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\sigma}^{**} = \varnothing \\ 1 & \text{otherwise} \end{cases}$$

(108) $$(\boldsymbol{\sigma}^{**}, g^{**}, A, \mathbf{o}) \equiv \Xi$$

(109) $$I_{\mathrm{a}} \equiv r$$

(110) $$I_{\mathrm{o}} \equiv o$$

(111) $$I_{\mathrm{p}} \equiv p$$

(112) $$I_{\mathbf{d}} \equiv \mathbf{d}$$

(113) $$I_{\mathrm{s}} \equiv s$$

(114) $$I_{\mathrm{v}} \equiv \tilde{v}$$

(115) $$I_{\mathrm{e}} \equiv e$$

(116) $$I_{\mathrm{w}} \equiv w$$

(117) $$\mathbf{t} \equiv \{s, r\}$$

(118)

[4]During initialization code execution, `EXTCODESIZE` on the address should return zero, which is the length of the code of the account while `CODESIZE` should return the length of the initialization code (as defined in H.2).

where

$$\Xi \equiv \begin{cases} \Xi_{\texttt{ECREC}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 1 \\ \Xi_{\texttt{SHA256}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 2 \\ \Xi_{\texttt{RIP160}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 3 \\ \Xi_{\texttt{ID}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 4 \\ \Xi_{\texttt{EXPMOD}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 5 \\ \Xi_{\texttt{BN\_ADD}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 6 \\ \Xi_{\texttt{BN\_MUL}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 7 \\ \Xi_{\texttt{SNARKV}}(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{if} \quad r = 8 \\ \Xi(\boldsymbol{\sigma}_1, g, I, \mathbf{t}) & \text{otherwise} \end{cases} \tag{119}$$

$$\text{Let } \texttt{KEC}(I_{\mathbf{b}}) = \boldsymbol{\sigma}[c]_{\text{c}} \tag{120}$$

It is assumed that the client will have stored the pair $(\texttt{KEC}(I_{\mathbf{b}}), I_{\mathbf{b}})$ at some point prior in order to make the determination of $I_{\mathbf{b}}$ feasible.

As can be seen, there are eight exceptions to the usage of the general execution framework Ξ for evaluation of the message call: these are eight so-called 'precompiled' contracts, meant as a preliminary piece of architecture that may later become *native extensions*. The eight contracts in addresses 1 to 8 execute the elliptic curve public key recovery function, the SHA2 256-bit hash scheme, the RIPEMD 160-bit hash scheme, the identity function, arbitrary precision modular exponentiation, elliptic curve addition, elliptic curve scalar multiplication and an elliptic curve pairing check respectively.

Their full formal definition is in Appendix E.

9. Execution Model

The execution model specifies how the system state is altered given a series of bytecode instructions and a small tuple of environmental data. This is specified through a formal model of a virtual state machine, known as the Ethereum Virtual Machine (EVM). It is a *quasi*-Turing-complete machine; the *quasi* qualification comes from the fact that the computation is intrinsically bounded through a parameter, *gas*, which limits the total amount of computation done.

9.1. **Basics.** The EVM is a simple stack-based architecture. The word size of the machine (and thus size of stack items) is 256-bit. This was chosen to facilitate the Keccak-256 hash scheme and elliptic-curve computations. The memory model is a simple word-addressed byte array. The stack has a maximum size of 1024. The machine also has an independent storage model; this is similar in concept to the memory but rather than a byte array, it is a word-addressable word array. Unlike memory, which is volatile, storage is non volatile and is maintained as part of the system state. All locations in both storage and memory are well-defined initially as zero.

The machine does not follow the standard von Neumann architecture. Rather than storing program code in generally-accessible memory or storage, it is stored separately in a virtual ROM interactable only through a specialised instruction.

The machine can have exceptional execution for several reasons, including stack underflows and invalid instructions. Like the out-of-gas exception, they do not leave state changes intact. Rather, the machine halts immediately and reports the issue to the execution agent (either the transaction processor or, recursively, the spawning execution environment) which will deal with it separately.

9.2. **Fees Overview.** Fees (denominated in gas) are charged under three distinct circumstances, all three as prerequisite to the execution of an operation. The first and most common is the fee intrinsic to the computation of the operation (see Appendix G). Secondly, gas may be deducted in order to form the payment for a subordinate message call or contract creation; this forms part of the payment for CREATE, CALL and CALLCODE. Finally, gas may be paid due to an increase in the usage of the memory.

Over an account's execution, the total fee for memory-usage payable is proportional to smallest multiple of 32 bytes that are required such that all memory indices (whether for read or write) are included in the range. This is paid for on a just-in-time basis; as such, referencing an area of memory at least 32 bytes greater than any previously indexed memory will certainly result in an additional memory usage fee. Due to this fee it is highly unlikely addresses will ever go above 32-bit bounds. That said, implementations must be able to manage this eventuality.

Storage fees have a slightly nuanced behaviour—to incentivise minimisation of the use of storage (which corresponds directly to a larger state database on all nodes), the execution fee for an operation that clears an entry in the storage is not only waived, a qualified refund is given; in fact, this refund is effectively paid up-front since the initial usage of a storage location costs substantially more than normal usage.

See Appendix H for a rigorous definition of the EVM gas cost.

9.3. **Execution Environment.** In addition to the system state $\boldsymbol{\sigma}$, and the remaining gas for computation g, there are several pieces of important information used in the execution environment that the execution agent must provide; these are contained in the tuple I:

- I_{a}, the address of the account which owns the code that is executing.
- I_{o}, the sender address of the transaction that originated this execution.
- I_{p}, the price of gas in the transaction that originated this execution.
- $I_{\mathbf{d}}$, the byte array that is the input data to this execution; if the execution agent is a transaction, this would be the transaction data.
- I_{s}, the address of the account which caused the code to be executing; if the execution agent is a transaction, this would be the transaction sender.
- I_{v}, the value, in Wei, passed to this account as part of the same procedure as execution; if the execution agent is a transaction, this would be the transaction value.
- $I_{\mathbf{b}}$, the byte array that is the machine code to be executed.
- $I_{\mathbf{H}}$, the block header of the present block.
- I_{e}, the depth of the present message-call or contract-creation (i.e. the number of CALLs or CREATEs being executed at present).
- I_{w}, the permission to make modifications to the state.

The execution model defines the function Ξ, which can compute the resultant state $\boldsymbol{\sigma}'$, the remaining gas g', the

accrued substate A and the resultant output, $\mathbf{o}$, given these definitions. For the present context, we will define it as:

$$(\boldsymbol{\sigma}', g', A, \mathbf{o}) \equiv \Xi(\boldsymbol{\sigma}, g, I) \tag{121}$$

where we will remember that A, the accrued substate is defined as the tuple of the selfdestructs set $\mathbf{s}$, the log series $\mathbf{l}$, the touched accounts $\mathbf{t}$ and the refunds r:

$$A \equiv (\mathbf{s}, \mathbf{l}, \mathbf{t}, r) \tag{122}$$

9.4. **Execution Overview.** We must now define the Ξ function. In most practical implementations this will be modelled as an iterative progression of the pair comprising the full system state, $\boldsymbol{\sigma}$ and the machine state, $\boldsymbol{\mu}$. Formally, we define it recursively with a function X. This uses an iterator function O (which defines the result of a single cycle of the state machine) together with functions Z which determines if the present state is an exceptional halting state of the machine and H, specifying the output data of the instruction if and only if the present state is a normal halting state of the machine.

The empty sequence, denoted (), is not equal to the empty set, denoted $\varnothing$; this is important when interpreting the output of H, which evaluates to $\varnothing$ when execution is to continue but a series (potentially empty) when execution should halt.

$$\Xi(\boldsymbol{\sigma}, g, I, T) \equiv (\boldsymbol{\sigma}', \boldsymbol{\mu}'_{\mathrm{g}}, A, \mathbf{o}) \tag{123}$$

$$(\boldsymbol{\sigma}', \boldsymbol{\mu}', A, ..., \mathbf{o}) \equiv X\big((\boldsymbol{\sigma}, \boldsymbol{\mu}, A^0, I)\big) \tag{124}$$

$$\boldsymbol{\mu}_{\mathrm{g}} \equiv g \tag{125}$$

$$\boldsymbol{\mu}_{\mathrm{pc}} \equiv 0 \tag{126}$$

$$\boldsymbol{\mu}_{\mathbf{m}} \equiv (0, 0, ...) \tag{127}$$

$$\boldsymbol{\mu}_{\mathrm{i}} \equiv 0 \tag{128}$$

$$\boldsymbol{\mu}_{\mathbf{s}} \equiv () \tag{129}$$

$$\boldsymbol{\mu}_{\mathbf{o}} \equiv () \tag{130}$$

(131)
$$X\big((\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I)\big) \equiv \begin{cases} \big(\varnothing, \boldsymbol{\mu}, A^0, I, \varnothing\big) & \text{if } Z(\boldsymbol{\sigma}, \boldsymbol{\mu}, I) \\ \big(\varnothing, \boldsymbol{\mu}', A^0, I, \mathbf{o}\big) & \text{if } w = \text{REVERT} \\ O(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \cdot \mathbf{o} & \text{if } \mathbf{o} \neq \varnothing \\ X\big(O(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I)\big) & \text{otherwise} \end{cases}$$

where

$$\mathbf{o} \equiv H(\boldsymbol{\mu}, I) \tag{132}$$

$$(a, b, c, d) \cdot e \equiv (a, b, c, d, e) \tag{133}$$

$$\boldsymbol{\mu}' \equiv \boldsymbol{\mu} \text{ except:} \tag{134}$$

$$\boldsymbol{\mu}'_{\mathrm{g}} \equiv \boldsymbol{\mu}_{\mathrm{g}} - C(\boldsymbol{\sigma}, \boldsymbol{\mu}, I) \tag{135}$$

Note that, when we evaluate Ξ, we drop the fourth element I' and extract the remaining gas $\boldsymbol{\mu}'_{\mathrm{g}}$ from the resultant machine state $\boldsymbol{\mu}'$.

X is thus cycled (recursively here, but implementations are generally expected to use a simple iterative loop) until either Z becomes true indicating that the present state is exceptional and that the machine must be halted and any changes discarded or until H becomes a series (rather than the empty set) indicating that the machine has reached a controlled halt.

9.4.1. *Machine State.* The machine state $\boldsymbol{\mu}$ is defined as the tuple $(g, pc, \mathbf{m}, i, \mathbf{s})$ which are the gas available, the program counter $pc \in \mathbb{N}_{256}$, the memory contents, the active number of words in memory (counting continuously from position 0), and the stack contents. The memory contents $\boldsymbol{\mu}_{\mathbf{m}}$ are a series of zeroes of size 2^{256}.

For the ease of reading, the instruction mnemonics, written in small-caps (e.g. ADD), should be interpreted as their numeric equivalents; the full table of instructions and their specifics is given in Appendix H.

For the purposes of defining Z, H and O, we define w as the current operation to be executed:

$$w \equiv \begin{cases} I_{\mathbf{b}}[\boldsymbol{\mu}_{\mathrm{pc}}] & \text{if } \boldsymbol{\mu}_{\mathrm{pc}} < \|I_{\mathbf{b}}\| \\ \text{STOP} & \text{otherwise} \end{cases} \tag{136}$$

We also assume the fixed amounts of δ and α, specifying the stack items removed and added, both subscriptable on the instruction and an instruction cost function C evaluating to the full cost, in gas, of executing the given instruction.

9.4.2. *Exceptional Halting.* The exceptional halting function Z is defined as:

(137)
$$\begin{aligned} Z(\boldsymbol{\sigma}, \boldsymbol{\mu}, I) \equiv\ & \boldsymbol{\mu}_g < C(\boldsymbol{\sigma}, \boldsymbol{\mu}, I) \quad \vee \\ & \delta_w = \varnothing \quad \vee \\ & \|\boldsymbol{\mu}_{\mathbf{s}}\| < \delta_w \quad \vee \\ & (w = \text{JUMP} \quad \wedge \quad \boldsymbol{\mu}_{\mathbf{s}}[0] \notin D(I_{\mathbf{b}})) \quad \vee \\ & (w = \text{JUMPI} \quad \wedge \quad \boldsymbol{\mu}_{\mathbf{s}}[1] \neq 0 \quad \wedge \\ & \quad \boldsymbol{\mu}_{\mathbf{s}}[0] \notin D(I_{\mathbf{b}})) \quad \vee \\ & (w = \text{RETURNDATACOPY} \wedge \\ & \quad \boldsymbol{\mu}_{\mathbf{s}}[1] + \boldsymbol{\mu}_{\mathbf{s}}[2] > \|\boldsymbol{\mu}_{\mathbf{o}}\|) \quad \vee \\ & \|\boldsymbol{\mu}_{\mathbf{s}}\| - \delta_w + \alpha_w > 1024 \vee (\neg I_{\mathrm{w}} \wedge W(w, \boldsymbol{\mu})) \end{aligned}$$

where

(138)
$$\begin{aligned} W(w, \boldsymbol{\mu}) \equiv\ & w \in \{\text{CREATE}, \text{SSTORE}, \\ & \text{SELFDESTRUCT}\} \vee \\ & \text{LOG0} \leq w \wedge w \leq \text{LOG4} \quad \vee \\ & w \in \{\text{CALL}, \text{CALLCODE}\} \wedge \boldsymbol{\mu}_{\mathbf{s}}[2] \neq 0 \end{aligned}$$

This states that the execution is in an exceptional halting state if there is insufficient gas, if the instruction is invalid (and therefore its δ subscript is undefined), if there are insufficient stack items, if a JUMP/JUMPI destination is invalid, the new stack size would be larger than 1024 or state modification is attempted during a static call. The astute reader will realise that this implies that no instruction can, through its execution, cause an exceptional halt.

9.4.3. *Jump Destination Validity.* We previously used D as the function to determine the set of valid jump destinations given the code that is being run. We define this as any position in the code occupied by a JUMPDEST instruction.

All such positions must be on valid instruction boundaries, rather than sitting in the data portion of PUSH operations and must appear within the explicitly defined portion of the code (rather than in the implicitly defined STOP operations that trail it).

Formally:

$$D(\mathbf{c}) \equiv D_J(\mathbf{c}, 0) \tag{139}$$

where:

(140)
$$D_J(\mathbf{c}, i) \equiv \begin{cases} \{\} & \text{if} \quad i \geqslant |\mathbf{c}| \\ \{i\} \cup D_J(\mathbf{c}, N(i, \mathbf{c}[i])) & \\ \quad \text{if} \quad \mathbf{c}[i] = \text{JUMPDEST} & \\ D_J(\mathbf{c}, N(i, \mathbf{c}[i])) & \text{otherwise} \end{cases}$$

where N is the next valid instruction position in the code, skipping the data of a PUSH instruction, if any:

(141)
$$N(i, w) \equiv \begin{cases} i + w - \text{PUSH1} + 2 & \\ \quad \text{if} \quad w \in [\text{PUSH1}, \text{PUSH32}] & \\ i + 1 & \text{otherwise} \end{cases}$$

9.4.4. *Normal Halting.* The normal halting function H is defined:

(142)
$$H(\boldsymbol{\mu}, I) \equiv \begin{cases} H_{\text{RETURN}}(\boldsymbol{\mu}) \text{ if} & w \in \{\text{RETURN}, \text{REVERT}\} \\ () & \text{if} \quad w \in \{\text{STOP}, \text{SELFDESTRUCT}\} \\ \varnothing & \text{otherwise} \end{cases}$$

The data-returning halt operations, RETURN and REVERT, have a special function H_{RETURN}. Note also the difference between the empty sequence and the empty set as discussed here.

9.5. **The Execution Cycle.** Stack items are added or removed from the left-most, lower-indexed portion of the series; all other items remain unchanged:

(143) $$O\big((\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I)\big) \equiv (\boldsymbol{\sigma}', \boldsymbol{\mu}', A', I)$$

(144) $$\Delta \equiv \alpha_w - \delta_w$$

(145) $$\|\boldsymbol{\mu}'_{\mathbf{s}}\| \equiv \|\boldsymbol{\mu}_{\mathbf{s}}\| + \Delta$$

(146) $$\forall x \in [\alpha_w, \|\boldsymbol{\mu}'_{\mathbf{s}}\|) : \boldsymbol{\mu}'_{\mathbf{s}}[x] \equiv \boldsymbol{\mu}_{\mathbf{s}}[x - \Delta]$$

The gas is reduced by the instruction's gas cost and for most instructions, the program counter increments on each cycle, for the three exceptions, we assume a function J, subscripted by one of two instructions, which evaluates to the according value:

(147) $$\boldsymbol{\mu}'_g \equiv \boldsymbol{\mu}_g - C(\boldsymbol{\sigma}, \boldsymbol{\mu}, I)$$

(148) $$\boldsymbol{\mu}'_{\text{pc}} \equiv \begin{cases} J_{\text{JUMP}}(\boldsymbol{\mu}) & \text{if} \quad w = \text{JUMP} \\ J_{\text{JUMPI}}(\boldsymbol{\mu}) & \text{if} \quad w = \text{JUMPI} \\ N(\boldsymbol{\mu}_{\text{pc}}, w) & \text{otherwise} \end{cases}$$

In general, we assume the memory, self-destruct set and system state don't change:

(149) $$\boldsymbol{\mu}'_{\mathbf{m}} \equiv \boldsymbol{\mu}_{\mathbf{m}}$$

(150) $$\boldsymbol{\mu}'_{\text{i}} \equiv \boldsymbol{\mu}_{\text{i}}$$

(151) $$A' \equiv A$$

(152) $$\boldsymbol{\sigma}' \equiv \boldsymbol{\sigma}$$

However, instructions do typically alter one or several components of these values. Altered components listed by instruction are noted in Appendix H, alongside values for α and δ and a formal description of the gas requirements.

10. Blocktree to Blockchain

The canonical blockchain is a path from root to leaf through the entire block tree. In order to have consensus over which path it is, conceptually we identify the path that has had the most computation done upon it, or, the *heaviest* path. Clearly one factor that helps determine the heaviest path is the block number of the leaf, equivalent to the number of blocks, not counting the unmined genesis block, in the path. The longer the path, the greater the total mining effort that must have been done in order to arrive at the leaf. This is akin to existing schemes, such as that employed in Bitcoin-derived protocols.

Since a block header includes the difficulty, the header alone is enough to validate the computation done. Any block contributes toward the total computation or *total difficulty* of a chain.

Thus we define the total difficulty of block B recursively as:

(153) $$B_{\text{t}} \equiv B'_{\text{t}} + B_{\text{d}}$$

(154) $$B' \equiv P(B_H)$$

As such given a block B, B_{t} is its total difficulty, B' is its parent block and B_{d} is its difficulty.

11. Block Finalisation

The process of finalising a block involves four stages:

(1) Validate (or, if mining, determine) ommers;
(2) validate (or, if mining, determine) transactions;
(3) apply rewards;
(4) verify (or, if mining, compute a valid) state and block nonce.

11.1. **Ommer Validation.** The validation of ommer headers means nothing more than verifying that each ommer header is both a valid header and satisfies the relation of Nth-generation ommer to the present block where $N \leq 6$. The maximum of ommer headers is two. Formally:

(155) $$\|B_{\mathbf{U}}\| \leqslant 2 \bigwedge_{\mathbf{U} \in B_{\mathbf{U}}} V(\mathbf{U}) \ \wedge \ k(\mathbf{U}, P(\mathbf{B_H})_{\mathbf{H}}, 6)$$

where k denotes the "is-kin" property:

(156)
$$k(U, H, n) \equiv \begin{cases} false & \text{if} \quad n = 0 \\ s(U, H) & \\ \quad \vee \, k(U, P(H)_H, n - 1) & \text{otherwise} \end{cases}$$

and s denotes the "is-sibling" property:

(157) $$s(U, H) \equiv (P(H) = P(U) \ \wedge \ H \neq U \ \wedge \ U \notin B(H)_{\mathbf{U}})$$

where $B(H)$ and $P(H)$ are the block and the parent block of the corresponding header H respectively.

11.2. **Transaction Validation.** The given **gasUsed** must correspond faithfully to the transactions listed: B_{Hg}, the total gas used in the block, must be equal to the accumulated gas used according to the final transaction:

(158) $$B_{Hg} = \ell(\mathbf{R})_{\text{u}}$$

11.3. **Reward Application.** The application of rewards to a block involves raising the balance of the accounts of the beneficiary address of the block and each ommer by a certain amount. We raise the block's beneficiary account by R_{block}; for each ommer, we raise the block's beneficiary by an additional $\frac{1}{32}$ of the block reward and the beneficiary of the ommer gets rewarded depending on the block number. Formally we define the function Ω:

(159)
$$\Omega(B, \boldsymbol{\sigma}) \equiv \boldsymbol{\sigma}' : \boldsymbol{\sigma}' = \boldsymbol{\sigma} \quad \text{except:}$$

(160) $$\boldsymbol{\sigma}'[\mathbf{B}_{Hc}]_{\text{b}} = \boldsymbol{\sigma}[\mathbf{B}_{Hc}]_{\text{b}} + \left(1 + \frac{\|\mathbf{B}_{\mathbf{U}}\|}{32}\right) R_{\text{block}}$$

(161) $$\forall_{\mathbf{U} \in \mathbf{B}_{\mathbf{U}}} :$$
$$\boldsymbol{\sigma}'[\mathbf{U}_{\text{c}}] = \begin{cases} \varnothing & \text{if } \boldsymbol{\sigma}[\mathbf{U}_{\text{c}}] = \varnothing \ \wedge \ R = 0 \\ \mathbf{a}' & \text{otherwise} \end{cases}$$

(162) $$\mathbf{a}' \equiv (\boldsymbol{\sigma}[U_{\text{c}}]_{\text{n}}, \boldsymbol{\sigma}[U_{\text{c}}]_{\text{b}} + R, \boldsymbol{\sigma}[U_{\text{c}}]_{\mathbf{s}}, \boldsymbol{\sigma}[U_{\text{c}}]_{\text{c}})$$

(163) $$R \equiv \left(1 + \frac{1}{8}(U_{\text{i}} - B_{H\text{i}})\right) R_{\text{block}}$$

If there are collisions of the beneficiary addresses between ommers and the block (i.e. two ommers with the same beneficiary address or an ommer with the same beneficiary address as the present block), additions are applied cumulatively.

We define the block reward as 3 Ether:

(164) $$\text{Let} \quad R_{\text{block}} = 3 \times 10^{18}$$

11.4. **State & Nonce Validation.** We may now define the function, Γ, that maps a block B to its initiation state:

(165)
$$\Gamma(B) \equiv \begin{cases} \boldsymbol{\sigma}_0 kern10pc \text{ if } & P(B_H) = \varnothing \\ \boldsymbol{\sigma}_{\text{i}} : \texttt{TRIE}(L_S(\boldsymbol{\sigma}_{\text{i}})) = P(B_H)_{H_{\text{r}}} & \text{otherwise} \end{cases}$$

Here, $\texttt{TRIE}(L_S(\boldsymbol{\sigma}_{\text{i}}))$ means the hash of the root node of a trie of state $\boldsymbol{\sigma}_{\text{i}}$; it is assumed that implementations will store this in the state database, which is trivial and efficient since the trie is by nature an immutable data structure.

And finally we define Φ, the block transition function, which maps an incomplete block B to a complete block B':

(166) $$\Phi(B) \equiv B' : \quad B' = B^* \quad \text{except:}$$

(167) $$B'_{\text{n}} = n : \quad x \leqslant \frac{2^{256}}{H_{\text{d}}}$$

(168) $$B'_{\text{m}} = m \quad \text{with } (x, m) = \texttt{PoW}(B^*_{\mathbf{H}}, n, \mathbf{d})$$

(169) $$B^* \equiv B \quad \text{except:} \quad B^*_{\text{r}} = r(\Pi(\Gamma(B), B))$$

With $\mathbf{d}$ being a dataset as specified in appendix J.

As specified at the beginning of the present work, Π is the state-transition function, which is defined in terms of Ω, the block finalisation function and Υ, the transaction-evaluation function, both now well-defined.

As previously detailed, $\mathbf{R}[n]_{\text{z}}$, $\mathbf{R}[n]_{\mathbf{l}}$ and $\mathbf{R}[n]_{\text{u}}$ are the nth corresponding status code, logs and cumulative gas used after each transaction ($\mathbf{R}[n]_{\text{b}}$, the fourth component in the tuple, has already been defined in terms of the logs). We also define the nth state $\boldsymbol{\sigma}[n]$, which is defined simply as the state resulting from applying the corresponding transaction to the state resulting from the previous transaction (or the block's initial state in the case of the first such transaction):

(170) $$\boldsymbol{\sigma}[n] = \begin{cases} \Gamma(B) & \text{if} \quad n < 0 \\ \Upsilon(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n]) & \text{otherwise} \end{cases}$$

In the case of $B_{\mathbf{R}}[n]_{\text{u}}$, we take a similar approach defining each item as the gas used in evaluating the corresponding transaction summed with the previous item (or zero, if it is the first), giving us a running total:

(171) $$\mathbf{R}[n]_{\text{u}} = \begin{cases} 0 & \text{if} \quad n < 0 \\ \Upsilon^g(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n]) & \\ \quad +\mathbf{R}[n-1]_{\text{u}} & \text{otherwise} \end{cases}$$

For $\mathbf{R}[n]_{\mathbf{l}}$, we utilise the $\Upsilon^{\mathbf{l}}$ function that we conveniently defined in the transaction execution function.

(172) $$\mathbf{R}[n]_{\mathbf{l}} = \Upsilon^{\mathbf{l}}(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n])$$

We define $\mathbf{R}[n]_{\text{z}}$ in a similar manner.

(173) $$\mathbf{R}[n]_{\text{z}} = \Upsilon^{z}(\boldsymbol{\sigma}[n-1], B_{\mathbf{T}}[n])$$

Finally, we define Π as the new state given the block reward function Ω applied to the final transaction's resultant state, $\ell(\boldsymbol{\sigma})$:

(174) $$\Pi(\boldsymbol{\sigma}, B) \equiv \Omega(B, \ell(\boldsymbol{\sigma}))$$

Thus the complete block-transition mechanism is defined, except for PoW, the proof-of-work function.

11.5. **Mining Proof-of-Work.** The mining proof-of-work (PoW) exists as a cryptographically secure nonce that proves beyond reasonable doubt that a particular amount of computation has been expended in the determination of some token value n. It is utilised to enforce the blockchain security by giving meaning and credence to the notion of difficulty (and, by extension, total difficulty). However, since mining new blocks comes with an attached reward, the proof-of-work not only functions as a method of securing confidence that the blockchain will remain canonical into the future, but also as a wealth distribution mechanism.

For both reasons, there are two important goals of the proof-of-work function; firstly, it should be as accessible as possible to as many people as possible. The requirement of, or reward from, specialised and uncommon hardware should be minimised. This makes the distribution model as open as possible, and, ideally, makes the act of mining a simple swap from electricity to Ether at roughly the same rate for anyone around the world.

Secondly, it should not be possible to make super-linear profits, and especially not so with a high initial barrier. Such a mechanism allows a well-funded adversary to gain a troublesome amount of the network's total mining power and as such gives them a super-linear reward (thus skewing distribution in their favour) as well as reducing the network security.

One plague of the Bitcoin world is ASICs. These are specialised pieces of compute hardware that exist only to do a single task (Smith [1997]). In Bitcoin's case the task is the SHA256 hash function (Courtois et al. [2014]). While ASICs exist for a proof-of-work function, both goals are placed in jeopardy. Because of this, a proof-of-work function that is ASIC-resistant (i.e. difficult or economically inefficient to implement in specialised compute hardware) has been identified as the proverbial silver bullet.

Two directions exist for ASIC resistance; firstly make it sequential memory-hard, i.e. engineer the function such that the determination of the nonce requires a lot of memory and bandwidth such that the memory cannot be used in parallel to discover multiple nonces simultaneously. The second is to make the type of computation it would need to do general-purpose; the meaning of "specialised hardware" for a general-purpose task set is, naturally, general purpose hardware and as such commodity desktop computers are likely to be pretty close to "specialised hardware" for the task. For Ethereum 1.0 we have chosen the first path.

More formally, the proof-of-work function takes the form of PoW:

(175)
$$m = H_{\mathrm{m}} \quad \wedge \quad n \leqslant \frac{2^{256}}{H_{\mathrm{d}}} \quad \text{with} \quad (m, n) = \mathtt{PoW}(H_{\not n}, H_{\mathrm{n}}, \mathbf{d})$$

Where $H_{\not n}$ is the new block's header but *without* the nonce and mix-hash components; H_{n} is the nonce of the header; $\mathbf{d}$ is a large data set needed to compute the mixHash and H_{d} is the new block's difficulty value (i.e. the block difficulty from section 10). PoW is the proof-of-work function which evaluates to an array with the first item being the mixHash and the second item being a pseudo-random number cryptographically dependent on H and $\mathbf{d}$. The underlying algorithm is called Ethash and is described below.

11.5.1. *Ethash.* Ethash is the PoW algorithm for Ethereum 1.0. It is the latest version of Dagger-Hashimoto, introduced by Buterin [2013b] and Dryja [2014], although it can no longer appropriately be called that since many of the original features of both algorithms were drastically changed with R&D from February 2015 until May 4 2015 (Jentzsch [2015]). The general route that the algorithm takes is as follows:

There exists a seed which can be computed for each block by scanning through the block headers up until that point. From the seed, one can compute a pseudorandom cache, $J_{cacheinit}$ bytes in initial size. Light clients store the cache. From the cache, we can generate a dataset, $J_{datasetinit}$ bytes in initial size, with the property that each item in the dataset depends on only a small number of items from the cache. Full clients and miners store the dataset. The dataset grows linearly with time.

Mining involves grabbing random slices of the dataset and hashing them together. Verification can be done with low memory by using the cache to regenerate the specific pieces of the dataset that you need, so you only need to store the cache. The large dataset is updated once every J_{epoch} blocks, so the vast majority of a miner's effort will be reading the dataset, not making changes to it. The mentioned parameters as well as the algorithm is explained in detail in appendix J.

12. Implementing Contracts

There are several patterns of contracts engineering that allow particular useful behaviours; two of these that we will briefly discuss are data feeds and random numbers.

12.1. **Data Feeds.** A data feed contract is one which provides a single service: it gives access to information from the external world within Ethereum. The accuracy and timeliness of this information is not guaranteed and it is the task of a secondary contract author—the contract that utilises the data feed—to determine how much trust can be placed in any single data feed.

The general pattern involves a single contract within Ethereum which, when given a message call, replies with some timely information concerning an external phenomenon. An example might be the local temperature of New York City. This would be implemented as a contract that returned that value of some known point in storage. Of course this point in storage must be maintained with the correct such temperature, and thus the second part of the pattern would be for an external server to run an Ethereum node, and immediately on discovery of a new block, creates a new valid transaction, sent to the contract, updating said value in storage. The contract's code would accept such updates only from the identity contained on said server.

12.2. **Random Numbers.** Providing random numbers within a deterministic system is, naturally, an impossible task. However, we can approximate with pseudo-random numbers by utilising data which is generally unknowable at the time of transacting. Such data might include the block's hash, the block's timestamp and the block's beneficiary address. In order to make it hard for malicious miners to control those values, one should use the BLOCKHASH operation in order to use hashes of the previous 256 blocks as pseudo-random numbers. For a series of such numbers, a trivial solution would be to add some constant amount and hashing the result.

13. Future Directions

The state database won't be forced to maintain all past state trie structures into the future. It should maintain an age for each node and eventually discard nodes that are neither recent enough nor checkpoints. Checkpoints, or a set of nodes in the database that allow a particular block's state trie to be traversed, could be used to place a maximum limit on the amount of computation needed in order to retrieve any state throughout the blockchain.

Blockchain consolidation could be used in order to reduce the amount of blocks a client would need to download to act as a full, mining, node. A compressed archive of the trie structure at given points in time (perhaps one in every 10,000th block) could be maintained by the peer network, effectively recasting the genesis block. This would reduce the amount to be downloaded to a single archive plus a hard maximum limit of blocks.

Finally, blockchain compression could perhaps be conducted: nodes in state trie that haven't sent/received a transaction in some constant amount of blocks could be thrown out, reducing both Ether-leakage and the growth of the state database.

13.1. **Scalability.** Scalability remains an eternal concern. With a generalised state transition function, it becomes difficult to partition and parallelise transactions to apply the divide-and-conquer strategy. Unaddressed, the dynamic value-range of the system remains essentially fixed and as the average transaction value increases, the less valuable of them become ignored, being economically pointless to include in the main ledger. However, several strategies exist that may potentially be exploited to provide a considerably more scalable protocol.

Some form of hierarchical structure, achieved by either consolidating smaller lighter-weight chains into the main

block or building the main block through the incremental combination and adhesion (through proof-of-work) of smaller transaction sets may allow parallelisation of transaction combination and block-building. Parallelism could also come from a prioritised set of parallel blockchains, consolidating each block and with duplicate or invalid transactions thrown out accordingly.

Finally, verifiable computation, if made generally available and efficient enough, may provide a route to allow the proof-of-work to be the verification of final state.

14. Conclusion

We have introduced, discussed and formally defined the protocol of Ethereum. Through this protocol the reader may implement a node on the Ethereum network and join others in a decentralised secure social operating system. Contracts may be authored in order to algorithmically specify and autonomously enforce rules of interaction.

15. Acknowledgements

Many thanks to Aeron Buchanan for authoring the *Homestead* revisions, Christoph Jentzsch for authoring the Ethash algorithm and Yoichi Hirai for doing most of the EIP-150 changes. Important maintenance, useful corrections and suggestions were provided by a number of others from the Ethereum DEV organisation and Ethereum community at large including Gustav Simonsson, Paweł Bylica, Jutta Steiner, Nick Savers, Viktor Trón, Marko Simovic, Giacomo Tazzari and, of course, Vitalik Buterin.

16. Availability

The source of this paper is maintained at `https://github.com/ethereum/yellowpaper/`. An auto-generated PDF is located at `https://ethereum.github.io/yellowpaper/paper.pdf`.

References

Jacob Aron. BitCoin software finds new life. *New Scientist*, 213(2847):20, 2012. URL `http://www.sciencedirect.com/science/article/pii/S0262407912601055`.

Adam Back. Hashcash - Amortizable Publicly Auditable Cost-Functions, 2002. URL `http://www.hashcash.org/papers/amortizable.pdf`.

Guido Bertoni, Joan Daemen, Michal Peeters, Gilles Van Assche, and Ronny Van Keer. KECCAK, 2017. URL `https://keccak.team/keccak.html`.

Roman Boutellier and Mareike Heinzen. Pirates, Pioneers, Innovators and Imitators. In *Growth Through Innovation*, pages 85–96. Springer, 2014. URL `https://www.springer.com/gb/book/9783319040158`.

Vitalik Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform, 2013a. URL `https://github.com/ethereum/wiki/wiki/White-Paper`.

Vitalik Buterin. Dagger: A Memory-Hard to Compute, Memory-Easy to Verify Scrypt Alternative, 2013b. URL `http://www.hashcash.org/papers/dagger.html`.

Vitalik Buterin. EIP-2: Homestead hard-fork changes, 2015. URL `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2.md`.

Vitalik Buterin. EIP-100: Change difficulty adjustment to target mean block time including uncles, April 2016. URL `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-100.md`.

Nicolas T. Courtois, Marek Grajek, and Rahul Naik. *Optimizing SHA256 in Bitcoin Mining*, pages 131–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-44893-9. doi: 10.1007/978-3-662-44893-9_12. URL `https://doi.org/10.1007/978-3-662-44893-9_12`.

B.A. Davey and H.A. Priestley. *Introduction to lattices and order. 2nd ed.* Cambridge: Cambridge University Press, 2nd ed. edition, 2002. ISBN 0-521-78451-4/pbk.

Thaddeus Dryja. Hashimoto: I/O bound proof of work, 2014. URL `http://diyhpl.us/~bryan/papers2/bitcoin/meh/hashimoto.pdf`.

Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *In 12th Annual International Cryptology Conference*, pages 139–147, 1992. URL `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.pdf`.

Phong Vo Glenn Fowler, Landon Curt Noll. FowlerNollVo hash function, 1991. URL `http://www.isthe.com/chongo/tech/comp/fnv/index.html`.

Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 119–132. Springer, 2004. URL `https://www.iacr.org/archive/ches2004/31560117/31560117.pdf`.

Christoph Jentzsch. Commit date for ethash, 2015. URL `https://github.com/ethereum/yellowpaper/commit/77a8cf2428ce245bf6e2c39c5e652ba58a278666#commitcomment-24644869`.

Don Johnson, Alfred Menezes, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA), 2001. URL `https://web.archive.org/web/20170921160141/http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf`. Accessed 21 September 2017, but the original link was inaccessible on 19 October 2017. Refer to section 6.2 for ECDSAPUBKEY, and section 7 for ECDSASIGN and ECDSARECOVER.

Sergio Demian Lerner. Strict Memory Hard Hashing Functions, 2014. URL `http://www.hashcash.org/papers/memohash.pdf`.

Mark Miller. The Future of Law. In *paper delivered at the Extro 3 Conference (August 9)*, 1997. URL `https://drive.google.com/file/d/0Bw0VXJKBgYPMS0J2VGIyWWlocms/edit?usp=sharing`.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL `http://www.bitcoin.org/bitcoin.pdf`.

Meni Rosenfeld, Yoni Assia, Vitalik Buterin, m liorhakiLior, Oded Leiba, Assaf Shomer, and Eliran Zach. Colored Coins Protocol Specification, 2012. URL `https://github.com/Colored-Coins/Colored-Coins-Protocol-Specification`.

Afri Schoedon and Vitalik Buterin. EIP-649: Metropolis difficulty bomb delay and block reward reduction, June 2017. URL `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-649.md`.

Michael John Sebastian Smith. *Application-Specific Integrated Circuits*. Addison-Wesley, 1997. ISBN 0201500221.

Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains, 2013. URL https://eprint.iacr.org/2013/881.

Simon Sprankel. Technical Basis of Digital Currencies, 2013. URL http://www.coderblog.de/wp-content/uploads/technical-basis-of-digital-currencies.pdf.

Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997. URL http://firstmonday.org/ojs/index.php/fm/article/view/548.

Vivek Vishnumurthy, Sangeeth Chandrakumar, and Emin Gn Sirer. KARMA: A secure economic framework for peer-to-peer resource sharing, 2003. URL https://www.cs.cornell.edu/people/egs/papers/karma.pdf.

J. R. Willett. MasterCoin Complete Specification, 2013. URL https://github.com/mastercoin-MSC/spec.

Appendix A. Terminology

External Actor: A person or other entity able to interface to an Ethereum node, but external to the world of Ethereum. It can interact with Ethereum through depositing signed Transactions and inspecting the blockchain and associated state. Has one (or more) intrinsic Accounts.

Address: A 160-bit code used for identifying Accounts.

Account: Accounts have an intrinsic balance and transaction count maintained as part of the Ethereum state. They also have some (possibly empty) EVM Code and a (possibly empty) Storage State associated with them. Though homogenous, it makes sense to distinguish between two practical types of account: those with empty associated EVM Code (thus the account balance is controlled, if at all, by some external entity) and those with non-empty associated EVM Code (thus the account represents an Autonomous Object). Each Account has a single Address that identifies it.

Transaction: A piece of data, signed by an External Actor. It represents either a Message or a new Autonomous Object. Transactions are recorded into each block of the blockchain.

Autonomous Object: A notional object existent only within the hypothetical state of Ethereum. Has an intrinsic address and thus an associated account; the account will have non-empty associated EVM Code. Incorporated only as the Storage State of that account.

Storage State: The information particular to a given Account that is maintained between the times that the Account's associated EVM Code runs.

Message: Data (as a set of bytes) and Value (specified as Ether) that is passed between two Accounts, either through the deterministic operation of an Autonomous Object or the cryptographically secure signature of the Transaction.

Message Call: The act of passing a message from one Account to another. If the destination account is associated with non-empty EVM Code, then the VM will be started with the state of said Object and the Message acted upon. If the message sender is an Autonomous Object, then the Call passes any data returned from the VM operation.

Gas: The fundamental network cost unit. Paid for exclusively by Ether (as of PoC-4), which is converted freely to and from Gas as required. Gas does not exist outside of the internal Ethereum computation engine; its price is set by the Transaction and miners are free to ignore Transactions whose Gas price is too low.

Contract: Informal term used to mean both a piece of EVM Code that may be associated with an Account or an Autonomous Object.

Object: Synonym for Autonomous Object.

App: An end-user-visible application hosted in the Ethereum Browser.

Ethereum Browser: (aka Ethereum Reference Client) A cross-platform GUI of an interface similar to a simplified browser (a la Chrome) that is able to host sandboxed applications whose backend is purely on the Ethereum protocol.

Ethereum Virtual Machine: (aka EVM) The virtual machine that forms the key part of the execution model for an Account's associated EVM Code.

Ethereum Runtime Environment: (aka ERE) The environment which is provided to an Autonomous Object executing in the EVM. Includes the EVM but also the structure of the world state on which the EVM relies for certain I/O instructions including CALL & CREATE.

EVM Code: The bytecode that the EVM can natively execute. Used to formally specify the meaning and ramifications of a message to an Account.

EVM Assembly: The human-readable form of EVM-code.

LLL: The Lisp-like Low-level Language, a human-writable language used for authoring simple contracts and general low-level language toolkit for trans-compiling to.

Appendix B. Recursive Length Prefix

This is a serialisation method for encoding arbitrarily structured binary data (byte arrays).

We define the set of possible structures $\mathbb{T}$:

$$\mathbb{T} \equiv \mathbb{L} \cup \mathbb{B} \tag{176}$$

$$\mathbb{L} \equiv \{\mathbf{t} : \mathbf{t} = (\mathbf{t}[0], \mathbf{t}[1], ...) \ \wedge \ \forall_{n<\|\mathbf{t}\|} \ \mathbf{t}[n] \in \mathbb{T}\} \tag{177}$$

$$\mathbb{B} \equiv \{\mathbf{b} : \mathbf{b} = (\mathbf{b}[0], \mathbf{b}[1], ...) \ \wedge \ \forall_{n<\|\mathbf{b}\|} \ \mathbf{b}[n] \in \mathbb{O}\} \tag{178}$$

Where $\mathbb{O}$ is the set of (8-bit) bytes. Thus $\mathbb{B}$ is the set of all sequences of bytes (otherwise known as byte-arrays, and a leaf if imagined as a tree), $\mathbb{L}$ is the set of all tree-like (sub-)structures that are not a single leaf (a branch node if imagined as a tree) and $\mathbb{T}$ is the set of all byte-arrays and such structural sequences.

We define the RLP function as $\texttt{RLP}$ through two sub-functions, the first handling the instance when the value is a byte array, the second when it is a sequence of further values:

$$\texttt{RLP}(\mathbf{x}) \equiv \begin{cases} R_{\text{b}}(\mathbf{x}) & \text{if} \quad \mathbf{x} \in \mathbb{B} \\ R_{\text{l}}(\mathbf{x}) & \text{otherwise} \end{cases} \tag{179}$$

If the value to be serialised is a byte-array, the RLP serialisation takes one of three forms:

- If the byte-array contains solely a single byte and that single byte is less than 128, then the input is exactly equal to the output.
- If the byte-array contains fewer than 56 bytes, then the output is equal to the input prefixed by the byte equal to the length of the byte array plus 128.
- Otherwise, the output is equal to the input prefixed by the minimal-length byte-array which when interpreted as a big-endian integer is equal to the length of the input byte array, which is itself prefixed by the number of bytes required to faithfully encode this length value plus 183.

Formally, we define R_{b}:

$$R_{\text{b}}(\mathbf{x}) \equiv \begin{cases} \mathbf{x} & \text{if} \quad \|\mathbf{x}\| = 1 \wedge \mathbf{x}[0] < 128 \\ (128 + \|\mathbf{x}\|) \cdot \mathbf{x} & \text{else if} \quad \|\mathbf{x}\| < 56 \\ \left(183 + \left\|\texttt{BE}(\|\mathbf{x}\|)\right\|\right) \cdot \texttt{BE}(\|\mathbf{x}\|) \cdot \mathbf{x} & \text{otherwise} \end{cases} \tag{180}$$

$$\texttt{BE}(x) \equiv (b_0, b_1, ...) : b_0 \neq 0 \wedge x = \sum_{n=0}^{n<\|\mathbf{b}\|} b_{\text{n}} \cdot 256^{\|\mathbf{b}\|-1-n} \tag{181}$$

$$(a) \cdot (b, c) \cdot (d, e) = (a, b, c, d, e) \tag{182}$$

Thus $\texttt{BE}$ is the function that expands a non-negative integer value to a big-endian byte array of minimal length and the dot operator performs sequence concatenation.

If instead, the value to be serialised is a sequence of other items then the RLP serialisation takes one of two forms:

- If the concatenated serialisations of each contained item is less than 56 bytes in length, then the output is equal to that concatenation prefixed by the byte equal to the length of this byte array plus 192.
- Otherwise, the output is equal to the concatenated serialisations prefixed by the minimal-length byte-array which when interpreted as a big-endian integer is equal to the length of the concatenated serialisations byte array, which is itself prefixed by the number of bytes required to faithfully encode this length value plus 247.

Thus we finish by formally defining R_{l}:

$$R_{\text{l}}(\mathbf{x}) \equiv \begin{cases} (192 + \|s(\mathbf{x})\|) \cdot s(\mathbf{x}) & \text{if} \quad \|s(\mathbf{x})\| < 56 \\ \left(247 + \left\|\texttt{BE}(\|s(\mathbf{x})\|)\right\|\right) \cdot \texttt{BE}(\|s(\mathbf{x})\|) \cdot s(\mathbf{x}) & \text{otherwise} \end{cases} \tag{183}$$

$$s(\mathbf{x}) \equiv \texttt{RLP}(\mathbf{x}_0) \cdot \texttt{RLP}(\mathbf{x}_1)... \tag{184}$$

If RLP is used to encode a scalar, defined only as a non-negative integer ($\mathbb{N}$ or any x for $\mathbb{N}_{\text{x}}$), it must be specified as the shortest byte array such that the big-endian interpretation of it is equal. Thus the RLP of some non-negative integer i is defined as:

$$\texttt{RLP}(i : i \in \mathbb{N}) \equiv \texttt{RLP}(\texttt{BE}(i)) \tag{185}$$

When interpreting RLP data, if an expected fragment is decoded as a scalar and leading zeroes are found in the byte sequence, clients are required to consider it non-canonical and treat it in the same manner as otherwise invalid RLP data, dismissing it completely.

There is no specific canonical encoding format for signed or floating-point values.

Appendix C. Hex-Prefix Encoding

Hex-prefix encoding is an efficient method of encoding an arbitrary number of nibbles as a byte array. It is able to store an additional flag which, when used in the context of the trie (the only context in which it is used), disambiguates between node types.

It is defined as the function $\texttt{HP}$ which maps from a sequence of nibbles (represented by the set $\mathbb{Y}$) together with a boolean value to a sequence of bytes (represented by the set $\mathbb{B}$):

$$\text{HP}(\mathbf{x}, t) : \mathbf{x} \in \mathbb{Y} \equiv \begin{cases} (16f(t), 16\mathbf{x}[0] + \mathbf{x}[1], 16\mathbf{x}[2] + \mathbf{x}[3], ...) & \text{if } \|\mathbf{x}\| \text{ is even} \\ (16(f(t) + 1) + \mathbf{x}[0], 16\mathbf{x}[1] + \mathbf{x}[2], 16\mathbf{x}[3] + \mathbf{x}[4], ...) & \text{otherwise} \end{cases} \tag{186}$$

$$f(t) \equiv \begin{cases} 2 & \text{if } t \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{187}$$

Thus the high nibble of the first byte contains two flags; the lowest bit encoding the oddness of the length and the second-lowest encoding the flag t. The low nibble of the first byte is zero in the case of an even number of nibbles and the first nibble in the case of an odd number. All remaining nibbles (now an even number) fit properly into the remaining bytes.

Appendix D. Modified Merkle Patricia Tree

The modified Merkle Patricia tree (trie) provides a persistent data structure to map between arbitrary-length binary data (byte arrays). It is defined in terms of a mutable data structure to map between 256-bit binary fragments and arbitrary-length binary data, typically implemented as a database. The core of the trie, and its sole requirement in terms of the protocol specification is to provide a single value that identifies a given set of key-value pairs, which may be either a 32-byte sequence or the empty byte sequence. It is left as an implementation consideration to store and maintain the structure of the trie in a manner that allows effective and efficient realisation of the protocol.

Formally, we assume the input value $\mathfrak{I}$, a set containing pairs of byte sequences with unique keys:

$$\mathfrak{I} = \{(\mathbf{k}_0 \in \mathbb{B}, \mathbf{v}_0 \in \mathbb{B}), (\mathbf{k}_1 \in \mathbb{B}, \mathbf{v}_1 \in \mathbb{B}), ...\} \tag{188}$$

When considering such a sequence, we use the common numeric subscript notation to refer to a tuple's key or value, thus:

$$\forall_{\mathbf{I} \in \mathfrak{I}} I \equiv (I_0, I_1) \tag{189}$$

Any series of bytes may also trivially be viewed as a series of nibbles, given an endian-specific notation; here we assume big-endian. Thus:

$$y(\mathfrak{I}) = \{(\mathbf{k}'_0 \in \mathbb{Y}, \mathbf{v}_0 \in \mathbb{B}), (\mathbf{k}'_1 \in \mathbb{Y}, \mathbf{v}_1 \in \mathbb{B}), ...\} \tag{190}$$

$$\forall_n \quad \forall_{i : i < 2\|\mathbf{k}_n\|} \quad \mathbf{k}'_n[i] \equiv \begin{cases} \lfloor \mathbf{k}_n[i \div 2] \div 16 \rfloor & \text{if } i \text{ is even} \\ \mathbf{k}_n[\lfloor i \div 2 \rfloor] \bmod 16 & \text{otherwise} \end{cases} \tag{191}$$

We define the function TRIE, which evaluates to the root of the trie that represents this set when encoded in this structure:

$$\text{TRIE}(\mathfrak{I}) \equiv \text{KEC}(c(\mathfrak{I}, 0)) \tag{192}$$

We also assume a function n, the trie's node cap function. When composing a node, we use RLP to encode the structure. As a means of reducing storage complexity, for nodes whose composed RLP is fewer than 32 bytes, we store the RLP directly; for those larger we assert prescience of the byte array whose Keccak hash evaluates to our reference. Thus we define in terms of c, the node composition function:

$$n(\mathfrak{I}, i) \equiv \begin{cases} () & \text{if } \mathfrak{I} = \varnothing \\ c(\mathfrak{I}, i) & \text{if } \|c(\mathfrak{I}, i)\| < 32 \\ \text{KEC}(c(\mathfrak{I}, i)) & \text{otherwise} \end{cases} \tag{193}$$

In a manner similar to a radix tree, when the trie is traversed from root to leaf, one may build a single key-value pair. The key is accumulated through the traversal, acquiring a single nibble from each branch node (just as with a radix tree). Unlike a radix tree, in the case of multiple keys sharing the same prefix or in the case of a single key having a unique suffix, two optimising nodes are provided. Thus while traversing, one may potentially acquire multiple nibbles from each of the other two node types, extension and leaf. There are three kinds of nodes in the trie:

Leaf: A two-item structure whose first item corresponds to the nibbles in the key not already accounted for by the accumulation of keys and branches traversed from the root. The hex-prefix encoding method is used and the second parameter to the function is required to be *true*.

Extension: A two-item structure whose first item corresponds to a series of nibbles of size greater than one that are shared by at least two distinct keys past the accumulation of the keys of nibbles and the keys of branches as traversed from the root. The hex-prefix encoding method is used and the second parameter to the function is required to be *false*.

Branch: A 17-item structure whose first sixteen items correspond to each of the sixteen possible nibble values for the keys at this point in their traversal. The 17th item is used in the case of this being a terminator node and thus a key being ended at this point in its traversal.

A branch is then only used when necessary; no branch nodes may exist that contain only a single non-zero entry. We may formally define this structure with the structural composition function c:

(194)
$$c(\mathfrak{I}, i) \equiv \begin{cases} \texttt{RLP}\Big(\big(\texttt{HP}(I_0[i..(\|I_0\|-1)], true), I_1\big)\Big) & \text{if} \quad \|\mathfrak{I}\| = 1 \quad \text{where } \exists I : I \in \mathfrak{I} \\ \texttt{RLP}\Big(\big(\texttt{HP}(I_0[i..(j-1)], false), n(\mathfrak{I}, j)\big)\Big) & \text{if} \quad i \neq j \quad \text{where } j = \arg\max_x : \exists \mathbf{l} : \|\mathbf{l}\| = x : \forall_{I \in \mathfrak{I}} : I_0[0..(x-1)] = \mathbf{l} \\ \texttt{RLP}\Big((u(0), u(1), ..., u(15), v)\Big) & \text{otherwise} \quad \text{where } u(j) \equiv n(\{I : I \in \mathfrak{I} \wedge I_0[i] = j\}, i+1) \\ & \qquad\qquad\qquad\quad v = \begin{cases} I_1 & \text{if} \quad \exists I : I \in \mathfrak{I} \wedge \|I_0\| = i \\ () & \text{otherwise} \end{cases} \end{cases}$$

D.1. **Trie Database.** Thus no explicit assumptions are made concerning what data is stored and what is not, since that is an implementation-specific consideration; we simply define the identity function mapping the key-value set $\mathfrak{I}$ to a 32-byte hash and assert that only a single such hash exists for any $\mathfrak{I}$, which though not strictly true is accurate within acceptable precision given the Keccak hash's collision resistance. In reality, a sensible implementation will not fully recompute the trie root hash for each set.

A reasonable implementation will maintain a database of nodes determined from the computation of various tries or, more formally, it will memoise the function c. This strategy uses the nature of the trie to both easily recall the contents of any previous key-value set and to store multiple such sets in a very efficient manner. Due to the dependency relationship, Merkle-proofs may be constructed with an $O(\log N)$ space requirement that can demonstrate a particular leaf must exist within a trie of a given root hash.

Appendix E. Precompiled Contracts

For each precompiled contract, we make use of a template function, $\Xi_{\texttt{PRE}}$, which implements the out-of-gas checking.

(195)
$$\Xi_{\texttt{PRE}}(\boldsymbol{\sigma}, g, I, T) \equiv \begin{cases} (\varnothing, 0, A^0, ()) & \text{if} \quad g < g_{\text{r}} \\ (\boldsymbol{\sigma}, g - g_{\text{r}}, A^0, \mathbf{o}) & \text{otherwise} \end{cases}$$

The precompiled contracts each use these definitions and provide specifications for the $\mathbf{o}$ (the output data) and g_{r}, the gas requirements.

We define $\Xi_{\texttt{ECREC}}$ as a precompiled contract for the elliptic curve digital signature algorithm (ECDSA) public key recovery function (ecrecover). See Appendix F for the definition of the function $\texttt{ECDSARECOVER}$. We also define $\mathbf{d}$ to be the input data, well-defined for an infinite length by appending zeroes as required. In the case of an invalid signature ($\texttt{ECDSARECOVER}(h, v, r, s) = \varnothing$), we return no output.

(196) $\Xi_{\texttt{ECREC}} \equiv \Xi_{\texttt{PRE}}$ where:

(197) $g_{\text{r}} = 3000$

(198) $$|\mathbf{o}| = \begin{cases} 0 & \text{if} \quad \texttt{ECDSARECOVER}(h, v, r, s) = \varnothing \\ 32 & \text{otherwise} \end{cases}$$

(199) if $|\mathbf{o}| = 32$:

(200) $\mathbf{o}[0..11] = 0$

(201) $\mathbf{o}[12..31] = \texttt{KEC}\big(\texttt{ECDSARECOVER}(h, v, r, s)\big)[12..31]$ where:

(202) $\mathbf{d}[0..(|I_{\mathbf{d}}|-1)] = I_{\mathbf{d}}$

(203) $\mathbf{d}[|I_{\mathbf{d}}|..] = (0, 0, ...)$

(204) $h = \mathbf{d}[0..31]$

(205) $v = \mathbf{d}[32..63]$

(206) $r = \mathbf{d}[64..95]$

(207) $s = \mathbf{d}[96..127]$

We define $\Xi_{\texttt{SHA256}}$ and $\Xi_{\texttt{RIP160}}$ as precompiled contracts implementing the SHA2-256 and RIPEMD-160 hash functions respectively. Their gas usage is dependent on the input data size, a factor rounded up to the nearest number of words.

$$\Xi_{\text{SHA256}} \equiv \Xi_{\text{PRE}} \quad \text{where:} \tag{208}$$
$$g_{\text{r}} = 60 + 12\left\lceil \frac{|I_{\mathbf{d}}|}{32} \right\rceil \tag{209}$$
$$\mathbf{o}[0..31] = \texttt{SHA256}(I_{\mathbf{d}}) \tag{210}$$
$$\Xi_{\text{RIP160}} \equiv \Xi_{\text{PRE}} \quad \text{where:} \tag{211}$$
$$g_{\text{r}} = 600 + 120\left\lceil \frac{|I_{\mathbf{d}}|}{32} \right\rceil \tag{212}$$
$$\mathbf{o}[0..11] = 0 \tag{213}$$
$$\mathbf{o}[12..31] = \texttt{RIPEMD160}(I_{\mathbf{d}}) \tag{214}$$

For the purposes here, we assume we have well-defined standard cryptographic functions for RIPEMD-160 and SHA2-256 of the form:

$$\texttt{SHA256}(\mathbf{i} \in \mathbb{B}) \equiv o \in \mathbb{B}_{32} \tag{215}$$
$$\texttt{RIPEMD160}(\mathbf{i} \in \mathbb{B}) \equiv o \in \mathbb{B}_{20} \tag{216}$$

The fourth contract, the identity function Ξ_{ID} simply defines the output as the input:

$$\Xi_{\text{ID}} \equiv \Xi_{\text{PRE}} \quad \text{where:} \tag{217}$$
$$g_{\text{r}} = 15 + 3\left\lceil \frac{|I_{\mathbf{d}}|}{32} \right\rceil \tag{218}$$
$$\mathbf{o} = I_{\mathbf{d}} \tag{219}$$

The fifth contract performs arbitrary-precision exponentiation under modulo. Here, 0^0 is taken to be one, and $x \bmod 0$ is zero for all x. The first word in the input specifies the number of bytes that the first non-negative integer B occupies. The second word in the input specifies the number of bytes that the second non-negative integer E occupies. The third word in the input specifies the number of bytes that the third non-negative integer M occupies. These three words are followed by B, E and M. The rest of the input is discarded. Whenever the input is too short, the missing bytes are considered to be zero. The output is encoded big-endian into the same format as M's.

$$\Xi_{\text{EXPMOD}} \equiv \Xi_{\text{PRE}} \quad \text{except:} \tag{220}$$
$$g_{\text{r}} = \left\lfloor \frac{f\big(\max(\ell_M, \ell_B)\big)\max(\ell'_E, 1)}{G_{quaddivisor}} \right\rfloor \tag{221}$$
$$f(x) \equiv \begin{cases} x^2 & \text{if } x \le 64 \\ \left\lfloor \frac{x^2}{4} \right\rfloor + 96x - 3072 & \text{if } 64 < x \le 1024 \\ \left\lfloor \frac{x^2}{16} \right\rfloor + 480x - 199680 & \text{otherwise} \end{cases} \tag{222}$$
$$\ell'_E = \begin{cases} 0 & \text{if } \ell_E \le 32 \wedge E = 0 \\ \lfloor \log_2(E) \rfloor & \text{if } \ell_E \le 32 \wedge E \ne 0 \\ 8(\ell_E - 32) + \lfloor \log_2(i[(96+\ell_B)..(127+\ell_B)]) \rfloor & \text{if } 32 < \ell_E \wedge i[(96+\ell_B)..(127+\ell_B)] \ne 0 \\ 8(\ell_E - 32) & \text{otherwise} \end{cases} \tag{223}$$
$$\mathbf{o} = \left(B^E \bmod M\right) \in \mathbb{N}_{8\ell_M} \tag{224}$$
$$\ell_B \equiv i[0..31] \tag{225}$$
$$\ell_E \equiv i[32..63] \tag{226}$$
$$\ell_M \equiv i[64..95] \tag{227}$$
$$B \equiv i[96..(95+\ell_B)] \tag{228}$$
$$E \equiv i[(96+\ell_B)..(95+\ell_B+\ell_E)] \tag{229}$$
$$M \equiv i[(96+\ell_B+\ell_E)..(95+\ell_B+\ell_E+\ell_M)] \tag{230}$$
$$i[x] \equiv \begin{cases} I_{\mathbf{d}}[x] & \text{if } x < |I_{\mathbf{d}}| \\ 0 & \text{otherwise} \end{cases} \tag{231}$$

E.1. **zkSNARK Related Precompiled Contracts.** We choose two numbers, both of which are prime.

$$p \equiv 21888242871839275222246405745257275088696311157297823662689037894645226208583 \tag{232}$$
$$q \equiv 21888242871839275222246405745257275088548364400416034343698204186575808495617 \tag{233}$$

Since p is a prime number, $\{0, 1, \ldots, p-1\}$ forms a field with addition and multiplication modulo p. We call this field F_{p}.

We define a set C_1 with

$$C_1 \equiv \{(X, Y) \in F_\mathrm{p} \times F_\mathrm{p} \mid Y^2 = X^3 + 3\} \cup \{(0, 0)\} \tag{234}$$

We define a binary operation $+$ on C_1 for distinct elements $(X_1, Y_1), (X_2, Y_2)$ with

$$\begin{aligned}(X_1, Y_1) + (X_2, Y_2) &\equiv \begin{cases}(X, Y) & \text{if } X_1 \neq X_2\\ (0, 0) & \text{otherwise}\end{cases}\\ \lambda &\equiv \frac{Y_2 - Y_1}{X_2 - X_1}\\ X &\equiv \lambda^2 - X_1 - X_2\\ Y &\equiv \lambda(X_1 - X) - Y_1\end{aligned} \tag{235}$$

In the case where $(X_1, Y_1) = (X_2, Y_2)$, we define $+$ on C_1 with

$$\begin{aligned}(X_1, Y_1) + (X_2, Y_2) &\equiv \begin{cases}(X, Y) & \text{if } Y_1 \neq 0\\ (0, 0) & \text{otherwise}\end{cases}\\ \lambda &\equiv \frac{3X_1^2}{2Y_1}\\ X &\equiv \lambda^2 - 2X_1\\ Y &\equiv \lambda(X_1 - X) - Y_1\end{aligned} \tag{236}$$

$(C_1, +)$ is known to form a group. We define scalar multiplication $\cdot$ with

$$n \cdot P \equiv (0, 0) + \underbrace{P + \cdots + P}_{n} \tag{237}$$

for a natural number n and a point P in C_1.

We define P_1 to be a point $(1, 2)$ on C_1. Let G_1 be the subgroup of $(C_1, +)$ generated by P_1. G_1 is known to be a cyclic group of order q. For a point P in G_1, we define $\log_{P_1}(P)$ to be the smallest natural number n satisfying $n \cdot P_1 = P$. $\log_{P_1}(P)$ is at most $q - 1$.

Let F_{p^2} be a field $F_p[i]/(i^2 + 1)$. We define a set C_2 with

$$C_2 \equiv \{(X, Y) \in F_{p^2} \times F_{p^2} \mid Y^2 = X^3 + 3(i + 9)^{-1}\} \cup \{(0, 0)\} \tag{238}$$

We define a binary operation $+$ and scalar multiplication $\cdot$ with the same equations (235), (236) and (237). $(C_2, +)$ is also known to be a group. We define P_2 in C_2 with

$$\begin{aligned}P_2 \equiv\ &(11559732032986387107991004021392285783925812861821192530917403151452391805634 \times i\\ &+10857046999023057135944570762232829481370756359578518086990519993285655852781,\\ &4082367875863433681332203403145435568316851327593401208105741076214120093531 \times i\\ &+84956539231234314176049732474892724384181905872636001487702806493069581019 30)\end{aligned} \tag{239}$$

We define G_2 to be the subgroup of $(C_2, +)$ generated by P_2. G_2 is known to be the only cyclic group of order q on C_2. For a point P in G_2, we define $\log_{P_2}(P)$ be the smallest natural number n satisfying $n \cdot P_2 = P$. With this definition, $\log_{P_2}(P)$ is at most $q - 1$.

Let G_T be the multiplicative abelian group underlying $F_{q^{12}}$. It is known that a non-degenerate bilinear map $e : G_1 \times G_2 \to G_T$ exists. This bilinear map is a type three pairing. There are several such bilinear maps, it does not matter which is chosen to be e. Let $P_T = e(P_1, P_2)$, a be a set of k points in G_1, and b be a set of k points in G_2. It follows from the definition of a pairing that the following are equivalent

$$\log_{P_1}(a_1) \times \log_{P_2}(b_1) + \cdots + \log_{P_1}(a_k) \times \log_{P_2}(b_k) \equiv 1 \mod q \tag{240}$$

$$\prod_{i=0}^{k} e(a_i, b_i) = P_T \tag{241}$$

Thus the pairing operation provides a method to verify (240).

A 32 byte number $\mathbf{x} \in \mathbf{P}_{256}$ might and might not represent an element of F_p.

$$\delta_\mathrm{p}(\mathbf{x}) \equiv \begin{cases}\mathbf{x} & \text{if } \mathbf{x} < p\\ \varnothing & \text{otherwise}\end{cases} \tag{242}$$

A 64 byte data $\mathbf{x} \in \mathbf{B}_{512}$ might and might not represent an element of G_1.

$$\delta_1(\mathbf{x}) \equiv \begin{cases} g_1 & \text{if } g_1 \in G_1 \\ \varnothing & \text{otherwise} \end{cases} \tag{243}$$

$$g_1 \equiv \begin{cases} (x, y) & \text{if } x \neq \varnothing \wedge y \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \tag{244}$$

$$x \equiv \delta_{\mathrm{p}}(\mathbf{x}[0..31]) \tag{245}$$

$$y \equiv \delta_{\mathrm{p}}(\mathbf{x}[32..63]) \tag{246}$$

A 128 byte data $\mathbf{x} \in \mathbf{B}_{1024}$ might and might not represent an element of G_2.

$$\delta_2(\mathbf{x}) \equiv \begin{cases} g_2 & \text{if } g_2 \in G_2 \\ \varnothing & \text{otherwise} \end{cases} \tag{247}$$

$$g_2 \equiv \begin{cases} ((x_0 i + y_0), (x_1 i + y_1)) & \text{if } x_0 \neq \varnothing \wedge y_0 \neq \varnothing \wedge x_1 \neq \varnothing \wedge y_1 \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \tag{248}$$

$$x_0 \equiv \delta_{\mathrm{p}}(\mathbf{x}[0..31]) \tag{249}$$

$$y_0 \equiv \delta_{\mathrm{p}}(\mathbf{x}[32..63]) \tag{250}$$

$$x_1 \equiv \delta_{\mathrm{p}}(\mathbf{x}[64..95]) \tag{251}$$

$$y_1 \equiv \delta_{\mathrm{p}}(\mathbf{x}[96..127]) \tag{252}$$

We define $\Xi_{\texttt{SNARKV}}$ as a precompiled contract which checks if (240) holds, for intended use in zkSNARK verification.

$$\Xi_{\texttt{SNARKV}} \equiv \Xi_{\texttt{PRE}} \quad \text{except:} \tag{253}$$

$$\Xi_{\texttt{SNARKV}}(\boldsymbol{\sigma}, g, I) = \left(\varnothing, 0, A^0, ()\right) \quad \text{if } F \tag{254}$$

$$F \equiv (|I_{\mathbf{d}}| \bmod 192 \neq 0 \vee (\exists j.\ a_{\mathrm{j}} = \varnothing \vee b_{\mathrm{j}} = \varnothing)) \tag{255}$$

$$k = \frac{|I_{\mathbf{d}}|}{192} \tag{256}$$

$$g_{\mathrm{r}} = 60000k + 40000 \tag{257}$$

$$\mathbf{o}[0..31] \equiv \begin{cases} 0x0000000000000000000000000000000000000000000000000000000000000001 & \text{if } v \wedge \neg F \\ 0x0000000000000000000000000000000000000000000000000000000000000000 & \text{if } \neg v \wedge \neg F \end{cases} \tag{258}$$

$$v \equiv (\log_{P_1}(a_1) \times \log_{P_2}(b_1) + \cdots + \log_{P_1}(a_k) \times \log_{P_2}(b_k) \equiv 1 \mod q) \tag{259}$$

$$a_1 \equiv \delta_1(I_{\mathbf{d}}[0..63]) \tag{260}$$

$$b_1 \equiv \delta_2(I_{\mathbf{d}}[64..191]) \tag{261}$$

$$\vdots$$

$$a_k \equiv \delta_1(I_{\mathbf{d}}[(|I_{\mathbf{d}}| - 192)..(|I_{\mathbf{d}}| - 129)]) \tag{262}$$

$$b_k \equiv \delta_2(I_{\mathbf{d}}[(|I_{\mathbf{d}}| - 128)..(|I_{\mathbf{d}}| - 1)]) \tag{263}$$

We define a precompiled contract for addition on G_1.

$$\Xi_{\texttt{BN\_ADD}} \equiv \Xi_{\texttt{BN\_PRE}} \quad \text{except:} \tag{264}$$

$$\Xi_{\texttt{BN\_ADD}}(\boldsymbol{\sigma}, g, I) = \left(\varnothing, 0, A^0, ()\right) \quad \text{if } x = \varnothing \vee y = \varnothing \tag{265}$$

$$g_{\mathrm{r}} = 500 \tag{266}$$

$$\mathbf{o} \equiv \delta_1^{-1}(x + y) \quad \text{where } + \text{ is the group operation in } G_1 \tag{267}$$

$$x \equiv \delta_1\left(\bar{I}_{\mathbf{d}}[0..63]\right) \tag{268}$$

$$y \equiv \delta_1\left(\bar{I}_{\mathbf{d}}[64..127]\right) \tag{269}$$

$$\bar{I}_{\mathbf{d}}[x] \equiv \begin{cases} I_{\mathbf{d}}[x] & \text{if } x < |I_{\mathbf{d}}| \\ 0 & \text{otherwise} \end{cases} \tag{270}$$

We define a precompiled contract for scalar multiplication on G_1, where $\bar{I}_{\mathbf{d}}$ is defined in (270).

$$\Xi_{\texttt{BN\_MUL}} \equiv \Xi_{\texttt{PRE}} \quad \text{except:} \tag{271}$$

$$\Xi_{\texttt{BN\_MUL}}(\boldsymbol{\sigma}, g, I) = \left(\varnothing, 0, A^0, ()\right) \quad \text{if } x = \varnothing \tag{272}$$

$$g_{\mathrm{r}} = 40000 \tag{273}$$

$$\mathbf{o} \equiv \delta_1^{-1}(n \cdot x) \quad \text{where } \cdot \text{ is the scalar multiplication in } G_1 \tag{274}$$

$$n \equiv \bar{I}_{\mathbf{d}}[0..31] \tag{275}$$

$$x \equiv \delta_1\left(\bar{I}_{\mathbf{d}}[32..95]\right) \tag{276}$$

Appendix F. Signing Transactions

Transactions are signed using recoverable ECDSA signatures. This method utilises the SECP-256k1 curve as described by Courtois et al. [2014], and is implemented similarly to as described by Gura et al. [2004] on p. 9 of 15, para. 3.

It is assumed that the sender has a valid private key p_{r}, which is a randomly selected positive integer (represented as a byte array of length 32 in big-endian form) in the range $[1, \texttt{secp256k1n} - 1]$.

We assume the existence of functions ECDSAPUBKEY, ECDSASIGN and ECDSARECOVER. These are formally defined in the literature, e.g. by Johnson et al. [2001].

$$\texttt{ECDSAPUBKEY}(p_{\mathrm{r}} \in \mathbb{B}_{32}) \equiv p_{\mathrm{u}} \in \mathbb{B}_{64} \tag{277}$$

$$\texttt{ECDSASIGN}(e \in \mathbb{B}_{32}, p_{\mathrm{r}} \in \mathbb{B}_{32}) \equiv (v \in \mathbb{B}_1, r \in \mathbb{B}_{32}, s \in \mathbb{B}_{32}) \tag{278}$$

$$\texttt{ECDSARECOVER}(e \in \mathbb{B}_{32}, v \in \mathbb{B}_1, r \in \mathbb{B}_{32}, s \in \mathbb{B}_{32}) \equiv p_{\mathrm{u}} \in \mathbb{B}_{64} \tag{279}$$

Where p_{u} is the public key, assumed to be a byte array of size 64 (formed from the concatenation of two positive integers each $< 2^{256}$), p_{r} is the private key, a byte array of size 32 (or a single positive integer in the aforementioned range) and e is the hash of the transaction, $h(T)$. It is assumed that v is the 'recovery identifier'. The recovery identifier is a 1 byte value specifying the parity and finiteness of the coordinates of the curve point for which r is the x-value; this value is in the range of $[27, 30]$, however we declare the upper two possibilities, representing infinite values, invalid. The value 27 represents an even y value and 28 represents an odd y value.

We declare that an ECDSA signature is invalid unless all the following conditions are true[5]:

$$0 < r < \texttt{secp256k1n} \tag{280}$$

$$0 < s < \texttt{secp256k1n} \div 2 + 1 \tag{281}$$

$$v \in \{27, 28\} \tag{282}$$

where:

$$\texttt{secp256k1n} = 115792089237316195423570985008687907852837564279074904382605163141518161494337 \tag{283}$$

For a given private key, p_{r}, the Ethereum address $A(p_{\mathrm{r}})$ (a 160-bit value) to which it corresponds is defined as the right most 160-bits of the Keccak hash of the corresponding ECDSA public key:

$$A(p_{\mathrm{r}}) = \mathcal{B}_{96..255}\big(\texttt{KEC}\big(\texttt{ECDSAPUBKEY}(p_{\mathrm{r}})\big)\big) \tag{284}$$

The message hash, $h(T)$, to be signed is the Keccak hash of the transaction. Two different flavors of signing schemes are available. One operates without the latter three signature components, formally described as T_{r}, T_{s} and T_{w}. The other operates on nine elements:

$$L_S(T) \equiv \begin{cases} (T_{\mathrm{n}}, T_{\mathrm{p}}, T_{\mathrm{g}}, T_{\mathrm{t}}, T_{\mathrm{v}}, \mathbf{p}) & \text{if } v \in \{27, 28\} \\ (T_{\mathrm{n}}, T_{\mathrm{p}}, T_{\mathrm{g}}, T_{\mathrm{t}}, T_{\mathrm{v}}, \mathbf{p}, \texttt{chain\_id}, (), ()) & \text{otherwise} \end{cases} \tag{285}$$

where

$$\mathbf{p} \equiv \begin{cases} T_{\mathbf{i}} & \text{if } T_{\mathrm{t}} = 0 \\ T_{\mathbf{d}} & \text{otherwise} \end{cases}$$

$$h(T) \equiv \texttt{KEC}(L_S(T)) \tag{286}$$

The signed transaction $G(T, p_{\mathrm{r}})$ is defined as:

$$G(T, p_{\mathrm{r}}) \equiv T \quad \text{except:} \tag{287}$$

$$(T_{\mathrm{w}}, T_{\mathrm{r}}, T_{\mathrm{s}}) = \texttt{ECDSASIGN}(h(T), p_{\mathrm{r}}) \tag{288}$$

Reiterating from previously:

$$T_{\mathrm{r}} = r \tag{289}$$

$$T_{\mathrm{s}} = s \tag{290}$$

T_{w} is either the recovery identifier or 'chain identifier doubled plus 35 or 36'. In the second case, where v is the chain identifier doubled plus 35 or 36, the values 35 and 36 assume the role of the 'recovery identifier' by specifying the parity of y, with the value 35 representing an even value and 36 representing an odd value.

We may then define the sender function S of the transaction as:

$$S(T) \equiv \mathcal{B}_{96..255}\big(\texttt{KEC}\big(\texttt{ECDSARECOVER}(h(T), v_0, T_{\mathrm{r}}, T_{\mathrm{s}})\big)\big) \tag{291}$$

$$v_0 \equiv \begin{cases} T_{\mathrm{w}} & \text{if } T_{\mathrm{w}} \in \{27, 28\} \\ 28 - (T_{\mathrm{w}} \bmod 2) & \text{otherwise} \end{cases} \tag{292}$$

The assertion that the sender of a signed transaction equals the address of the signer should be self-evident:

$$\forall T : \forall p_{\mathrm{r}} : S(G(T, p_{\mathrm{r}})) \equiv A(p_{\mathrm{r}}) \tag{293}$$

[5] A signature of a transaction can be valid not only with a recovery identifier but with some other numbers. See how the component T_w of a transaction is interpreted.

Appendix G. Fee Schedule

The fee schedule G is a tuple of 31 scalar values corresponding to the relative costs, in gas, of a number of abstract operations that a transaction may effect.

Name	Value	Description*
G_{zero}	0	Nothing paid for operations of the set W_{zero}.
G_{base}	2	Amount of gas to pay for operations of the set W_{base}.
$G_{verylow}$	3	Amount of gas to pay for operations of the set $W_{verylow}$.
G_{low}	5	Amount of gas to pay for operations of the set W_{low}.
G_{mid}	8	Amount of gas to pay for operations of the set W_{mid}.
G_{high}	10	Amount of gas to pay for operations of the set W_{high}.
$G_{extcode}$	700	Amount of gas to pay for operations of the set $W_{extcode}$.
$G_{balance}$	400	Amount of gas to pay for a BALANCE operation.
G_{sload}	200	Paid for a SLOAD operation.
$G_{jumpdest}$	1	Paid for a JUMPDEST operation.
G_{sset}	20000	Paid for an SSTORE operation when the storage value is set to non-zero from zero.
G_{sreset}	5000	Paid for an SSTORE operation when the storage value's zeroness remains unchanged or is set to zero.
R_{sclear}	15000	Refund given (added into refund counter) when the storage value is set to zero from non-zero.
$R_{selfdestruct}$	24000	Refund given (added into refund counter) for self-destructing an account.
$G_{selfdestruct}$	5000	Amount of gas to pay for a SELFDESTRUCT operation.
G_{create}	32000	Paid for a CREATE operation.
$G_{codedeposit}$	200	Paid per byte for a CREATE operation to succeed in placing code into state.
G_{call}	700	Paid for a CALL operation.
$G_{callvalue}$	9000	Paid for a non-zero value transfer as part of the CALL operation.
$G_{callstipend}$	2300	A stipend for the called contract subtracted from $G_{callvalue}$ for a non-zero value transfer.
$G_{newaccount}$	25000	Paid for a CALL or SELFDESTRUCT operation which creates an account.
G_{exp}	10	Partial payment for an EXP operation.
$G_{expbyte}$	50	Partial payment when multiplied by $\lceil \log_{256}(exponent) \rceil$ for the EXP operation.
G_{memory}	3	Paid for every additional word when expanding memory.
G_{txcreate}	32000	Paid by all contract-creating transactions after the *Homestead* transition.
$G_{txdatazero}$	4	Paid for every zero byte of data or code for a transaction.
$G_{txdatanonzero}$	68	Paid for every non-zero byte of data or code for a transaction.
$G_{transaction}$	21000	Paid for every transaction.
G_{log}	375	Partial payment for a LOG operation.
G_{logdata}	8	Paid for each byte in a LOG operation's data.
G_{logtopic}	375	Paid for each topic of a LOG operation.
G_{sha3}	30	Paid for each SHA3 operation.
$G_{sha3word}$	6	Paid for each word (rounded up) for input data to a SHA3 operation.
G_{copy}	3	Partial payment for *COPY operations, multiplied by words copied, rounded up.
$G_{blockhash}$	20	Payment for BLOCKHASH operation.
$G_{quaddivisor}$	100	The quadratic coefficient of the input sizes of the exponentiation-over-modulo precompiled contract.

Appendix H. Virtual Machine Specification

When interpreting 256-bit binary values as integers, the representation is big-endian.

When a 256-bit machine datum is converted to and from a 160-bit address or hash, the rightwards (low-order for BE) 20 bytes are used and the left most 12 are discarded or filled with zeroes, thus the integer values (when the bytes are interpreted as big-endian) are equivalent.

H.1. **Gas Cost.** The general gas cost function, C, is defined as:

(294)
$$C(\boldsymbol{\sigma}, \boldsymbol{\mu}, I) \equiv C_{mem}(\boldsymbol{\mu}'_{\mathrm{i}}) - C_{mem}(\boldsymbol{\mu}_{\mathrm{i}}) + \begin{cases} C_{\mathrm{SSTORE}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) & \text{if } w = \text{SSTORE} \\ G_{exp} & \text{if } w = \text{EXP} \wedge \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ G_{exp} + G_{expbyte} \times (1 + \lfloor \log_{256}(\boldsymbol{\mu}_{\mathbf{s}}[1]) \rfloor) & \text{if } w = \text{EXP} \wedge \boldsymbol{\mu}_{\mathbf{s}}[1] > 0 \\ G_{verylow} + G_{copy} \times \lceil \boldsymbol{\mu}_{\mathbf{s}}[2] \div 32 \rceil & \text{if } w = \text{CALLDATACOPY} \vee \\ & \quad \text{CODECOPY} \vee \text{RETURNDATACOPY} \\ G_{extcode} + G_{copy} \times \lceil \boldsymbol{\mu}_{\mathbf{s}}[3] \div 32 \rceil & \text{if } w = \text{EXTCODECOPY} \\ G_{log} + G_{logdata} \times \boldsymbol{\mu}_{\mathbf{s}}[1] & \text{if } w = \text{LOG0} \\ G_{log} + G_{logdata} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + G_{logtopic} & \text{if } w = \text{LOG1} \\ G_{log} + G_{logdata} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + 2G_{logtopic} & \text{if } w = \text{LOG2} \\ G_{log} + G_{logdata} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + 3G_{logtopic} & \text{if } w = \text{LOG3} \\ G_{log} + G_{logdata} \times \boldsymbol{\mu}_{\mathbf{s}}[1] + 4G_{logtopic} & \text{if } w = \text{LOG4} \\ C_{\mathrm{CALL}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) & \text{if } w = \text{CALL} \vee \text{CALLCODE} \vee \\ & \quad \text{DELEGATECALL} \\ C_{\mathrm{SELFDESTRUCT}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) & \text{if } w = \text{SELFDESTRUCT} \\ G_{create} & \text{if } w = \text{CREATE} \\ G_{sha3} + G_{sha3word} \lceil \mathbf{s}[1] \div 32 \rceil & \text{if } w = \text{SHA3} \\ G_{jumpdest} & \text{if } w = \text{JUMPDEST} \\ G_{sload} & \text{if } w = \text{SLOAD} \\ G_{zero} & \text{if } w \in W_{zero} \\ G_{base} & \text{if } w \in W_{base} \\ G_{verylow} & \text{if } w \in W_{verylow} \\ G_{\mathrm{low}} & \text{if } w \in W_{\mathrm{low}} \\ G_{mid} & \text{if } w \in W_{mid} \\ G_{\mathrm{high}} & \text{if } w \in W_{\mathrm{high}} \\ G_{extcode} & \text{if } w \in W_{extcode} \\ G_{balance} & \text{if } w = \text{BALANCE} \\ G_{blockhash} & \text{if } w = \text{BLOCKHASH} \end{cases}$$

(295)
$$w \equiv \begin{cases} I_{\mathbf{b}}[\boldsymbol{\mu}_{\mathrm{pc}}] & \text{if } \boldsymbol{\mu}_{\mathrm{pc}} < \|I_{\mathbf{b}}\| \\ \text{STOP} & \text{otherwise} \end{cases}$$

where:

(296)
$$C_{mem}(a) \equiv G_{memory} \cdot a + \left\lfloor \frac{a^2}{512} \right\rfloor$$

with C_{CALL}, $C_{\mathrm{SELFDESTRUCT}}$ and C_{SSTORE} as specified in the appropriate section below. We define the following subsets of instructions:

$W_{zero} = \{\text{STOP, RETURN, REVERT}\}$
$W_{base} = \{\text{ADDRESS, ORIGIN, CALLER, CALLVALUE, CALLDATASIZE, CODESIZE, GASPRICE, COINBASE, TIMESTAMP, NUMBER, DIFFICULTY, GASLIMIT, RETURNDATASIZE, POP, PC, MSIZE, GAS}\}$
$W_{verylow} = \{\text{ADD, SUB, NOT, LT, GT, SLT, SGT, EQ, ISZERO, AND, OR, XOR, BYTE, CALLDATALOAD, MLOAD, MSTORE, MSTORE8, PUSH*, DUP*, SWAP*}\}$
$W_{\mathrm{low}} = \{\text{MUL, DIV, SDIV, MOD, SMOD, SIGNEXTEND}\}$
$W_{mid} = \{\text{ADDMOD, MULMOD, JUMP}\}$
$W_{\mathrm{high}} = \{\text{JUMPI}\}$
$W_{extcode} = \{\text{EXTCODESIZE}\}$

Note the memory cost component, given as the product of G_{memory} and the maximum of 0 & the ceiling of the number of words in size that the memory must be over the current number of words, $\boldsymbol{\mu}_{\mathrm{i}}$ in order that all accesses reference valid memory whether for read or write. Such accesses must be for non-zero number of bytes.

Referencing a zero length range (e.g. by attempting to pass it as the input range to a CALL) does not require memory to be extended to the beginning of the range. $\boldsymbol{\mu}'_{\mathrm{i}}$ is defined as this new maximum number of words of active memory; special-cases are given where these two are not equal.

Note also that C_{mem} is the memory cost function (the expansion function being the difference between the cost before and after). It is a polynomial, with the higher-order coefficient divided and floored, and thus linear up to 724B of memory used, after which it costs substantially more.

While defining the instruction set, we defined the memory-expansion for range function, M, thus:

$$M(s, f, l) \equiv \begin{cases} s & \text{if} \quad l = 0 \\ \max(s, \lceil (f + l) \div 32 \rceil) & \text{otherwise} \end{cases} \tag{297}$$

Another useful function is "all but one 64th" function L defined as:

$$L(n) \equiv n - \lfloor n/64 \rfloor \tag{298}$$

H.2. **Instruction Set.** As previously specified in section 9, these definitions take place in the final context there. In particular we assume O is the EVM state-progression function and define the terms pertaining to the next cycle's state $(\boldsymbol{\sigma}', \boldsymbol{\mu}')$ such that:

$$O(\boldsymbol{\sigma}, \boldsymbol{\mu}, A, I) \equiv (\boldsymbol{\sigma}', \boldsymbol{\mu}', A', I) \quad \text{with exceptions, as noted} \tag{299}$$

Here given are the various exceptions to the state transition rules given in section 9 specified for each instruction, together with the additional instruction-specific definitions of J and C. For each instruction, also specified is α, the additional items placed on the stack and δ, the items removed from stack, as defined in section 9.

0s: Stop and Arithmetic Operations

All arithmetic is modulo 2^{256} unless otherwise noted. The zero-th power of zero 0^0 is defined to be one.

Value	Mnemonic	δ	α	Description
0x00	STOP	0	0	Halts execution.
0x01	ADD	2	1	Addition operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1]$
0x02	MUL	2	1	Multiplication operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0] \times \boldsymbol{\mu}_{\mathbf{s}}[1]$
0x03	SUB	2	1	Subtraction operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0] - \boldsymbol{\mu}_{\mathbf{s}}[1]$
0x04	DIV	2	1	Integer division operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ \lfloor \boldsymbol{\mu}_{\mathbf{s}}[0] \div \boldsymbol{\mu}_{\mathbf{s}}[1] \rfloor & \text{otherwise} \end{cases}$
0x05	SDIV	2	1	Signed integer division operation (truncated). $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{if } \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ -2^{255} & \text{if } \boldsymbol{\mu}_{\mathbf{s}}[0] = -2^{255} \wedge \boldsymbol{\mu}_{\mathbf{s}}[1] = -1 \\ \mathbf{sgn}(\boldsymbol{\mu}_{\mathbf{s}}[0] \div \boldsymbol{\mu}_{\mathbf{s}}[1]) \lfloor \lvert \boldsymbol{\mu}_{\mathbf{s}}[0] \div \boldsymbol{\mu}_{\mathbf{s}}[1] \rvert \rfloor & \text{otherwise} \end{cases}$ Where all values are treated as two's complement signed 256-bit integers. Note the overflow semantic when -2^{255} is negated.
0x06	MOD	2	1	Modulo remainder operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ \boldsymbol{\mu}_{\mathbf{s}}[0] \bmod \boldsymbol{\mu}_{\mathbf{s}}[1] & \text{otherwise} \end{cases}$
0x07	SMOD	2	1	Signed modulo remainder operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \\ \mathbf{sgn}(\boldsymbol{\mu}_{\mathbf{s}}[0])(\lvert \boldsymbol{\mu}_{\mathbf{s}}[0] \rvert \bmod \lvert \boldsymbol{\mu}_{\mathbf{s}}[1] \rvert) & \text{otherwise} \end{cases}$ Where all values are treated as two's complement signed 256-bit integers.
0x08	ADDMOD	3	1	Modulo addition operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] = 0 \\ (\boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1]) \bmod \boldsymbol{\mu}_{\mathbf{s}}[2] & \text{otherwise} \end{cases}$ All intermediate calculations of this operation are not subject to the 2^{256} modulo.
0x09	MULMOD	3	1	Modulo multiplication operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 0 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[2] = 0 \\ (\boldsymbol{\mu}_{\mathbf{s}}[0] \times \boldsymbol{\mu}_{\mathbf{s}}[1]) \bmod \boldsymbol{\mu}_{\mathbf{s}}[2] & \text{otherwise} \end{cases}$ All intermediate calculations of this operation are not subject to the 2^{256} modulo.
0x0a	EXP	2	1	Exponential operation. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]^{\boldsymbol{\mu}_{\mathbf{s}}[1]}$
0x0b	SIGNEXTEND	2	1	Extend length of two's complement signed integer. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_{\mathrm{i}} \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{s}}[1]_{\mathrm{t}} & \text{if} \quad i \leqslant t \quad \text{where } t = 256 - 8(\boldsymbol{\mu}_{\mathbf{s}}[0] + 1) \\ \boldsymbol{\mu}_{\mathbf{s}}[1]_{\mathrm{i}} & \text{otherwise} \end{cases}$

$\boldsymbol{\mu}_{\mathbf{s}}[x]_{\mathrm{i}}$ gives the ith bit (counting from zero) of $\boldsymbol{\mu}_{\mathbf{s}}[x]$

Value	Mnemonic	δ	α	Description
				10s: Comparison & Bitwise Logic Operations
0x10	LT	2	1	Less-than comparison. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] < \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{otherwise} \end{cases}$
0x11	GT	2	1	Greater-than comparison. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] > \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{otherwise} \end{cases}$
0x12	SLT	2	1	Signed less-than comparison. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] < \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{otherwise} \end{cases}$ Where all values are treated as two's complement signed 256-bit integers.
0x13	SGT	2	1	Signed greater-than comparison. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] > \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{otherwise} \end{cases}$ Where all values are treated as two's complement signed 256-bit integers.
0x14	EQ	2	1	Equality comparison. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] = \boldsymbol{\mu}_{\mathbf{s}}[1] \\ 0 & \text{otherwise} \end{cases}$
0x15	ISZERO	1	1	Simple not operator. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} 1 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[0] = 0 \\ 0 & \text{otherwise} \end{cases}$
0x16	AND	2	1	Bitwise AND operation. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]_i \wedge \boldsymbol{\mu}_{\mathbf{s}}[1]_i$
0x17	OR	2	1	Bitwise OR operation. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]_i \vee \boldsymbol{\mu}_{\mathbf{s}}[1]_i$
0x18	XOR	2	1	Bitwise XOR operation. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]_i \oplus \boldsymbol{\mu}_{\mathbf{s}}[1]_i$
0x19	NOT	1	1	Bitwise NOT operation. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \begin{cases} 1 & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[0]_i = 0 \\ 0 & \text{otherwise} \end{cases}$
0x1a	BYTE	2	1	Retrieve single byte from word. $\forall i \in [0..255] : \boldsymbol{\mu}'_{\mathbf{s}}[0]_i \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{s}}[1]_{(i+8\boldsymbol{\mu}_{\mathbf{s}}[0])} & \text{if} \quad i < 8 \wedge \boldsymbol{\mu}_{\mathbf{s}}[0] < 32 \\ 0 & \text{otherwise} \end{cases}$ For the Nth byte, we count from the left (i.e. N=0 would be the most significant in big endian).
				20s: SHA3
0x20	SHA3	2	1	Compute Keccak-256 hash. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \texttt{Keccak}(\boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \ldots (\boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1] - 1)])$ $\boldsymbol{\mu}'_{\mathbf{i}} \equiv M(\boldsymbol{\mu}_{\mathbf{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[1])$

30s: Environmental Information

Value	Mnemonic	δ	α	Description
0x30	ADDRESS	0	1	Get address of currently executing account. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{a}}$
0x31	BALANCE	1	1	Get balance of the given account. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \begin{cases} \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0]]_{\mathrm{b}} & \text{if} \quad \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \mod 2^{160}] \neq \varnothing \\ 0 & \text{otherwise} \end{cases}$
0x32	ORIGIN	0	1	Get execution origination address. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{o}}$ This is the sender of original transaction; it is never an account with non-empty associated code.
0x33	CALLER	0	1	Get caller address. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{s}}$ This is the address of the account that is directly responsible for this execution.
0x34	CALLVALUE	0	1	Get deposited value by the instruction/transaction responsible for this execution. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{v}}$
0x35	CALLDATALOAD	1	1	Get input data of current environment. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathbf{d}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + 31)] \quad \text{with} \quad I_{\mathbf{d}}[x] = 0 \quad \text{if} \quad x \geqslant \|I_{\mathbf{d}}\|$ This pertains to the input data passed with the message call instruction or transaction.
0x36	CALLDATASIZE	0	1	Get size of input data in current environment. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \|I_{\mathbf{d}}\|$ This pertains to the input data passed with the message call instruction or transaction.
0x37	CALLDATACOPY	3	0	Copy input data in current environment to memory. $\forall_{i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[2]-1\}} \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0]+i] \equiv \begin{cases} I_{\mathbf{d}}[\boldsymbol{\mu}_{\mathbf{s}}[1]+i] & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1]+i < \|I_{\mathbf{d}}\| \\ 0 & \text{otherwise} \end{cases}$ The additions in $\boldsymbol{\mu}_{\mathbf{s}}[1]+i$ are not subject to the 2^{256} modulo. $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[2])$ This pertains to the input data passed with the message call instruction or transaction.
0x38	CODESIZE	0	1	Get size of code running in current environment. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \|I_{\mathbf{b}}\|$
0x39	CODECOPY	3	0	Copy code running in current environment to memory. $\forall_{i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[2]-1\}} \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0]+i] \equiv \begin{cases} I_{\mathbf{b}}[\boldsymbol{\mu}_{\mathbf{s}}[1]+i] & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1]+i < \|I_{\mathbf{b}}\| \\ \text{STOP} & \text{otherwise} \end{cases}$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[2])$ The additions in $\boldsymbol{\mu}_{\mathbf{s}}[1]+i$ are not subject to the 2^{256} modulo.
0x3a	GASPRICE	0	1	Get price of gas in current environment. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{\mathrm{p}}$ This is gas price specified by the originating transaction.
0x3b	EXTCODESIZE	1	1	Get size of an account's code. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \|\boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \mod 2^{160}]_{\mathrm{c}}\|$
0x3c	EXTCODECOPY	4	0	Copy an account's code to memory. $\forall_{i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[3]-1\}} \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[1]+i] \equiv \begin{cases} \mathbf{c}[\boldsymbol{\mu}_{\mathbf{s}}[2]+i] & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[2]+i < \|\mathbf{c}\| \\ \text{STOP} & \text{otherwise} \end{cases}$ where $\mathbf{c} \equiv \boldsymbol{\sigma}[\boldsymbol{\mu}_{\mathbf{s}}[0] \mod 2^{160}]_{\mathrm{c}}$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[1], \boldsymbol{\mu}_{\mathbf{s}}[3])$ The additions in $\boldsymbol{\mu}_{\mathbf{s}}[2]+i$ are not subject to the 2^{256} modulo.
0x3d	RETURNDATASIZE	0	1	Get size of output data from the previous call from the current environment. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \|\boldsymbol{\mu}_{\mathbf{o}}\|$
0x3e	RETURNDATACOPY	3	0	Copy output data from the previous call to memory. $\forall_{i \in \{0 \dots \boldsymbol{\mu}_{\mathbf{s}}[2]-1\}} \boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0]+i] \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{o}}[\boldsymbol{\mu}_{\mathbf{s}}[1]+i] & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1]+i < \|\boldsymbol{\mu}_{\mathbf{o}}\| \\ 0 & \text{otherwise} \end{cases}$ The additions in $\boldsymbol{\mu}_{\mathbf{s}}[1]+i$ are not subject to the 2^{256} modulo. $\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[2])$

40s: Block Information

Value	Mnemonic	δ	α	Description
0x40	BLOCKHASH	1	1	Get the hash of one of the 256 most recent complete blocks. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv P(I_{\mathbf{H}_\mathrm{p}}, \boldsymbol{\mu}_{\mathbf{s}}[0], 0)$ where P is the hash of a block of a particular number, up to a maximum age. 0 is left on the stack if the looked for block number is greater than the current block number or more than 256 blocks behind the current block. $P(h,n,a) \equiv \begin{cases} 0 & \text{if} \quad n > H_\mathrm{i} \vee a = 256 \vee h = 0 \\ h & \text{if} \quad n = H_\mathrm{i} \\ P(H_\mathrm{p}, n, a+1) & \text{otherwise} \end{cases}$ and we assert the header H can be determined from its hash h unless as its hash is the parent h is zero.
0x41	COINBASE	0	1	Get the block's beneficiary address. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{H\mathrm{c}}$
0x42	TIMESTAMP	0	1	Get the block's timestamp. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{H\mathrm{s}}$
0x43	NUMBER	0	1	Get the block's number. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{H\mathrm{i}}$
0x44	DIFFICULTY	0	1	Get the block's difficulty. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{H\mathrm{d}}$
0x45	GASLIMIT	0	1	Get the block's gas limit. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv I_{H\mathrm{l}}$

50s: Stack, Memory, Storage and Flow Operations				
Value	**Mnemonic**	δ	α	**Description**
0x50	POP	1	0	Remove item from stack.
0x51	MLOAD	1	1	Load word from memory. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + 31)]$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv \max(\boldsymbol{\mu}_{\mathrm{i}}, \lceil (\boldsymbol{\mu}_{\mathbf{s}}[0] + 32) \div 32 \rceil)$ The addition in the calculation of $\boldsymbol{\mu}'_{\mathrm{i}}$ is not subject to the 2^{256} modulo.
0x52	MSTORE	2	0	Save word to memory. $\boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + 31)] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv \max(\boldsymbol{\mu}_{\mathrm{i}}, \lceil (\boldsymbol{\mu}_{\mathbf{s}}[0] + 32) \div 32 \rceil)$ The addition in the calculation of $\boldsymbol{\mu}'_{\mathrm{i}}$ is not subject to the 2^{256} modulo.
0x53	MSTORE8	2	0	Save byte to memory. $\boldsymbol{\mu}'_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0]] \equiv (\boldsymbol{\mu}_{\mathbf{s}}[1] \bmod 256)$ $\boldsymbol{\mu}'_{\mathrm{i}} \equiv \max(\boldsymbol{\mu}_{\mathrm{i}}, \lceil (\boldsymbol{\mu}_{\mathbf{s}}[0] + 1) \div 32 \rceil)$ The addition in the calculation of $\boldsymbol{\mu}'_{\mathrm{i}}$ is not subject to the 2^{256} modulo.
0x54	SLOAD	1	1	Load word from storage. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]]$
0x55	SSTORE	2	0	Save word to storage. $\boldsymbol{\sigma}'[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$ $C_{\mathrm{SSTORE}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv \begin{cases} G_{sset} & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] \neq 0 \ \wedge \ \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]] = 0 \\ G_{sreset} & \text{otherwise} \end{cases}$ $A'_r \equiv A_r + \begin{cases} R_{sclear} & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] = 0 \ \wedge \ \boldsymbol{\sigma}[I_{\mathrm{a}}]_{\mathbf{s}}[\boldsymbol{\mu}_{\mathbf{s}}[0]] \neq 0 \\ 0 & \text{otherwise} \end{cases}$
0x56	JUMP	1	0	Alter the program counter. $J_{\mathrm{JUMP}}(\boldsymbol{\mu}) \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$ This has the effect of writing said value to $\boldsymbol{\mu}_{\mathrm{pc}}$. See section 9.
0x57	JUMPI	2	0	Conditionally alter the program counter. $J_{\mathrm{JUMPI}}(\boldsymbol{\mu}) \equiv \begin{cases} \boldsymbol{\mu}_{\mathbf{s}}[0] & \text{if} \quad \boldsymbol{\mu}_{\mathbf{s}}[1] \neq 0 \\ \boldsymbol{\mu}_{\mathrm{pc}} + 1 & \text{otherwise} \end{cases}$ This has the effect of writing said value to $\boldsymbol{\mu}_{\mathrm{pc}}$. See section 9.
0x58	PC	0	1	Get the value of the program counter *prior* to the increment corresponding to this instruction. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathrm{pc}}$
0x59	MSIZE	0	1	Get the size of active memory in bytes. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv 32\boldsymbol{\mu}_i$
0x5a	GAS	0	1	Get the amount of available gas, including the corresponding reduction for the cost of this instruction. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_g$
0x5b	JUMPDEST	0	0	Mark a valid destination for jumps. This operation has no effect on machine state during execution.

60s & 70s: Push Operations

Value	Mnemonic	δ	α	Description
0x60	PUSH1	0	1	Place 1 byte item on stack. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv c(\boldsymbol{\mu}_{\mathrm{pc}} + 1)$ where $c(x) \equiv \begin{cases} I_{\mathbf{b}}[x] & \text{if } x < \|I_{\mathbf{b}}\| \\ 0 & \text{otherwise} \end{cases}$ The bytes are read in line from the program code's bytes array. The function c ensures the bytes default to zero if they extend past the limits. The byte is right-aligned (takes the lowest significant place in big endian).
0x61	PUSH2	0	1	Place 2-byte item on stack. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{c}\big((\boldsymbol{\mu}_{\mathrm{pc}} + 1) \dots (\boldsymbol{\mu}_{\mathrm{pc}} + 2)\big)$ with $\boldsymbol{c}(\boldsymbol{x}) \equiv (c(\boldsymbol{x}_0), ..., c(\boldsymbol{x}_{\|x\|-1}))$ with c as defined as above. The bytes are right-aligned (takes the lowest significant place in big endian).
⋮	⋮	⋮	⋮	⋮
0x7f	PUSH32	0	1	Place 32-byte (full word) item on stack. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{c}\big((\boldsymbol{\mu}_{\mathrm{pc}} + 1) \dots (\boldsymbol{\mu}_{\mathrm{pc}} + 32)\big)$ where $\boldsymbol{c}$ is defined as above. The bytes are right-aligned (takes the lowest significant place in big endian).

80s: Duplication Operations

Value	Mnemonic	δ	α	Description
0x80	DUP1	1	2	Duplicate 1st stack item. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$
0x81	DUP2	2	3	Duplicate 2nd stack item. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$
⋮	⋮	⋮	⋮	⋮
0x8f	DUP16	16	17	Duplicate 16th stack item. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[15]$

90s: Exchange Operations

Value	Mnemonic	δ	α	Description
0x90	SWAP1	2	2	Exchange 1st and 2nd stack items. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[1]$ $\boldsymbol{\mu}'_{\mathbf{s}}[1] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$
0x91	SWAP2	3	3	Exchange 1st and 3rd stack items. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[2]$ $\boldsymbol{\mu}'_{\mathbf{s}}[2] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$
⋮	⋮	⋮	⋮	⋮
0x9f	SWAP16	17	17	Exchange 1st and 17th stack items. $\boldsymbol{\mu}'_{\mathbf{s}}[0] \equiv \boldsymbol{\mu}_{\mathbf{s}}[16]$ $\boldsymbol{\mu}'_{\mathbf{s}}[16] \equiv \boldsymbol{\mu}_{\mathbf{s}}[0]$

a0s: Logging Operations

For all logging operations, the state change is to append an additional log entry on to the substate's log series:

$$\text{A } 1A'_{\mathbf{l}} \equiv A_{\mathbf{l}} \cdot (I_{\mathrm{a}}, \mathbf{t}, \boldsymbol{\mu}_{\mathbf{m}}[\boldsymbol{\mu}_{\mathbf{s}}[0] \dots (\boldsymbol{\mu}_{\mathbf{s}}[0] + \boldsymbol{\mu}_{\mathbf{s}}[1] - 1)])$$

and to update the memory consumption counter:
$\boldsymbol{\mu}'_{\mathrm{i}} \equiv M(\boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\mu}_{\mathbf{s}}[0], \boldsymbol{\mu}_{\mathbf{s}}[1])$
The entry's topic series, $\mathbf{t}$, differs accordingly:

Value	Mnemonic	δ	α	Description
0xa0	LOG0	2	0	Append log record with no topics. $\mathbf{t} \equiv ()$
0xa1	LOG1	3	0	Append log record with one topic. $\mathbf{t} \equiv (\boldsymbol{\mu}_{\mathbf{s}}[2])$
⋮	⋮	⋮	⋮	⋮
0xa4	LOG4	6	0	Append log record with four topics. $\mathbf{t} \equiv (\boldsymbol{\mu}_{\mathbf{s}}[2], \boldsymbol{\mu}_{\mathbf{s}}[3], \boldsymbol{\mu}_{\mathbf{s}}[4], \boldsymbol{\mu}_{\mathbf{s}}[5])$

f0s: System operations

Value	Mnemonic	δ	α	Description
0xf0	CREATE	3	1	Create a new account with associated code. $\mathbf{i} \equiv \boldsymbol{\mu}_\mathbf{m}[\boldsymbol{\mu}_\mathbf{s}[1] \dots (\boldsymbol{\mu}_\mathbf{s}[1] + \boldsymbol{\mu}_\mathbf{s}[2] - 1)]$ $(\boldsymbol{\sigma}', \boldsymbol{\mu}'_\mathrm{g}, A^+, \mathbf{o}) \equiv \begin{cases} \Lambda(\boldsymbol{\sigma}^*, I_\mathrm{a}, I_\mathrm{o}, L(\boldsymbol{\mu}_\mathrm{g}), I_\mathrm{p}, \boldsymbol{\mu}_\mathbf{s}[0], \mathbf{i}, I_\mathrm{e}+1, I_\mathrm{w}) & \text{if } \boldsymbol{\mu}_\mathbf{s}[0] \leqslant \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} \\ & \quad \wedge\, I_\mathrm{e} < 1024 \\ (\boldsymbol{\sigma}, \boldsymbol{\mu}_\mathrm{g}, \varnothing) & \text{otherwise} \end{cases}$ $\boldsymbol{\sigma}^* \equiv \boldsymbol{\sigma}$ except $\boldsymbol{\sigma}^*[I_\mathrm{a}]_\mathrm{n} = \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{n} + 1$ $A' \equiv A \Cup A^+$ which abbreviates: $A'_\mathbf{s} \equiv A_\mathbf{s} \cup A^+_\mathbf{s} \quad \wedge \quad A'_\mathbf{l} \equiv A_\mathbf{l} \cdot A^+_\mathbf{l} \quad \wedge$ $A'_\mathbf{t} \equiv A_\mathbf{t} \cup A^+_\mathbf{t} \wedge \quad A'_\mathbf{r} \equiv A_\mathbf{r} + A^+_\mathbf{r}$ $\boldsymbol{\mu}'_\mathbf{s}[0] \equiv x$ where $x = 0$ if the code execution for this operation failed due to an exceptional halting (or for a REVERT) $\boldsymbol{\sigma}' = \varnothing$, or $I_\mathrm{e} = 1024$ (the maximum call depth limit is reached) or $\boldsymbol{\mu}_\mathbf{s}[0] > \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b}$ (balance of the caller is too low to fulfil the value transfer); and otherwise $x = A(I_\mathrm{a}, \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{n})$, the address of the newly created account. $\boldsymbol{\mu}'_\mathrm{i} \equiv M(\boldsymbol{\mu}_\mathrm{i}, \boldsymbol{\mu}_\mathbf{s}[1], \boldsymbol{\mu}_\mathbf{s}[2])$ $\boldsymbol{\mu}'_\mathbf{o} \equiv ()$ Thus the operand order is: value, input offset, input size.
0xf1	CALL	7	1	Message-call into an account. $\mathbf{i} \equiv \boldsymbol{\mu}_\mathbf{m}[\boldsymbol{\mu}_\mathbf{s}[3] \dots (\boldsymbol{\mu}_\mathbf{s}[3] + \boldsymbol{\mu}_\mathbf{s}[4] - 1)]$ $(\boldsymbol{\sigma}', g', A^+, \mathbf{o}) \equiv \begin{cases} \Theta(\boldsymbol{\sigma}, I_\mathrm{a}, I_\mathrm{o}, t, t, C_\mathrm{CALLGAS}(\boldsymbol{\mu}), & \text{if } \boldsymbol{\mu}_\mathbf{s}[2] \leqslant \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} \wedge \\ \quad I_\mathrm{p}, \boldsymbol{\mu}_\mathbf{s}[2], \boldsymbol{\mu}_\mathbf{s}[2], \mathbf{i}, I_\mathrm{e}+1, I_\mathrm{w}) & \quad I_\mathrm{e} < 1024 \\ (\boldsymbol{\sigma}, g, \varnothing, ()) & \text{otherwise} \end{cases}$ $n \equiv \min(\{\boldsymbol{\mu}_\mathbf{s}[6], |\mathbf{o}|\})$ $\boldsymbol{\mu}'_\mathbf{m}[\boldsymbol{\mu}_\mathbf{s}[5] \dots (\boldsymbol{\mu}_\mathbf{s}[5] + n - 1)] = \mathbf{o}[0 \dots (n-1)]$ $\boldsymbol{\mu}'_\mathbf{o} = \mathbf{o}$ $\boldsymbol{\mu}'_\mathrm{g} \equiv \boldsymbol{\mu}_\mathrm{g} + g'$ $\boldsymbol{\mu}'_\mathbf{s}[0] \equiv x$ $A' \equiv A \Cup A^+$ $t \equiv \boldsymbol{\mu}_\mathbf{s}[1] \mod 2^{160}$ where $x = 0$ if the code execution for this operation failed due to an exceptional halting (or for a REVERT) $\boldsymbol{\sigma}' = \varnothing$ or if $\boldsymbol{\mu}_\mathbf{s}[2] > \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b}$ (not enough funds) or $I_\mathrm{e} = 1024$ (call depth limit reached); $x = 1$ otherwise. $\boldsymbol{\mu}'_\mathrm{i} \equiv M(M(\boldsymbol{\mu}_\mathrm{i}, \boldsymbol{\mu}_\mathbf{s}[3], \boldsymbol{\mu}_\mathbf{s}[4]), \boldsymbol{\mu}_\mathbf{s}[5], \boldsymbol{\mu}_\mathbf{s}[6])$ Thus the operand order is: gas, to, value, in offset, in size, out offset, out size. $C_\mathrm{CALL}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv C_\mathrm{GASCAP}(\boldsymbol{\sigma}, \boldsymbol{\mu}) + C_\mathrm{EXTRA}(\boldsymbol{\sigma}, \boldsymbol{\mu})$ $C_\mathrm{CALLGAS}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv \begin{cases} C_\mathrm{GASCAP}(\boldsymbol{\sigma}, \boldsymbol{\mu}) + G_{callstipend} & \text{if } \boldsymbol{\mu}_\mathbf{s}[2] \neq 0 \\ C_\mathrm{GASCAP}(\boldsymbol{\sigma}, \boldsymbol{\mu}) & \text{otherwise} \end{cases}$ $C_\mathrm{GASCAP}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv \begin{cases} \min\{L(\boldsymbol{\mu}_\mathrm{g} - C_\mathrm{EXTRA}(\boldsymbol{\sigma}, \boldsymbol{\mu})), \boldsymbol{\mu}_\mathbf{s}[0]\} & \text{if } \boldsymbol{\mu}_\mathrm{g} \geq C_\mathrm{EXTRA}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \\ \boldsymbol{\mu}_\mathbf{s}[0] & \text{otherwise} \end{cases}$ $C_\mathrm{EXTRA}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv G_{call} + C_\mathrm{XFER}(\boldsymbol{\mu}) + C_\mathrm{NEW}(\boldsymbol{\sigma}, \boldsymbol{\mu})$ $C_\mathrm{XFER}(\boldsymbol{\mu}) \equiv \begin{cases} G_{callvalue} & \text{if } \boldsymbol{\mu}_\mathbf{s}[2] \neq 0 \\ 0 & \text{otherwise} \end{cases}$ $C_\mathrm{NEW}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv \begin{cases} G_{newaccount} & \text{if } \texttt{DEAD}(\boldsymbol{\sigma}, \boldsymbol{\mu}_\mathbf{s}[1] \mod 2^{160}) \wedge \boldsymbol{\mu}_\mathbf{s}[2] \neq 0 \\ 0 & \text{otherwise} \end{cases}$
0xf2	CALLCODE	7	1	Message-call into this account with an alternative account's code. Exactly equivalent to CALL except: $(\boldsymbol{\sigma}', g', A^+, \mathbf{o}) \equiv \begin{cases} \Theta(\boldsymbol{\sigma}^*, I_\mathrm{a}, I_\mathrm{o}, I_\mathrm{a}, t, C_\mathrm{CALLGAS}(\boldsymbol{\mu}), & \text{if } \boldsymbol{\mu}_\mathbf{s}[2] \leqslant \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} \wedge \\ \quad I_\mathrm{p}, \boldsymbol{\mu}_\mathbf{s}[2], \boldsymbol{\mu}_\mathbf{s}[2], \mathbf{i}, I_\mathrm{e}+1, I_\mathrm{w}) & \quad I_\mathrm{e} < 1024 \\ (\boldsymbol{\sigma}, g, \varnothing, ()) & \text{otherwise} \end{cases}$ Note the change in the fourth parameter to the call Θ from the 2nd stack value $\boldsymbol{\mu}_\mathbf{s}[1]$ (as in CALL) to the present address I_a. This means that the recipient is in fact the same account as at present, simply that the code is overwritten.
0xf3	RETURN	2	0	Halt execution returning output data. $H_\mathrm{RETURN}(\boldsymbol{\mu}) \equiv \boldsymbol{\mu}_\mathbf{m}[\boldsymbol{\mu}_\mathbf{s}[0] \dots (\boldsymbol{\mu}_\mathbf{s}[0] + \boldsymbol{\mu}_\mathbf{s}[1] - 1)]$ This has the effect of halting the execution at this point with output defined. See section 9. $\boldsymbol{\mu}'_\mathrm{i} \equiv M(\boldsymbol{\mu}_\mathrm{i}, \boldsymbol{\mu}_\mathbf{s}[0], \boldsymbol{\mu}_\mathbf{s}[1])$

0xf4	DELEGATECALL	6	1	Message-call into this account with an alternative account's code, but persisting the current values for *sender* and *value*. Compared with CALL, DELEGATECALL takes one fewer arguments. The omitted argument is $\boldsymbol{\mu}_\mathbf{s}[2]$. As a result, $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$, $\boldsymbol{\mu}_\mathbf{s}[5]$ and $\boldsymbol{\mu}_\mathbf{s}[6]$ in the definition of CALL should respectively be replaced with $\boldsymbol{\mu}_\mathbf{s}[2]$, $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$ and $\boldsymbol{\mu}_\mathbf{s}[5]$. Otherwise it is equivalent to CALL except: $(\boldsymbol{\sigma}', g', A^+, \mathbf{o}) \equiv \begin{cases} \Theta(\boldsymbol{\sigma}^*, I_\mathrm{s}, I_\mathrm{o}, I_\mathrm{a}, t, \\ \quad \boldsymbol{\mu}_\mathbf{s}[0], I_\mathrm{p}, 0, I_\mathrm{v}, \mathbf{i}, I_\mathrm{e}+1, I_\mathrm{w}) & \text{if} \quad I_\mathrm{v} \leqslant \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} \wedge \\ I_\mathrm{e} < 1024 \\ (\boldsymbol{\sigma}, g, \varnothing, ()) & \text{otherwise} \end{cases}$ Note the changes (in addition to that of the fourth parameter) to the second and ninth parameters to the call Θ. This means that the recipient is in fact the same account as at present, simply that the code is overwritten *and* the context is almost entirely identical.
0xfa	STATICCALL	6	1	Static message-call into an account. Exactly equivalent to CALL except: The argument $\boldsymbol{\mu}_\mathbf{s}[2]$ is replaced with 0. The deeper argument $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$, $\boldsymbol{\mu}_\mathbf{s}[5]$ and $\boldsymbol{\mu}_\mathbf{s}[6]$ are respectively replaced with $\boldsymbol{\mu}_\mathbf{s}[2]$, $\boldsymbol{\mu}_\mathbf{s}[3]$, $\boldsymbol{\mu}_\mathbf{s}[4]$ and $\boldsymbol{\mu}_\mathbf{s}[5]$. The last argument of Θ is $\bot$.
0xfd	REVERT	2	0	Halt execution reverting state changes but returning data and remaining gas. The effect of this operation is described in (131). For the gas calculation, we use the memory expansion function, $\boldsymbol{\mu}'_\mathrm{i} \equiv M(\boldsymbol{\mu}_\mathrm{i}, \boldsymbol{\mu}_\mathbf{s}[0], \boldsymbol{\mu}_\mathbf{s}[1])$
0xfe	INVALID	$\varnothing$	$\varnothing$	Designated invalid instruction.
0xff	SELFDESTRUCT	1	0	Halt execution and register account for later deletion. $A'_\mathbf{s} \equiv A_\mathbf{s} \cup \{I_\mathrm{a}\}$ $\boldsymbol{\sigma}'[r] \equiv \begin{cases} \varnothing & \text{if } \boldsymbol{\sigma}[r] = \varnothing \ \wedge \ \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} = 0 \\ (\boldsymbol{\sigma}[r]_\mathrm{n}, \boldsymbol{\sigma}[r]_\mathrm{b} + \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b}, \boldsymbol{\sigma}[r]_\mathbf{s}, \boldsymbol{\sigma}[r]_\mathrm{c}) & \text{if } r \neq I_\mathrm{a} \\ (\boldsymbol{\sigma}[r]_\mathrm{n}, 0, \boldsymbol{\sigma}[r]_\mathbf{s}, \boldsymbol{\sigma}[r]_\mathrm{c}) & \text{otherwise} \end{cases}$ where $r = \boldsymbol{\mu}_\mathbf{s}[0] \bmod 2^{160}$ $\boldsymbol{\sigma}'[I_\mathrm{a}]_\mathrm{b} = 0$ $C_{\text{SELFDESTRUCT}}(\boldsymbol{\sigma}, \boldsymbol{\mu}) \equiv G_{selfdestruct} + \begin{cases} G_{newaccount} & \text{if} \quad n \\ 0 & \text{otherwise} \end{cases}$ $n \equiv \texttt{DEAD}(\boldsymbol{\sigma}, \boldsymbol{\mu}_\mathbf{s}[0] \mod 2^{160}) \wedge \boldsymbol{\sigma}[I_\mathrm{a}]_\mathrm{b} \neq 0$

Appendix I. Genesis Block

The genesis block is 15 items, and is specified thus:

$$(300) \qquad \big((0_{256}, \texttt{KEC}\big(\texttt{RLP}\big(()\big)\big), 0_{160}, \mathit{stateRoot}, 0, 0, 0_{2048}, 2^{17}, 0, 0, 3141592, \mathit{time}, 0, 0_{256}, \texttt{KEC}\big((42)\big)\big), (), ()\big)$$

Where 0_{256} refers to the parent hash, a 256-bit hash which is all zeroes; 0_{160} refers to the beneficiary address, a 160-bit hash which is all zeroes; 0_{2048} refers to the log bloom, 2048-bit of all zeros; 2^{17} refers to the difficulty; the transaction trie root, receipt trie root, gas used, block number and extradata are both 0, being equivalent to the empty byte array. The sequences of both ommers and transactions are empty and represented by (). $\texttt{KEC}\big((42)\big)$ refers to the Keccak hash of a byte array of length one whose first and only byte is of value 42, used for the nonce. $\texttt{KEC}\big(\texttt{RLP}\big(()\big)\big)$ value refers to the hash of the ommer list in RLP, both empty lists.

The proof-of-concept series include a development premine, making the state root hash some value *stateRoot*. Also *time* will be set to the initial timestamp of the genesis block. The latest documentation should be consulted for those values.

Appendix J. Ethash

J.1. **Definitions.** We employ the following definitions:

Name	Value	Description
$J_{wordbytes}$	4	Bytes in word.
$J_{datasetinit}$	2^{30}	Bytes in dataset at genesis.
$J_{datasetgrowth}$	2^{23}	Dataset growth per epoch.
$J_{cacheinit}$	2^{24}	Bytes in cache at genesis.
$J_{cachegrowth}$	2^{17}	Cache growth per epoch.
J_{epoch}	30000	Blocks per epoch.
$J_{mixbytes}$	128	mix length in bytes.
$J_{\text{hashbytes}}$	64	Hash length in bytes.
J_{parents}	256	Number of parents of each dataset element.
$J_{cacherounds}$	3	Number of rounds in cache production.
J_{accesses}	64	Number of accesses in hashimoto loop.

J.2. **Size of dataset and cache.** The size for Ethash's cache $\mathbf{c} \in \mathbb{B}$ and dataset $\mathbf{d} \in \mathbb{B}$ depend on the epoch, which in turn depends on the block number.

$$E_{epoch}(H_{\text{i}}) = \left\lfloor \frac{H_{\text{i}}}{J_{epoch}} \right\rfloor \tag{301}$$

The size of the dataset growth by $J_{datasetgrowth}$ bytes, and the size of the cache by $J_{cachegrowth}$ bytes, every epoch. In order to avoid regularity leading to cyclic behavior, the size must be a prime number. Therefore the size is reduced by a multiple of $J_{mixbytes}$, for the dataset, and $J_{\text{hashbytes}}$ for the cache. Let $d_{size} = \|\mathbf{d}\|$ be the size of the dataset. Which is calculated using

$$d_{size} = E_{\text{prime}}(J_{datasetinit} + J_{datasetgrowth} \cdot E_{epoch} - J_{mixbytes}, J_{mixbytes}) \tag{302}$$

The size of the cache, c_{size}, is calculated using

$$c_{size} = E_{\text{prime}}(J_{cacheinit} + J_{cachegrowth} \cdot E_{epoch} - J_{\text{hashbytes}}, J_{\text{hashbytes}}) \tag{303}$$

$$E_{\text{prime}}(x, y) = \begin{cases} x & \text{if} \quad x/y \in \mathbb{N} \\ E_{\text{prime}}(x - 2 \cdot y, y) & \text{otherwise} \end{cases} \tag{304}$$

J.3. **Dataset generation.** In order to generate the dataset we need the cache $\mathbf{c}$, which is an array of bytes. It depends on the cache size c_{size} and the seed hash $\mathbf{s} \in \mathbb{B}_{32}$.

J.3.1. *Seed hash.* The seed hash is different for every epoch. For the first epoch it is the Keccak-256 hash of a series of 32 bytes of zeros. For every other epoch it is always the Keccak-256 hash of the previous seed hash:

$$\mathbf{s} = C_{seedhash}(H_{\text{i}}) \tag{305}$$

$$C_{seedhash}(H_{\text{i}}) = \begin{cases} \mathbf{0}_{32} & \text{if} \quad E_{epoch}(H_{\text{i}}) = 0 \\ \texttt{KEC}(C_{seedhash}(H_{\text{i}} - J_{epoch})) & \text{otherwise} \end{cases} \tag{306}$$

With $\mathbf{0}_{32}$ being 32 bytes of zeros.

J.3.2. *Cache.* The cache production process involves using the seed hash to first sequentially filling up c_{size} bytes of memory, then performing $J_{cacherounds}$ passes of the RandMemoHash algorithm created by Lerner [2014]. The initial cache $\mathbf{c}'$, being an array of arrays of single bytes, will be constructed as follows.

We define the array $\mathbf{c}_i$, consisting of 64 single bytes, as the ith element of the initial cache:

$$\mathbf{c}_i = \begin{cases} \texttt{KEC512}(\mathbf{s}) & \text{if} \quad i = 0 \\ \texttt{KEC512}(\mathbf{c}_{i-1}) & \text{otherwise} \end{cases} \tag{307}$$

Therefore $\mathbf{c}'$ can be defined as

$$\mathbf{c}'[i] = \mathbf{c}_i \quad \forall \quad i < n \tag{308}$$

$$n = \left\lfloor \frac{c_{size}}{J_{\text{hashbytes}}} \right\rfloor \tag{309}$$

The cache is calculated by performing $J_{cacherounds}$ rounds of the RandMemoHash algorithm to the initial cache $\mathbf{c}'$:

$$\mathbf{c} = E_{cacherounds}(\mathbf{c}', J_{cacherounds}) \tag{310}$$

$$E_{cacherounds}(\mathbf{x}, y) = \begin{cases} \mathbf{x} & \text{if} \quad y = 0 \\ E_{\text{RMH}}(\mathbf{x}) & \text{if} \quad y = 1 \\ E_{cacherounds}(E_{\text{RMH}}(\mathbf{x}), y - 1) & \text{otherwise} \end{cases} \tag{311}$$

Where a single round modifies each subset of the cache as follows:

$$E_{\text{RMH}}(\mathbf{x}) = \big(E_{rmh}(\mathbf{x}, 0), E_{rmh}(\mathbf{x}, 1), ..., E_{rmh}(\mathbf{x}, n - 1)\big) \tag{312}$$

$$E_{rmh}(\mathbf{x}, i) = \texttt{KEC512}(\mathbf{x}'[(i - 1 + n) \mod n] \oplus \mathbf{x}'[\mathbf{x}'[i][0] \mod n]) \quad \text{with} \quad \mathbf{x}' = \mathbf{x} \quad \text{except} \quad \mathbf{x}'[j] = E_{rmh}(\mathbf{x}, j) \quad \forall \quad j < i \tag{313}$$

J.3.3. *Full dataset calculation.* Essentially, we combine data from J_{parents} pseudorandomly selected cache nodes, and hash that to compute the dataset. The entire dataset is then generated by a number of items, each $J_{\text{hashbytes}}$ bytes in size:

$$\mathbf{d}[i] = E_{datasetitem}(\mathbf{c}, i) \quad \forall \quad i < \left\lfloor \frac{d_{size}}{J_{\text{hashbytes}}} \right\rfloor \tag{314}$$

In order to calculate the single item we use an algorithm inspired by the FNV hash (Glenn Fowler [1991]) in some cases as a non-associative substitute for XOR.

$$E_{\text{FNV}}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot (0\text{x}01000193 \oplus \mathbf{y})) \mod 2^{32} \tag{315}$$

The single item of the dataset can now be calculated as:

$$E_{datasetitem}(\mathbf{c}, i) = E_{\text{parents}}(\mathbf{c}, i, -1, \varnothing) \tag{316}$$

$$E_{\text{parents}}(\mathbf{c}, i, p, \mathbf{m}) = \begin{cases} E_{\text{parents}}(\mathbf{c}, i, p + 1, E_{mix}(\mathbf{m}, \mathbf{c}, i, p + 1)) & \text{if} \quad p < J_{\text{parents}} - 2 \\ E_{mix}(\mathbf{m}, \mathbf{c}, i, p + 1) & \text{otherwise} \end{cases} \tag{317}$$

$$E_{mix}(\mathbf{m}, \mathbf{c}, i, p) = \begin{cases} \texttt{KEC512}(\mathbf{c}[i \mod c_{size}] \oplus i) & \text{if} \quad p = 0 \\ E_{\text{FNV}}\big(\mathbf{m}, \mathbf{c}[E_{\text{FNV}}(i \oplus p, \mathbf{m}[p \mod \lfloor J_{\text{hashbytes}}/J_{wordbytes} \rfloor]) \mod c_{size}]\big) & \text{otherwise} \end{cases} \tag{318}$$

J.4. **Proof-of-work function.** Essentially, we maintain a "mix" $J_{mixbytes}$ bytes wide, and repeatedly sequentially fetch $J_{mixbytes}$ bytes from the full dataset and use the E_{FNV} function to combine it with the mix. $J_{mixbytes}$ bytes of sequential access are used so that each round of the algorithm always fetches a full page from RAM, minimizing translation lookaside buffer misses which ASICs would theoretically be able to avoid.

If the output of this algorithm is below the desired target, then the nonce is valid. Note that the extra application of KEC at the end ensures that there exists an intermediate nonce which can be provided to prove that at least a small amount of work was done; this quick outer PoW verification can be used for anti-DDoS purposes. It also serves to provide statistical assurance that the result is an unbiased, 256 bit number.

The PoW-function returns an array with the compressed mix as its first item and the Keccak-256 hash of the concatenation of the compressed mix with the seed hash as the second item:

$$\texttt{PoW}(H_{\not H}, H_{\text{n}}, \mathbf{d}) = \{\mathbf{m}_{\text{c}}(\texttt{KEC}(\texttt{RLP}(L_H(H_{\not H}))), H_{\text{n}}, \mathbf{d}), \texttt{KEC}(\mathbf{s}_{\text{h}}(\texttt{KEC}(\texttt{RLP}(L_H(H_{\not H}))), H_{\text{n}}) + \mathbf{m}_{\text{c}}(\texttt{KEC}(\texttt{RLP}(L_H(H_{\not H}))), H_{\text{n}}, \mathbf{d}))\} \tag{319}$$

With $H_{\not H}$ being the hash of the header without the nonce. The compressed mix $\mathbf{m}_{\text{c}}$ is obtained as follows:

$$\mathbf{m}_{\text{c}}(\mathbf{h}, \mathbf{n}, \mathbf{d}) = E_{compress}(E_{\text{accesses}}(\mathbf{d}, \sum_{i=0}^{n_{mix}} \mathbf{s}_{\text{h}}(\mathbf{h}, \mathbf{n}), \mathbf{s}_{\text{h}}(\mathbf{h}, \mathbf{n}), -1), -4) \tag{320}$$

The seed hash being:

$$\mathbf{s}_{\text{h}}(\mathbf{h}, \mathbf{n}) = \texttt{KEC512}(\mathbf{h} + E_{revert}(\mathbf{n})) \tag{321}$$

$E_{revert}(\mathbf{n})$ returns the reverted bytes sequence of the nonce $\mathbf{n}$:

$$E_{revert}(\mathbf{n})[i] = \mathbf{n}[\|\mathbf{n}\| - i] \tag{322}$$

We note that the "+"-operator between two byte sequences results in the concatenation of both sequences.

The dataset $\mathbf{d}$ is obtained as described in section J.3.3.

The number of replicated sequences in the mix is:

$$n_{mix} = \left\lfloor \frac{J_{mixbytes}}{J_{\text{hashbytes}}} \right\rfloor \tag{323}$$

In order to add random dataset nodes to the mix, the E_{accesses} function is used:

$$E_{\text{accesses}}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i) = \begin{cases} E_{mixdataset}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i) & \text{if} \quad i = J_{\text{accesses}} - 2 \\ E_{\text{accesses}}(E_{mixdataset}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i), \mathbf{s}, i + 1) & \text{otherwise} \end{cases} \tag{324}$$

$$E_{mixdataset}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i) = E_{\text{FNV}}(\mathbf{m}, E_{newdata}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i) \tag{325}$$

$E_{newdata}$ returns an array with n_{mix} elements:

$$E_{newdata}(\mathbf{d}, \mathbf{m}, \mathbf{s}, i)[j] = \mathbf{d}[E_{\text{FNV}}(i \oplus \mathbf{s}[0], \mathbf{m}[i \mod \left\lfloor \frac{J_{mixbytes}}{J_{wordbytes}} \right\rfloor]) \mod \left\lfloor \frac{d_{size}/J_{\text{hashbytes}}}{n_{mix}} \right\rfloor \cdot n_{mix} + j] \quad \forall \quad j < n_{mix} \tag{326}$$

The mix is compressed as follows:

$$E_{compress}(\mathbf{m}, i) = \begin{cases} \mathbf{m} & \text{if} \quad i \geqslant \|\mathbf{m}\| - 8 \\ E_{compress}(E_{\text{FNV}}(E_{\text{FNV}}(E_{\text{FNV}}(\mathbf{m}[i + 4], \mathbf{m}[i + 5]), \mathbf{m}[i + 6]), \mathbf{m}[i + 7]), i + 8) & \text{otherwise} \end{cases} \tag{327}$$

Appendix K. Anomalies on the Main Network

K.1. **Deletion of an Account Despite Out-of-gas.** At block 2675119, in the transaction 0xcf416c536ec1a19ed1fb89e4ec7ffb3cf73aa413b3aa9b77d60e4fd81a4296ba, an account at address 0x03 was called and an out-of-gas occurred during the call. Against the equation (195), this added 0x03 in the set of touched addresses, and this transaction turned $\boldsymbol{\sigma}[0x03]$ into $\varnothing$.

Appendix L. List of mathematical symbols

Symbol	Latex Command	Description
$\bigvee$	`\bigvee`	This is the least upper bound, supremum, or join of all elements operated on. Thus it is the greatest element of such elements (Davey and Priestley [2002]).

Wikipedia:What Wikipedia is not

From Wikipedia, the free encyclopedia

"WP:NOT" redirects here. For Wikipedia's notability guidelines, see Wikipedia:Notability.
"WP:!" redirects here. For other uses of ! in Wikipedia jargon, see Wikipedia:Glossary.
"WP:DEMOCRACY" redirects here. For Wikipedia's democratic structures, see WP:WikiProject Democracy.

This page documents an English Wikipedia policy.
It describes a widely accepted standard that all editors should normally follow. Changes made to it should reflect consensus.

Shortcut
WP:NOT

This page in a nutshell:

1. The amount of information on Wikipedia is practically unlimited, but Wikipedia is a digital encyclopedia and therefore does not aim to contain all data or expression found elsewhere.
2. Although anyone can be an editor, Wikipedia's community processes and standards make it neither an anarchy, democracy, nor bureaucracy.
3. Wikipedia is not a place to promote things, is not a thought-book, a website primarily used for communication, a freely-licensed media repository, nor a censored encyclopedia.

Wikipedia is an online encyclopedia and, as a means to that end, an online community of individuals interested in building and using a high-quality encyclopedia in a spirit of mutual respect. Therefore, **there are certain things that Wikipedia is *not***.

Core content policies
Neutral point of view
No original research
Verifiability
Other content policies
Article titles
Biographies of living persons
Image use policy
What Wikipedia is not
v • t • e

Contents [hide]

Style and format

Wikipedia is not a paper encyclopedia

Policy shortcuts
WP:NOTPAPER
WP:PAPER
WP:INKLESS

Wikipedia is not a paper encyclopedia, but a digital encyclopedia project. Other than verifiability and the other points presented on this page, there is no practical limit to the number of topics Wikipedia can cover or the total amount of content. However, there is an important distinction between what *can* be done, and what *should* be done, which is covered under § Encyclopedic content below. Consequently, this policy is not a free pass for inclusion: articles must abide by the appropriate content policies, particularly those covered in the **five pillars**.

Keeping articles to a reasonable size is important for Wikipedia's accessibility, especially for dial-up and mobile browser readers, since it directly affects page download time (see Wikipedia:Article size). Splitting long articles and leaving adequate summaries is a natural part of growth for a topic (see Wikipedia:Summary style). Some topics are covered by print encyclopedias only in short, static articles, but Wikipedia can include more information, provide more external links, and update more quickly.

Encyclopedic content

Policy shortcut
WP:NOTEVERYTHING

Information should not be included in this encyclopedia solely because it is true or useful. A Wikipedia article should not be a complete exposition of all possible details, but a *summary* of accepted knowledge regarding its subject.[1] Verifiable and sourced statements should be treated with appropriate weight. Although there are debates about the encyclopedic merits of several classes of entries, consensus is that the following are good examples of what Wikipedia is not. The examples under each section are not intended to be exhaustive.

Bahasa Indonesia
Interlingua
Íslenska
Italiano
עברית
Kalaallisut
ქართული
Kaszëbsczi
Қазақша
Ladino
Latviešu
Lëtzebuergesch
Lietuvių
Limburgs
Lumbaart
Magyar
Македонски
Malagasy
മലയാളം
मराठी
مصرى
Bahasa Melayu
Baso Minangkabau
Монгол
Nederlands
नेपाली
Napulitano
Nordfriisk
Norsk
Norsk nynorsk
Occitan
ਪੰਜਾਬੀ
Plattdüütsch
Polski
Поντιακά
Português
Ripoarisch
Română
Русиньскый
Русский
Саха тыла
Scots
Seeltersk
Shqip
සිංහල
Simple English
سنڌي
Slovenčina
Slovenščina
Ślůnski
Soomaaliga
کوردی
Српски / srpski
Srpskohrvatski / српскохрватски
Suomi
Svenska
தமிழ்
Tarandíne
Татарча/tatarça
ไทย
Тоҷикӣ
ᏣᎳᎩ
Türkçe
Türkmençe
Українська
اردو
Vèneto
Tiếng Việt
ייִדיש
Yorùbá
Žemaitėška
Edit links

Wikipedia is not a dictionary

Main page: Wikipedia:Wikipedia is not a dictionary

Policy shortcuts
WP:NOT#DICDEF
WP:NOT#DICT
WP:NOT#DICTIONARY

Wikipedia is not a dictionary, or a usage or jargon guide. Wikipedia articles are not:

1. **Definitions.** Articles should begin with a good definition or description, but articles that contain nothing more than a definition should be expanded with additional encyclopedic content. If they cannot be expanded beyond a definition, Wikipedia is not the place for them. In some cases, the definition of a word may be an encyclopedic subject, such as the definition of *planet*. For a wiki that *is* a dictionary, visit our sister project Wiktionary. Dictionary definitions should be transwikied there.
2. **Dictionary entries.** Encyclopedia articles are about a person, or a group, a concept, a place, a thing, an event, etc. In some cases, a word or phrase itself may be an encyclopedic subject, such as Macedonia (terminology) or truthiness. However, articles rarely, if ever, contain more than one *distinct* definition or usage of the article's title. Articles about the cultural or mathematical significance of individual numbers are also acceptable.
3. **Usage, slang, or idiom guides.** Descriptive articles about languages, dialects, or types of slang (such as Klingon language, Cockney, or Leet) are desirable. Prescriptive guides for prospective speakers of such languages are not. See "Wikipedia is not a manual, guidebook, textbook, or scientific journal" below for more information. For a wiki that *is* a collection of textbooks, visit our sister project Wikibooks. Prescriptive guides for prospective speakers of a language should be transwikied there.

Visit Wiktionary, our sister project, if you want to help build a dictionary. They would greatly appreciate your help.

Wikipedia is not a publisher of original thought

"WP:FORUM" redirects here. You may be looking for Wikipedia:Forum shopping or Wikipedia:Village pump.

Policy shortcuts
WP:FORUM
WP:NOTFORUM
WP:NOTESSAY
WP:NOTFANSITE

Wikipedia is not a place to publish your own thoughts and analyses or to publish new information. Per our policy on original research, please **do not use Wikipedia for any of the following**:

1. **Primary (original) research**, such as proposing theories and solutions, original ideas, defining terms, coining new words, etc. If you have completed primary research on a topic, your results should be published in other venues, such as peer-reviewed journals, other printed forms, open research, or respected online publications. Wikipedia can report your work after it is published and becomes part of accepted knowledge; however, citations of reliable sources are needed to demonstrate that material is verifiable, and not merely the editor's opinion.
2. **Personal inventions.** If you or a friend invented a drinking game, a new type of dance move, or even the word *frindle*, it is not notable enough to be given an article until multiple, independent, and reliable secondary sources report on it. And Wikipedia is *certainly* not for things made up one day.
3. **Personal essays** that state your particular feelings about a topic (rather than the opinions of experts). Although Wikipedia is supposed to compile human knowledge, it is not a vehicle to make personal opinions become part of such knowledge. In the unusual situation where the opinions of an individual are important enough to discuss, it is preferable to let other people write about them. (Personal essays on Wikipedia-related topics are welcome in your user namespace or on the Meta-wiki.)
4. **Discussion forums.** Please try to stay on the task of creating an encyclopedia. You can chat with people about Wikipedia-related topics on their user talk pages, and should resolve problems with articles on the relevant talk pages, but please do not take discussion into articles. In addition, bear in mind that article talk pages exist solely to discuss how to improve articles; they are not for general discussion about the subject of the article, nor are they a help desk for obtaining instructions or technical assistance. Material unsuitable for talk pages may be subject to removal per the talk page guidelines. If you wish to ask a specific question on a topic, Wikipedia has a Reference desk; questions should be asked there rather than on talk pages.

Wikipedia is not a soapbox or means of promotion

"WP:PROMOTION" redirects here. For other pages about advertising and promotion, see Wikipedia:Advertising.
"WP:SOAP" redirects here. For the Soap Operas WikiProject, see Wikipedia:WikiProject Soap Operas.

Policy shortcuts
WP:NOTADVERTISING
WP:NOTADVOCACY
WP:NOTBROCHURE
WP:NOTGOSSIP
WP:NOTCLICKBAIT
WP:NOTOPINION
WP:NOTSCANDAL
WP:NOTSOAPBOX
WP:NOTPROMOTION
WP:NOTPRESSRELEASE
WP:NOTPROMO
WP:NOTPROPAGANDA
WP:PROMOTION
WP:PRESSRELEASE
WP:PROMO
WP:PROPAGANDA
WP:NOTPLUG
WP:PLUG
WP:SOAP
WP:SOAPBOX

Wikipedia is not a soapbox, a battleground, or a vehicle for propaganda, advertising and showcasing. This applies to usernames, articles, categories, files, talk page discussions, templates, and user pages. Therefore, content hosted in Wikipedia is not for:

1. **Advocacy, propaganda, or recruitment** of any kind: commercial, political, scientific, religious, national, sports-related, or otherwise. An article can report objectively *about* such things, as long as an attempt is made to describe the topic from a neutral point of view. You might wish to start a blog or visit a forum if you want to convince people of the merits of your opinions.[2]
2. **Opinion pieces.** Although some topics, particularly those concerning current affairs and politics, may stir passions and tempt people to "climb soapboxes", Wikipedia is not the medium for this. Articles must be balanced to put entries, especially for current events, in a reasonable perspective, and represent a neutral point of view. Furthermore, Wikipedia authors should strive to write articles that will not quickly become obsolete. However, Wikipedia's sister project Wikinews allows commentaries on its articles.
3. **Scandal mongering,** promoting things "heard through the grapevine" or gossiping. Articles and content about living people are required to meet an especially high standard, as they may otherwise be libellous or infringe the subjects' right to privacy. Articles must not be written purely to attack the reputation of another person.
4. **Self-promotion.** It can be tempting to write about yourself or projects in which you have a strong personal involvement. However, remember that the standards for encyclopedic articles apply to such pages just like any other. This includes the requirement to maintain a neutral point of view, which can be difficult when writing about yourself or about projects close to you. Creating overly abundant links and references to autobiographical sources is unacceptable. See Wikipedia:Autobiography, Wikipedia:Notability and Wikipedia:Conflict of interest.
5. **Advertising, marketing or public relations.** Information about companies and products must be written in an objective and unbiased style, free of puffery. All article topics must be verifiable with independent, third-party sources, so articles about very small garage bands or local companies are typically unacceptable. Wikipedia articles about a company or organization are not an extension of their website or other social media marketing efforts. External links to commercial organizations are acceptable if they identify notable organizations which are the topic of the article. Wikipedia neither endorses organizations nor runs affiliate programs. See also Wikipedia:Notability (organizations and companies) for guidelines on corporate notability. Those promoting causes or events, or issuing public service announcements, even if noncommercial, should use a forum other than Wikipedia to do so. Contributors must disclose any payments they receive for editing Wikipedia. See also Wikipedia:Conflict of interest.

Non-disruptive statements of opinion on internal Wikipedia policies and guidelines may be made on user pages and within the *Wikipedia:* namespace, as

they are relevant to the current and future operation of the project. However, article talk pages should not be used by editors as platforms for their personal views on a subject (see Wikipedia:Talk page guidelines).

Wikipedia is not a mirror or a repository of links, images, or media files

Policy shortcuts
WP:LINKFARM
WP:NOTLINKFARM
WP:NOTLINK
WP:NOTMIRROR
WP:NOTREPOSITORY
WP:NOTIMAGE
WP:NOTGALLERY

Wikipedia is neither a mirror nor a repository of links, images, or media files.[3] Wikipedia articles are not merely collections of:

1. **External links** or **Internet directories.** There is nothing wrong with adding one or more useful content-relevant links to the external links section of an article; however, excessive lists can dwarf articles and detract from the purpose of Wikipedia. On articles about topics with many fansites, for example, including a link to one major fansite may be appropriate. See Wikipedia:External links for some guidelines.
2. **Internal links,** except for disambiguation pages when an article title is ambiguous, and for lists for browsing or to assist with article organization and navigation; for these, please follow relevant guidance at Wikipedia:Manual of Style/Lists, Wikipedia:Stand-alone lists and Wikipedia:Manual of Style/Embedded lists.
3. **Public domain or other source material** such as entire books or source code, original historical documents, letters, laws, proclamations, and other source material that are only useful when presented with their original, unmodified wording. Complete copies of primary sources may go into Wikisource, but not on Wikipedia. Public domain resources such as the 1911 *Encyclopædia Britannica* may be used to add content to an article (see Plagiarism guideline: Public-domain sources for guidelines on doing so). See also Wikipedia:Do not include the full text of lengthy primary sources and Wikisource's inclusion policy.
4. **Photographs or media files** with no accompanying text. If you are interested in presenting a picture, please provide an encyclopedic context, or consider adding it to Wikimedia Commons. If a picture comes from a public domain source on a website, then consider adding it to Wikipedia:Images with missing articles or Wikipedia:Public domain image resources.

Wikipedia is not a blog, web hosting service, social networking service, or memorial site

"WP:MEMORIAL" redirects here. For a list of deceased Wikipedians, see Wikipedia:Deceased Wikipedians.
Further information: Wikipedia:User pages

Policy shortcuts
WP:NOTWEBHOST
WP:NOTHOSTING
WP:NOTSTORAGE
WP:NOTWIKIA
WP:NOTBLOG
WP:NOTRESUME
WP:NOTSOCIAL
WP:NOTSOCIALMEDIA
WP:NOTSOCIALNETWORK
WP:NOTOBITUARY
WP:NOTMEMORIAL
WP:NOTDATINGSERVICE
WP:NOTGAMEHOST
WP:NOTCV

Wikipedia is not a social networking service like Facebook or Twitter. You may not host your own website, blog, wiki, résumé, or cloud on Wikipedia. Wikipedia pages, **including those in user space**, are not:

1. **Personal web pages.** Wikipedians have individual user pages, but they should be used primarily to present information relevant to work on the encyclopedia. Limited autobiographical information is allowed, but user pages do not serve as personal webpages, blogs, or repositories for large amounts of material irrelevant to collaborating on Wikipedia. If you want to post your résumé or make a personal webpage, please use one of the many free providers on the Internet or any hosting included with your Internet service provider. The focus of user pages *should not* be social networking or amusement, but rather providing a foundation for effective collaboration. Humorous pages that refer to Wikipedia in some way may be created in an appropriate namespace. Personal web pages are often speedily deleted. Wikipedia articles use formal English and are not written in Internet posting style.
2. **File storage areas.** Please upload only files that are used (or will be used) in encyclopedia articles or project pages; anything else will be deleted. If you have extra relevant images, consider uploading them to the Wikimedia Commons, where they can be linked from Wikipedia.
3. **Dating services.** Wikipedia is not an appropriate place to pursue relationships or sexual encounters. User pages that move beyond broad expressions of sexual orientation are unacceptable. However, you very well may form new friendships as you go about improving the encyclopedia.
4. **Memorials.** Subjects of encyclopedia articles must satisfy Wikipedia's notability requirements. Wikipedia is not the place to memorialize deceased friends, relatives, acquaintances, or others who **do not meet such requirements**. (However, for the Wikipedia page for deceased Wikipedia editors, see WP:RIP).
5. **Content for projects unrelated to Wikipedia.** Do not store material unrelated to Wikipedia, including in userspace. Please see WP:UPNOT for examples of what may not be included.

If you are interested in using the wiki technology for a collaborative effort on something else, even just a single page, many free and commercial sites provide wiki hosting. You can also install wiki software on your server. See the installation guide at MediaWiki.org for information on doing this. See also Wikipedia:Alternative outlets.

Your user page is **not yours**. It is a part of Wikipedia, and exists to make collaboration among Wikipedians easier, not for self-promotion. See Wikipedia:User pages for current consensus guidelines on user pages.

Wikipedia is not a directory

"WP:DIRECTORY" redirects here. For a listing of Wikipedia's directories and indexes, see Wikipedia:Directory.
See also: Wikipedia:Manual of Style/Lists, Wikipedia:Stand-alone lists, and Wikipedia:Embedded lists

Policy shortcuts
WP:NOTBIBLIOGRAPHY
WP:NOTCATALOG
WP:NOTCATALOGUE
WP:NOTDIR
WP:NOTDIRECTORY
WP:NOTGENEALOGY
WP:NOTQUOTE
WP:NOTRADIOGUIDE
WP:NOTTVGUIDE
WP:NOTWHITE
WP:NOTYELLOW
WP:YELLOW
WP:YELLOWPAGES

Wikipedia encompasses many lists of links to articles within Wikipedia that are used for internal organization or to describe a notable subject. In that sense, Wikipedia functions as an index or directory of its own content. However, Wikipedia is not a directory of everything in the universe that exists or has existed. Please see Wikipedia:Alternative outlets for alternatives. Wikipedia articles are not:

1. **Lists or repositories of loosely associated topics** such as (but not limited to) quotations, aphorisms, or persons (real or fictional). If you want to enter lists of quotations, put them into our sister project Wikiquote. Of course, there is nothing wrong with having lists if their entries are relevant *because* they are associated with or significantly contribute to the list topic. Wikipedia also includes reference tables and tabular information for quick reference. *Merged groups of small articles* based on a core topic are permitted. (See Wikipedia:Stand-alone lists § Appropriate topics for lists for clarification.)
2. **Genealogical entries**. Family histories should be presented only where appropriate to support the reader's understanding of a notable topic.
3. **The White or Yellow Pages**. Contact information such as phone numbers, fax numbers and e-mail addresses is not encyclopedic. Likewise, disambiguation pages (such as John Smith) are not intended to be complete listings of every person in the world named John Smith—just the notable ones.
4. **Directories, directory entries, electronic program guides, or resources for conducting business**. For example, an article on a broadcaster should not list upcoming events, current promotions, current schedules, format clocks, etc., although mention of major events, promotions or historically significant program lists and schedules may be acceptable. Likewise an article on

Nuh-uh

a business should not contain a list of all the company's patent filings. Furthermore, the Talk pages associated with an article are for talking about the article, not for conducting the business of the topic of the article.

5. **Sales catalogues**. An article should not include product pricing or availability information unless there is an independent source *and* a justified reason for the mention. Encyclopedic significance may be indicated if mainstream media sources (not just product reviews) provide commentary on these details instead of just passing mention. Prices and product availability can vary widely from place to place and over time. Wikipedia is not a price comparison service to compare the prices of competing products, or the prices and availability of a single product from different vendors or retailers.
6. **Non-encyclopedic cross-categorizations**, such as "people from ethnic / cultural / religious group X employed by organization Y" or "restaurants specializing in food type X in city Y". Cross-categories such as these are not considered a sufficient basis for creating an article, unless the intersection of those categories is in some way a culturally significant phenomenon. See also Wikipedia:Overcategorization for this issue in categories.
7. **Simple listings** without context information. Examples include, but are not limited to: listings of business alliances, clients, competitors, employees (except CEOs, supervisory directors and similar top functionaries), equipment, estates, offices, store locations, products and services, sponsors, subdivisions and tourist attractions. Information about **relevant** single entries with encyclopedic information should be added as sourced prose. Lists of creative works in a wider context are permitted.

Wikipedia is not a manual, guidebook, textbook, or scientific journal

Policy shortcuts
WP:GAMEGUIDE
WP:NOTADVICE
WP:NOTCASE
WP:NOTCOOKBOOK
WP:NOTFAQ
WP:NOTGUIDE
WP:NOTHOWTO
WP:NOTJARGON
WP:NOTMANUAL
WP:NOTJOURNAL
WP:NOTPAPERS
WP:NOTRECIPE
WP:NOTTEXTBOOK
WP:NOTTRAVEL

Wikipedia is an encyclopedic reference, not an instruction manual, guidebook, or textbook. Wikipedia articles should not read like:

1. **Instruction manuals.** While Wikipedia has descriptions of people, places and things, an article should not read like a "how-to" style owner's manual, cookbook, advice column (legal, medical or otherwise) or suggestion box. This includes tutorials, instruction manuals, game guides, and recipes. Describing to the reader how people or things use or do something is encyclopedic; instructing the reader in the imperative mood about how to use or do something is not.[4] Such guides may be welcome at Wikibooks instead.
2. **Travel guides.** An article on Paris should mention landmarks, such as the Eiffel Tower and the Louvre, but not the telephone number or street address of the "best" restaurants, nor the current price of a *café au lait* on the Champs-Élysées. Wikipedia is not the place to recreate content more suited to entries in hotel or culinary guides, travelogues, and the like. Notable locations may meet the inclusion criteria, but the resulting articles need not include every tourist attraction, restaurant, hotel or venue, etc. While travel guides for a city will often mention distant attractions, a Wikipedia article for a city should only list those that are actually in the city. If you *do* wish to help write a travel guide, your contributions would be welcome at our sister project, Wikivoyage.
3. **Video game guides.** An article about a video game should briefly summarize the story and the main actions the player performs in the game; however, avoid lists of gameplay weapons, items, or concepts, unless these are notable as discussed in secondary sources in their own right in gaming context (such as the BFG9000 from the *Doom* series). Walk-throughs or detailed coverage of specific point values, achievements, time-limits, levels, types of enemies, character moves, character weight classes, and so on are also considered inappropriate. A concise summary is appropriate if it is essential to understanding the game or its significance in the industry. See also WP:VGSCOPE.
4. **Internet guides.** Wikipedia articles should not exist ***only*** to describe the nature, appearance or services a website offers, but should also describe the site in an *encyclopedic manner*, offering detail on a website's achievements, impact or historical significance, which can be kept significantly more up-to-date than most reference sources, since editors can incorporate new developments and facts as they are made known. See the Current events portal for examples.
5. **FAQs.** Wikipedia articles should not list frequently asked questions (FAQs). Instead, format the information as neutral prose within the appropriate article(s).
6. **Textbooks and annotated texts.** Wikipedia is an encyclopedic reference, not a textbook. The purpose of Wikipedia is to present facts, not to teach subject matter. It is not appropriate to create or edit articles that read as textbooks, with leading questions and systematic problem solutions as examples. These belong on our sister projects, such as Wikibooks, Wikisource, and Wikiversity. Some kinds of examples, specifically those intended to *inform* rather than to *instruct*, may be appropriate for inclusion in a Wikipedia article.
7. **Scientific journals.** A Wikipedia article should not be presented on the assumption that the reader is well-versed in the topic's field. Introductory language in the lead (and sometimes the initial sections) of the article should be written in plain terms and concepts that can be understood by any literate reader of Wikipedia without any knowledge in the given field before advancing to more detailed explanations of the topic. While wikilinks should be provided for advanced terms and concepts in that field, articles should be written on the assumption that the reader will not or cannot follow these links, instead attempting to infer their meaning from the text. See Wikipedia:Manual of Style/Linking.
8. **Academic language.** Texts should be written for everyday readers, not just for academics. Article titles should reflect common usage, not academic terminology, whenever possible.
9. **Case studies.** Many topics are based on the relationship of *factor X* to *factor Y*, resulting in one or more full articles. For example, this could refer to *situation X* in *location Y*, or *version X* of *item Y*. This is perfectly acceptable when the two variables put together represent some culturally significant phenomenon or some otherwise notable interest. Often, separate articles are needed for a subject within a range of different countries, due to substantial differences across international borders; articles such as "Slate industry in Wales" and "Island fox" are fitting examples. Writing about "Oak trees in North Carolina" or "Blue trucks", however, would likely constitute a POV fork or original research, and would certainly not result in an encyclopedic article.

It's a cookbook!
(But Wikipedia isn't.)

Wikipedia is not a crystal ball

Policy shortcuts
WP:BALL
WP:CBALL
WP:CRYSTAL
WP:CRYSTALBALL
WP:NOTCRYSTAL
WP:NOTCRYSTALBALL
WP:FUTURE
WP:NOTFUTURE
WP:RUMOUR
WP:NOTRUMOR
WP:SPECULATION
WP:NOTSPECULATION

Wikipedia is not a collection of unverifiable speculation or presumptions. Wikipedia does not predict the future. All articles about anticipated events must be verifiable, and the subject matter must be of sufficiently wide interest that it would merit an article if the event had already occurred. It *is* appropriate to report discussion and arguments about the prospects for success of future proposals and projects or whether some development will occur, if discussion is properly referenced. It is *not* appropriate for editors to insert their own opinions or analyses. Predictions, speculation, forecasts and theories stated by reliable, expert sources or recognized entities in a field may be included, though editors should be aware of creating undue bias to any specific point-of-view. In forward-looking articles about unreleased products, such as films and games, take special care to avoid advertising and unverified claims (for films, see WP:NFF). In particular:

1. Individual **scheduled or expected future events** should be included only if the event is notable and almost certain to take place. Dates are **not definite** until the event actually takes place. If preparation for the event is not already in progress, speculation about it must be well documented. Examples of appropriate topics include the 2020 U.S. presidential election and 2024 Summer Olympics. By comparison, the 2032 U.S. presidential election and 2040 Summer Olympics or events surrounding the 250th anniversary of the United States of America in 2026 are not appropriate article topics if nothing can be

said about them that is verifiable and not original research. Avoid predicted sports team line-ups, which are inherently unverifiable and speculative. A schedule of future events may be appropriate if it can be verified. As an exception, even highly speculative articles about events that may or may not occur far in the future might be appropriate, where coverage in reliable sources is sufficient. For example, Ultimate fate of the universe is an acceptable topic.

2. Individual items from a **predetermined list or a systematic pattern of names,** pre-assigned to future events or discoveries, are not suitable article topics, if only generic information is known about the item. Lists of tropical cyclone names is encyclopedic; "Tropical Storm Arthur (2020)" is not, even though it is virtually certain that such a storm will occur. Similarly, articles about **words formed on a predictable numeric system** (such as "septenquinquagintillion") are not encyclopedic unless they are defined on good authority, or genuinely in use. Certain scientific extrapolations are considered to be encyclopedic, such as chemical elements documented by IUPAC before isolation in the laboratory, provided that scientists have made significant non-trivial predictions of their properties.
3. Articles that present original research in the form of **extrapolation, speculation, and "future history"** are inappropriate. Although scientific and cultural norms continually evolve, we must wait for this evolution to happen, rather than try to predict it. Of course, we do and should have articles *about* **notable** *artistic works, essays, or credible research* that embody predictions. An article on weapons in *Star Trek* is appropriate; an article on "Weapons to be used in World War III" is not.
4. Although currently accepted scientific paradigms may later be rejected, and hypotheses previously held to be controversial or incorrect sometimes become accepted by the scientific community, it is not the place of Wikipedia to venture such projections.

... but Wikipedia does not.

5. **Wikipedia is not a collection of product announcements and rumors.** Although Wikipedia includes up-to-date knowledge about newly revealed products, short articles that consist only of product announcement information are not appropriate. Until such time that more encyclopedic knowledge about the product can be verified, product announcements should be merged to a larger topic (such as an article about the creator(s), a series of products, or a previous product) if applicable. Speculation and rumor, even from reliable sources, are not appropriate encyclopedic content.

Wikipedia is not a newspaper

See also: Wikipedia:Notability (events) and Wikipedia:Too much detail

Policy shortcuts
WP:NOTNEWS
WP:NOTNEWSPAPER
WP:NOTWHOSWHO
WP:NOTDIARY
WP:NOTADIARY
WP:DIARY

Editors are encouraged to include current and up-to-date information within its coverage, and to develop stand-alone articles on significant current events. However, not all verifiable events are suitable for inclusion in Wikipedia. Ensure that Wikipedia articles are not:

1. **Original reporting.** Wikipedia should not offer first-hand news reports on breaking stories. Wikipedia does not constitute a primary source. However, our sister projects Wikisource and Wikinews do exactly that, and *are* intended to be primary sources. Wikipedia does have many *encyclopedia articles* on topics of historical significance that are currently in the news, and can be updated with recently verified information.
2. **News reports.** Wikipedia considers the enduring notability of persons and events. While news coverage can be useful source material for encyclopedic topics, most newsworthy events do not qualify for inclusion. For example, routine news reporting of announcements, sports, or celebrities is not a sufficient basis for inclusion in the encyclopedia. While including information on recent developments is sometimes appropriate, breaking news should not be emphasized or otherwise treated differently from other information. Timely news subjects not suitable for Wikipedia may be suitable for our sister project Wikinews. Wikipedia is also not written in news style.
3. **Who's who.** Even when an event is notable, individuals involved in it may not be. Unless news coverage of an individual goes beyond the context of a single event, our coverage of that individual should be limited to the article about that event, in proportion to their importance to the overall topic. (See Wikipedia:Biographies of living persons for more details.)
4. **A diary.** Even when an individual is notable, not all events they are involved in are. For example, news reporting about celebrities and sports figures can be very frequent and cover a lot of trivia, but using all these sources would lead to over-detailed articles that look like a diary. Not every match played or goal scored is significant enough to be included in the biography of a person.

Wikipedia is not an indiscriminate collection of information

"WP:PLOT" redirects here. For information regarding plot summary manuals of style, see MOS:PLOT.
See also: Wikipedia:Notability and Wikipedia:Manual of Style/Trivia sections

Policy shortcuts
WP:IINFO
WP:INDISCRIMINATE
WP:NOTCHANGELOG
WP:NOTLYRICS
WP:NOTSTATS
WP:NOTSTATSBOOK
WP:PLOT
WP:NOTPLOT
WP:RAWDATA
WP:WHIM
WP:RELEASENOTES

To provide encyclopedic value, data should be put in context with explanations referenced to independent sources. As explained in § Encyclopedic content above, merely being true, or even verifiable, does not automatically make something suitable for inclusion in the encyclopedia. Wikipedia articles should not be:

1. **Summary-only descriptions of works.** Wikipedia treats creative works (including, for example, works of art or fiction, video games, documentaries, research books or papers, and religious texts) in an encyclopedic manner, discussing the development, design, reception, significance, and influence of works in addition to concise summaries of those works. For more information regarding summaries, see Wikipedia:Manual of Style/Writing about fiction § Contextual presentation.
2. **Lyrics databases.** An article about a song should provide information about authorship, date of publication, social impact, and so on. Quotations from a song should be kept to a reasonable length relative to the rest of the article, and used to facilitate discussion, or to illustrate the style; the full text can be put on Wikisource and linked to from the article. Most song lyrics published after 1922 are protected by copyright; any quotation of them must be kept to a minimum, and used for direct commentary or to illustrate some aspect of style. Never link to the lyrics of copyrighted songs unless the linked-to site clearly has the right to distribute the work. See Wikipedia:Do not include the full text of lengthy primary sources for full discussion.
3. **Excessive listings of unexplained statistics.** Statistics that lack context or explanation can reduce readability and may be confusing; accordingly, statistics should be placed in tables to enhance readability, and articles with statistics should include explanatory text providing context. Where statistics are so lengthy as to impede the readability of the article, the statistics can be split into a separate article and summarized in the main article. (e.g., statistics from the main article United States presidential election, 2012 have been moved to a related article Nationwide opinion polling for the United States presidential election, 2012). Wikipedia:Notability#Stand-alone lists offers more guidance on what kind of lists are acceptable, and Wikipedia:Stand-alone lists#Selection criteria offers guidance on what entries should be included.
4. **Exhaustive logs of software updates.** Use reliable third-party (not self-published or official) sources in articles dealing with software updates to describe the versions listed or discussed in the article. Common sense must be applied with regard to the level of detail to be included.

Wikipedia is not censored

Main page: Wikipedia:Content disclaimer

Policy shortcuts

See also: Censorship of Wikipedia

Policy shortcuts
WP:CENSOR
WP:CENSORED
WP:UNCENSORED
WP:NOTCENSORED

Wikipedia may contain content that some readers consider objectionable or offensive—even exceedingly so. Attempting to ensure that articles and images will be acceptable to all readers, or will adhere to general social or religious norms, is incompatible with the purposes of an encyclopedia.

Content *will* be removed if it is judged to violate Wikipedia policies (especially those on biographies of living persons and neutral point of view) or the laws of the United States (where Wikipedia is hosted). However, because most edits are displayed immediately, inappropriate material may be visible to readers, for a time, before being detected and removed.

> The University is not engaged in making ideas safe for students. It is engaged in making students safe for ideas. Thus it permits the freest expression of views before students, trusting to their good sense in passing judgment on these views.
>
> — University of California President Clark Kerr (1961)

Some articles may include images, text, or links which are relevant to the topic but that some people find objectionable. Discussion of potentially objectionable content should usually focus not on its potential offensiveness but on whether it is an appropriate image, text, or link. Beyond that, "being objectionable" is generally not sufficient grounds for the removal of content. The Wikipedia:Offensive material guideline can help assess appropriate actions to take in the case of content that may be considered offensive.

Some organizations' rules or traditions call for secrecy with regard to certain information about them. Such restrictions do not apply to Wikipedia, because Wikipedia is not a member of those organizations; thus Wikipedia will not remove such information from articles if it is otherwise encyclopedic.

Community

The above policies are about Wikipedia's content. The following relate to Wikipedia's governance and processes.

Wikipedia is not an anarchy or forum for free speech

WP is Encyclopedists' Corner, not Speakers' Corner.

"WP:ANARCHY" redirects here. For WikiProject Anarchism, see Wikipedia:WikiProject Philosophy/Anarchism.
See also: m:Power structure, WP:User access levels, and WP:Enforcement
Main page: Wikipedia:Administration

Policy shortcuts
WP:NOTANARCHY
WP:NOTFREESPEECH
WP:CHAOS

Wikipedia is free and open, but restricts both freedom and openness where they interfere with creating an encyclopedia. Accordingly, Wikipedia is not a forum for unregulated free speech. The fact that Wikipedia is an open, self-governing project does not mean that any part of its purpose is to explore the viability of anarchist communities. Our purpose is to build an encyclopedia, not to test the limits of anarchism.

Wikipedia is not a democracy

See also: Wikipedia:Polling is not a substitute for discussion and Wikipedia:Elections

Policy shortcuts
WP:DEM
WP:NOTDEM
WP:DEMOCRACY
WP:NOTDEMOCRACY

Wikipedia is not an experiment in democracy or any other political system. Its primary (though not exclusive) means of decision making and conflict resolution is editing and discussion leading to consensus—*not* voting (voting is used for certain matters such as electing the Arbitration Committee). Straw polls are sometimes used to test for consensus, but polls or surveys can impede, rather than foster, discussion and should be used with caution.

Wikipedia is not a bureaucracy

See also: Wikipedia:Ignore all rules
"WP:BURO" and "WP:BUREAU" redirect here. For the "bureaucrat" user access level, see WP:CRAT.

Policy shortcuts
WP:BURO
WP:NOTBURO
WP:NOTBUREAUCRACY
WP:BUREAU
WP:NOTBUREAU
WP:NOTLAW
WP:NOTSTATUTE
WP:NOTCOURT

While Wikipedia has many elements of a bureaucracy,[5] it is not governed by statute: it is not a quasi-judicial body, and rules are not the purpose of the community. Although some rules may be enforced, the written rules themselves do not set accepted practice. Rather, they document already existing community consensus regarding what should be accepted and what should be rejected.

While Wikipedia's written policies and guidelines should be taken seriously, they can be misused. Do not follow an overly strict interpretation of the *letter* of policies without consideration for their *principles*. If the rules truly prevent you from improving the encyclopedia, ignore them. Disagreements are resolved through consensus-based discussion, not by tightly sticking to rules and procedures. Furthermore, policies and guidelines themselves may be changed to reflect evolving consensus.

A procedural error made in a proposal or request is not grounds for rejecting that proposal or request.

A procedural, coding, or grammatical error in a new contribution is not grounds for reverting it, unless the error cannot easily be fixed.

Wikipedia is not a laboratory

Policy shortcut
WP:NOTLAB

Research about Wikipedia's content, processes, and the people involved[6] can provide valuable insights and understanding that benefit public knowledge, scholarship, and the Wikipedia community, but Wikipedia is not a public laboratory. Research that analyzes articles, talk pages, or other content on Wikipedia is not typically controversial, since all of Wikipedia is open and freely usable. However, research projects that are disruptive to the community or which negatively affect articles—even temporarily—are not allowed and can result in loss of editing privileges. Before starting a potentially controversial project,[7] researchers should open discussion at the Village Pump to ensure it will not interfere with Wikipedia's mission. Regardless of the type of project, researchers are advised to be as transparent as possible on their user pages, disclosing information such as institutional connections and intentions.[8]

Some editors explicitly request to not be subjects in research and experiments. A list of some of those editors can be found here. Please respect the wish of editors to opt-out of research .

Wikipedia is not a battleground

See also: Wikipedia:Wikipedia is not about winning and Wikipedia:Edit warring

Policy shortcuts
WP:BATTLEGROUND
WP:NOTBATTLE
WP:NOTBATTLEGROUND
WP:NOTFACTIONS
WP:BATTLE
WP:FACTIONS

Wikipedia is not a place to hold grudges, import personal conflicts, carry on ideological battles, or nurture prejudice, hatred, or fear. Making personal battles out of Wikipedia discussions goes directly against our policies and goals. In addition to avoiding battles in discussions, do not try to advance your position in disagreements by making unilateral changes to policies. Do not disrupt Wikipedia to illustrate a point.

Every user is expected to interact with others civilly, calmly, and in a spirit of cooperation. Do not insult, harass, or intimidate those with whom you have a disagreement. Rather, approach the matter intelligently and engage in polite discussion. If another user behaves in an

uncivil, uncooperative, or insulting manner, or even tries to harass or intimidate you, this does not give you an excuse to respond in kind. Address only the factual points brought forward, ignoring the inappropriate comments, or disregard that user entirely. If necessary, point out gently that you think the comments might be considered uncivil, and make it clear that you want to move on and focus on the content issue. If a conflict continues to bother you, take advantage of Wikipedia's dispute resolution process. There are always users willing to mediate and arbitrate disputes between others.

In large disputes, resist the urge to turn Wikipedia into a battleground between factions. Assume good faith that every editor and group is here to improve Wikipedia—especially if they hold a point of view with which you disagree. Work with whomever you like, but do not organize a faction that disrupts (or aims to disrupt) Wikipedia's fundamental decision-making process, which is based on building a consensus. Editors in large disputes should work in good faith to find broad principles of agreement between different viewpoints.

Do not use Wikipedia to make legal or other threats against Wikipedia, Wikipedians, or the Wikimedia Foundation—other means already exist to communicate legal problems.[9] Threats are not tolerated and may result in a ban.

Wikipedia is not compulsory

See also: Wikipedia:Wikipedia is a volunteer service

Policy shortcuts
WP:CHOICE
WP:COMPULSORY
WP:NOTCOMPULSORY
WP:NOTREQUIRED
WP:REQUIRED

Wikipedia is a volunteer community and does not require the Wikipedians to give any more time and effort than they wish. Focus on improving the encyclopedia itself, rather than demanding more from other Wikipedians. Editors are free to take a break or leave Wikipedia at any time.

And finally…

Policy shortcuts
WP:BADIDEA
WP:NOTSTUPID

Wikipedia is not any of a very long list of terrible ideas. We cannot anticipate every bad idea that someone might have. Almost everything on this page made it here because somebody managed to come up with some new bad idea that had not been anticipated. (See WP:BEANS—it is, in fact, *strongly discouraged* to anticipate them.) In general, "that is a terrible idea" is always sufficient grounds to avoid doing something, provided there is a good reason that the idea is terrible.

When you wonder what to do

Policy shortcut
WP:WHATISTOBEDONE

When you wonder what should or should not be in an article, ask yourself what a reader would expect to find under the same heading *in an encyclopedia*.

When you wonder whether the rules given above are being violated, consider:

- Modifying the content of an article (normal editing).
- Turning the page into a redirect, preserving the page history.
- Nominating the page for deletion if it meets grounds for such action under the Deletion policy page. To develop an understanding of what kinds of contributions are in danger of being deleted, you have to regularly follow discussions there.
- Changing the rules on this page after a consensus has been reached following appropriate discussion with other Wikipedians via the Talk page. When adding new options, please be as clear as possible and provide counter-examples of similar, but permitted, subjects.

Wikipedia:Articles for deletion/Common outcomes is not official policy, but can be referred to as a record of what has and has not been considered encyclopedic in the past.

See also

- Wikipedia:Template messages/Cleanup § Style of writing—a list of templates that can be used to tag potentially inappropriate content when you can't fix the problem immediately yourself
- wmf:Resolution:Controversial content
- Texts titled "Wikipedia is ..." and "Wikipedia is not..."
- Wikipedia:Avoiding common mistakes
- Wikipedia:Alternative outlets
- Wikipedia:Articles for deletion/Common outcomes
- Wikipedia:Here to build an encyclopedia
- Wikipedia:Recentism
- Wikipedia:Why was the page I created deleted?

Notes

1. ^ See Wikipedia:Requests for arbitration/Rex071404 § Final decision, which suggested a similar principle in November 2004.
2. ^ Wikipedia article pages (and various navigational pages: categories, navboxes, disambiguation pages, etc.) are off limits for any advocacy. Talk pages, use space pages and essays are venues where you can advocate your opinions provided that they are directly related to the improvement of Wikipedia and are n disruptive.
3. ^ The English Wikipedia incorporates many images and some text which are considered "fair use" into its free content articles. (Other language Wikipedias often *do not*.) See also Wikipedia:Copyrights.
4. ^ The how-to restriction does not apply to the project namespace, where "how-to"s relevant to editing Wikipedia itself are appropriate, such as Wikipedia:How to draw a diagram with Dia.
5. ^ Joseph Michael Reagle, Jr.; Lawrence Lessig (2010). *Good Faith Collaboration: The Culture of Wikipedia*. MIT Press. pp. 90–91. ISBN 9780262014472.
6. ^ See list of academic studies of Wikipedia, Research resources at Wikimedia Meta, the Meta research newsletter, and the Wikimedia Foundation research blog.
7. ^ "Projects that are "potentially controversial" include, but are not limited to, any project that involves directly changing article content (contributors are expected to have as their primary motivation the betterment of the encyclopedia, without a competing motivation such as research objectives), any project that involves contacting a very large number of editors, and any project that involves asking sensitive questions about their real-life identities.
8. ^ See also Researching Wikipedia, Ethically researching Wikipedia, as well as the conflict of interest guideline and paid-contribution disclosure policy (if researchers editing Wikipedia are being paid under grants to do so, this is paid editing that must be disclosed).
9. ^ If you believe that your legal rights are being violated, you may discuss this with other users involved, take the matter to the appropriate mailing list, contact the Wikimedia Foundation, or in cases of copyright violations notify us at Wikipedia:Contact us/Article problem/Copyright.

External links

Similar official policies on sister projects

- Wikibooks:What is Wikibooks
- Wikimedia Commons: What Commons is not
- Wikinews:What Wikinews is not
- Wikisource:What is Wikisource?
- Wikispecies: What Wikispecies is not
- Wiktionary:What Wiktionary is not
- Wikiquote:What Wikiquote is not
- Wikiversity:What Wikiversity is not
- Wikivoyage:Goals and non-goals

v · t · e	**Wikipedia principles**			
Five pillars Statement of our principles	**Jimbo's statement** Historic principles	**Simplified ruleset** Synopsis of our conventions	**Wikimedia principles** Common to all projects (in Meta-Wiki)	**Principles** Other essays on Wikipedia's principles

v · t · e	**Wikipedia key policies and guidelines**		
Five pillars (**What Wikipedia is not** · Ignore all rules)			
Content	✓	Verifiability · No original research · Neutral point of view · **What Wikipedia is not** · Biographies of living persons · Image use · Article titles	
	✓	Notability · Autobiography · Citing sources · Reliable sources (medicine) · Do not include copies of lengthy primary sources · Plagiarism · Don't create hoaxes · Fringe theories · Patent nonsense · External links · Portal namespace	
Conduct	✓	Civility · Consensus · Editing policy · Harassment · Vandalism · Ignore all rules · No personal attacks · Ownership of content · Edit warring · Dispute resolution · Sock puppetry · No legal threats · Child protection · Paid-contribution disclosure	
	✓	Assume good faith · Conflict of interest · Disruptive editing · Do not disrupt Wikipedia to illustrate a point · Etiquette · Gaming the system · Please do not bite the newcomers · Courtesy vanishing · Responding to threats of harm	
Deletion	✓	Deletion policy · Proposed deletion (Biographies · Books) · Criteria for speedy deletion · Attack page · Oversight · Revision deletion	
Enforcement	✓	Administrators · Banning · Blocking · Page protection	
Editing	✓	Article size · Be bold · Disambiguation · Hatnotes · Talk page guidelines (Signatures) · Broad-concept article	
		Style	Manual of Style (Contents) · Accessibility (Understandability) · Dates and numbers · Images · Layout · Lead section · Linking · Lists
		Classification	Categories, lists, and navigation templates · Categorization · Template namespace
Project content	✓	Project namespace (WikiProjects) · User pages (User boxes) · Shortcuts · Subpages	
WMF	✓	List of policies · Friendly space policy · Licensing and copyright · Privacy policy · Values · FAQ	
List of all policies and guidelines (✓ List of policies · ✓ List of guidelines) · Lists of attempts in creating fundamental principles			

Categories: Wikipedia policies | Wikipedia content policies

This page was last edited on 24 February 2019, at 07:43 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Cookie statement Mobile view

Wikipedia:Verifiability

From Wikipedia, the free encyclopedia

"WP:V" redirects here. To discuss particular sources, see the reliable sources noticeboard. For vandalism, see WP:VD. For the default Wikipedia skin, see WP:VECTOR.

This page documents an English Wikipedia policy.
It describes a widely accepted standard that all editors should normally follow. Changes made to it should reflect consensus.

Shortcuts
WP:V
WP:VER
WP:VERIFY

This page in a nutshell: Readers must be able to check that any of the information within Wikipedia articles is not just made up. This means all material must be attributable to reliable, published sources. Additionally, quotations and any material challenged or likely to be challenged must be supported by inline citations.

In Wikipedia, **verifiability** means that other people using the encyclopedia can check that the information comes from a reliable source. Wikipedia does not publish original research. Its content is determined by previously published information rather than the beliefs or experiences of its editors. Even if you're sure something is true, it must be verifiable before you can add it.[1] If reliable sources disagree, then maintain a neutral point of view and present what the various sources say, giving each side its due weight.

All material in Wikipedia mainspace, including everything in articles, lists and captions, must be verifiable. All quotations, and any material whose verifiability has been challenged or is likely to be challenged, must include an inline citation that directly supports the material. Any material that needs a source but does not have one may be removed. Please immediately remove contentious material about living people that is unsourced or poorly sourced.

For how to write citations, see citing sources. Verifiability, no original research, and neutral point of view are Wikipedia's core content policies. They work together to determine content, so editors should understand the key points of all three. Articles must also comply with the copyright policy.

Core content policies
Neutral point of view
No original research
Verifiability

Other content policies
Article titles
Biographies of living persons
Image use policy
What Wikipedia is not

V • T • E

Contents [hide]

Responsibility for providing citations

"WP:BURDEN" redirects here. For the responsibility to get consensus on content inclusion, see WP:ONUS.
"WP:PROVEIT" redirects here. For the editing tool, see Wikipedia:ProveIt.
See also: Wikipedia:Editing policy § Try to fix problems

Policy shortcuts
WP:UNSOURCED
WP:CHALLENGE
WP:BURDEN
WP:PROVEIT

All content must be verifiable. The **burden to demonstrate verifiability lies with the editor who adds or restores material**, and is satisfied by providing an inline citation to a reliable source that directly supports[2] the contribution.[3]

Attribute all quotations and any material whose verifiability is **challenged or likely to be challenged** to a reliable, published source using an inline citation. The cited source must clearly support the material as presented in the article. Cite the source clearly and precisely (specifying page, section, or such divisions as may be appropriate). See Citing sources for details of how to do this.

Any material lacking a reliable source directly supporting it may be removed and should not be restored without an inline citation to a reliable source.

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons
Wikiversity

Languages
Afrikaans
العربية
অসমীয়া
Asturianu
Azərbaycanca
বাংলা
Беларуская
Беларуская (тарашкевіца)
भोजपुरी
Български
Català
Čeština
Cymraeg
Dansk
Deutsch
Eesti
Ελληνικά
Español
Esperanto
Euskara
فارسی
Français
Galego
گیلکی
ગુજરાતી
□□□
Hausa
Հայերեն
हिन्दी
Hrvatski
Ilokano
Bahasa Indonesia
Interlingua
Íslenska
Italiano
■■■■■
ქართული
Қазақша
Latina
Lietuvių
Magyar
Македонски
മലയാളം

Bahasa Melayu
Baso Minangkabau
Nederlands
नेपाली
□□□
Napulitano
Norsk
ଓଡ଼ିଆ
ਪੰਜਾਬੀ
پښتو
Polski
Português
Română
Русиньскый
Русский
Scots
Shqip
Sicilianu
සිංහල
Simple English
سنڌي
Slovenčina
Slovenščina
کوردی
Српски / srpski
Srpskohrvatski / српскохрватски
Suomi
Svenska
தமிழ்
Татарча/tatarça
ไทย
Тоҷикӣ
Türkçe
Türkmençe
Українська
اردو
Tiếng Việt
ייִדיש
□□
□□
Edit links

Whether and how quickly material should be initially removed for not having an inline citation to a reliable source depends on the material and the overall state of the article. In some cases, editors may object if you remove material without giving them time to provide references; consider adding a citation needed tag as an interim step.[4] When tagging or removing material for lacking an inline citation, please state your concern that it may not be possible to find a published reliable source for the content, and therefore it may not be verifiable.[5] If you think the material is verifiable, you are encouraged to provide an inline citation yourself before considering whether to remove or tag it.

Do *not* leave unsourced or poorly sourced material in an article if it might damage the reputation of living people[6] or existing groups, and do not move it to the talk page. You should also be aware of how the BLP policy applies to groups.

Reliable sources

"WP:SOURCE" redirects here. For the `<source>` *tag, see Wikipedia:Syntax highlighting. For how to reference sources, see Help:Referencing for beginners.*

Policy shortcuts
WP:SOURCE
WP:SOURCES

What counts as a reliable source

Further information: Wikipedia:Identifying reliable sources

The word "source" *when citing sources on Wikipedia* has three related meanings:

- The piece of work itself (the article, book)
- The creator of the work (the writer, journalist)
- The publisher of the work (for example, Random House or Cambridge University Press)

All three can affect reliability.

Articles must be based on reliable, third-party, published sources with a reputation for fact-checking and accuracy. Source material must have been **published**, the definition of which for our purposes is "made available to the public in some form".[7] **Unpublished** materials are not considered reliable. Use sources that directly support the material presented in an article and are appropriate to the claims made. The appropriateness of any source depends on the context. The best sources have a professional structure in place for checking or analyzing facts, legal issues, evidence, and arguments. The greater the degree of scrutiny given to these issues, the more reliable the source. Be especially careful when sourcing content related to living people or medicine.

If available, academic and peer-reviewed publications are usually the most reliable sources, such as in history, medicine, and science.

Editors may also use material from reliable non-academic sources, particularly if it appears in respected mainstream publications. Other reliable sources include:

- University-level textbooks
- Books published by respected publishing houses
- Magazines
- Journals
- Mainstream newspapers

Editors may also use electronic media, subject to the same criteria. See details in *Wikipedia:Identifying reliable sources* and *Wikipedia:Search engine test*.

Newspaper and magazine blogs

Policy shortcut
WP:NEWSBLOG

Several newspapers, magazines, and other news organizations host columns on their web sites that they call blogs. These may be acceptable sources if the writers are professionals, but use them with caution because the blog may not be subject to the news organization's normal fact-checking process.[8] If a news organization publishes an opinion piece in a blog, attribute the statement to the writer (e.g. "Jane Smith wrote..."). Never use as sources the blog comments that are left by readers. For personal or group blogs that are *not* reliable sources, see Self-published sources below.

Reliable sources noticeboard and guideline

Further information: Wikipedia:Reliable sources/Noticeboard and Wikipedia:Reliable sources

To discuss the reliability of a specific source for a particular statement, consult the reliable sources noticeboard, which seeks to apply this policy to particular cases. For a guideline discussing the reliability of particular *types* of sources, see Wikipedia:Reliable sources (WP:RS). In the case of inconsistency between this policy and the WP:RS guideline, or any other guideline related to sourcing, this policy has priority.

Sources that are usually not reliable

See also: Wikipedia:Identifying reliable sources § Questionable and self-published sources, and Wikipedia:Identifying reliable sources/Perennial sources

Questionable sources

Policy shortcuts
WP:NOTRELIABLE
WP:NOTRS
WP:QS

Questionable sources are those that have a poor reputation for checking the facts, lack meaningful editorial oversight, or have an apparent conflict of interest.[9]

Such sources include websites and publications expressing views that are widely considered by other sources to be extremist or promotional, or that rely heavily on unsubstantiated gossip, rumor or personal opinion. Questionable sources should only be used as sources for material on *themselves*, such as in articles about themselves; see below. They are not suitable sources for contentious claims about others.

Predatory open access journals are also questionable, due to lack of effective peer-review.

Self-published sources

Further information: Wikipedia:Biographies of living persons § Avoid self-published sources, Wikipedia:List of self-publishing companies, and Wikipedia:Identifying and using self-published works

Policy shortcuts
WP:SPS
WP:SELFPUBLISH
WP:BLOGS
WP:UGS

Anyone can create a personal web page or publish their own book, and also claim to be an expert in a certain field. For that reason, self-published media, or user generated sources, such as books, patents, newsletters, personal websites, open wikis, personal or group blogs (as distinguished from newsblogs, above), content farms, Internet forum postings, and social media postings, are largely not acceptable as sources. Self-published expert sources may be considered reliable when produced by an established expert on the subject matter, whose work **in the relevant field** has previously been published by reliable third-party publications.[8] Exercise caution when using such sources: if the information in question is suitable for inclusion, someone else will probably have published it in independent reliable sources.[10] **Never** use self-

published sources as third-party sources about living people, even if the author is an expert, well-known professional researcher, or writer.

Self-published or questionable sources as sources on themselves

"WP:SOCIALMEDIA" redirects here. For the policy on what Wikipedia is not, see WP:NOTSOCIALNETWORK.
"WP:TWITTER" redirects here. For the external links essay, see WP:Twitter-EL.

Policy shortcuts
WP:ABOUTSELF
WP:SELFPUB
WP:TWITTER
WP:SOCIALMEDIA

Self-published and questionable sources may be used as sources of information **about themselves**, usually in articles about themselves or their activities, without the self-published source requirement that they be published experts in the field, so long as:

1. the material is neither unduly self-serving nor an exceptional claim;
2. it does not involve claims about third parties;
3. it does not involve claims about events not directly related to the source;
4. there is no reasonable doubt as to its authenticity; and
5. the article is not based primarily on such sources.

This policy also applies to material published by the subject on social networking websites such as Twitter, Tumblr, Reddit, and Facebook.

Wikipedia and sources that mirror or use it

"WP:CIRCULAR" redirects here. For links on a page that redirect back to the same page, see WP:SELFRED.
See also: WP:COPYWITHIN, Wikipedia:List of citogenesis incidents, and Wikipedia:Citing Wikipedia

Policy shortcuts
WP:CIRC
WP:CIRCULAR
WP:REFLOOP

Do not use articles from Wikipedia (whether this English Wikipedia or Wikipedias in other languages) as sources. Also, do not use websites that mirror Wikipedia content or publications that rely on material from Wikipedia as sources. Content from a Wikipedia article is not considered reliable unless it is backed up by citing reliable sources. Confirm that these sources support the content, then use them directly.[11] (There is also a risk of circular reference/circular reporting when using a Wikipedia article or derivative work as a source.)

An exception is allowed when Wikipedia itself is being discussed in the article, which may cite an article, guideline, discussion, statistic, or other content from Wikipedia (or a sister project) to support a statement about Wikipedia. Wikipedia or the sister project is a primary source in this case, and may be used following the policy for primary sources. Any such use should avoid original research, undue emphasis on Wikipedia's role or views, and inappropriate self-reference. The article text should make it clear that the material is sourced from Wikipedia so the reader is made aware of the potential bias.

Accessibility

Access to sources

See also: Wikipedia:Offline sources, Wikipedia:WikiProject Resource Exchange/Resource Request, and Wikipedia:Reliable sources/Cost

Policy shortcuts
WP:PAYWALL
WP:SOURCEACCESS

Some reliable sources may not be easily accessible. For example, an online source may require payment, and a print-only source may be available only in university libraries. Rare historical sources may even be available only in special museum collections and archives. Do not reject reliable sources just because they are difficult or costly to access. If you have trouble accessing a source, others may be able to do so on your behalf (see WikiProject Resource Exchange).

Non-English sources

See also: Wikipedia:Translators available and Wikipedia:No original research § Translations and transcriptions

Policy shortcuts
WP:RSUE
WP:NOENG
WP:NONENG

Citing

Citations to non-English reliable sources are allowed on the English Wikipedia. However, because this project is in English, English-language sources are preferred over non-English ones when available and of equal quality and relevance. As with sources in English, if a dispute arises involving a citation to a non-English source, editors may request that a quotation of relevant portions of the original source be provided, either in text, in a footnote, or on the article talk page.[12] (See Template:Request quotation.)

Quoting

If you quote a non-English reliable source (whether in the main text or in a footnote), a translation into English should always accompany the quote. Translations published by reliable sources are preferred over translations by Wikipedians, but translations by Wikipedians are preferred over machine translations. When using a machine translation of source material, editors should be reasonably certain that the translation is accurate and the source is appropriate. Editors should not rely upon machine translations of non-English sources in contentious articles or biographies of living people. If needed, ask an editor who can translate it for you.

In articles, the original text is usually included with the translated text when translated by Wikipedians, and the translating editor is usually not cited. When quoting any material, whether in English or in some other language, be careful not to violate copyright; see the fair-use guideline.

Other issues

Verifiability does not guarantee inclusion

"WP:ONUS" redirects here. For the responsibility to demonstrate verifiability, see WP:BURDEN.
See also: WP:UNDUE, WP:PAGEDECIDE, WP:PRESERVE, WP:SUMMARY, and WP:IINFO

Shortcuts
WP:VNOTSUFF
WP:ONUS

While information must be verifiable in order to be included in an article, this does not mean that all verifiable information must be included in an article. Consensus may determine that certain information does not improve an article, and that it should be omitted or presented instead in a different article. The onus to achieve consensus for inclusion is on those seeking to include disputed content.

Tagging a sentence, section, or article

Further information: Wikipedia:Citation needed and Wikipedia:Template messages/Sources of articles

Shortcut
WP:FAILEDVERIFICATION

If you want to request a source for an unsourced statement, you can tag a sentence with the {{ citation needed}} template by writing {{cn}} or {{fact}}. There are other templates here for tagging sections or entire articles. You can also leave a note on the talk page asking for a source, or move the material to the talk page and ask for a source there. To request verification that a reference supports the text, tag it with {{verification needed}}. Material that fails verification may be tagged with {{failed verification}} or removed. When using templates to tag material, it is helpful to other editors if you explain your rationale in the template, edit summary, or on the talk page.

Take special care with material about living people. Contentious material about living people that is unsourced or poorly sourced should be removed

immediately, not tagged or moved to the talk page.

Exceptional claims require exceptional sources

See also: Wikipedia:Fringe theories

Policy shortcuts
WP:REDFLAG
WP:EXCEPTIONAL
WP:EXTRAORDINARY

Any exceptional claim requires *multiple* high-quality sources.[13] Red flags that should prompt extra caution include:

- surprising or apparently important claims not covered by multiple mainstream sources;
- challenged claims that are supported purely by primary or self-published sources or those with an apparent conflict of interest;[9]
- reports of a statement by someone that seems out of character, or against an interest they had previously defended;
- claims that are contradicted by the prevailing view within the relevant community, or that would significantly alter mainstream assumptions, especially in science, medicine, history, politics, and biographies of living people. This is especially true when proponents say there is a conspiracy to silence them.

Verifiability and other principles

Copyright and plagiarism

Further information: Wikipedia:Copyright, Wikipedia:Plagiarism, Wikipedia:Copying within Wikipedia, Wikipedia:MOS § Attribution, and Wikipedia:CITE § In-text attribution

Policy shortcut
WP:YTCOPYRIGHT

Do not plagiarize or breach copyright when using sources. Summarize source material in your own words as much as possible; when quoting or closely paraphrasing a source use an inline citation, and in-text attribution where appropriate.

Do not link to any source that violates the copyrights of others per contributors' rights and obligations. You can link to websites that display copyrighted works as long as the website has licensed the work, or uses the work in a way compliant with fair use. Knowingly directing others to material that violates copyright may be considered contributory copyright infringement. If there is reason to think a source violates copyright, do not cite it. *This is particularly relevant when linking to sites such as Scribd or YouTube, where due care should be taken to avoid linking to material that violates copyright.*

Neutrality

Further information: Wikipedia:Neutral point of view

Even when information is cited to reliable sources, you must present it with a neutral point of view (NPOV). Articles should be based on thorough research of sources. All articles must adhere to NPOV, fairly representing all majority and significant-minority viewpoints published by reliable sources, in rough proportion to the prominence of each view. Tiny-minority views need not be included, except in articles devoted to them. If there is disagreement between sources, use in-text attribution: "John Smith argues that X, while Paul Jones maintains that Y," followed by an inline citation. Sources themselves do not need to maintain a neutral point of view. Indeed, many reliable sources are *not* neutral. Our job as editors is simply to summarize what the reliable sources say.

Notability

Further information: Wikipedia:Notability

If no reliable third-party sources can be found on a topic, Wikipedia should not have an article on it.

Original research

Further information: Wikipedia:No original research

The "No original research" policy (NOR) is closely related to the Verifiability policy. Among its requirements are:

1. All material in Wikipedia articles must be *attributable* to a reliable published source. This means that a reliable published source must exist for it, whether or not it is cited in the article.
2. Sources must support the material clearly and directly: drawing inferences from multiple sources to advance a novel position is prohibited by the NOR policy.[12]
3. Base articles largely on reliable secondary sources. While primary sources are appropriate in some cases, relying on them can be problematic. For more information, see the Primary, secondary, and tertiary sources section of the NOR policy, and the Misuse of primary sources section of the BLP policy.

See also

Guidelines

- Reliable sources
- Identifying reliable sources (medicine)

Information pages

- Wikipedia is not a reliable source
- Core content policies
- How to mine a source
- Identifying and using independent sources
- Identifying and using primary sources
- Video links

Resources

- Improving referencing efforts
- Template messages/Sources of articles
- WikiProject Reliability

Essays

- Citation clutter
- Identifying and using tertiary sources

- Verifiability, not truth

Notes

1. ^ This principle was previously expressed on this policy page as "the threshold for inclusion is **verifiability, not truth**." See the essay, WP:Verifiability, not truth.
2. ^ A source "directly supports" a given piece of material if that information is directly present in the source, so that using this source to support this material is not a violation of Wikipedia:No original research. The location of any citation – including whether one is present in the article at all – is unrelated to whether a source directly supports the material. For questions about where and how to place citations, see WP:CITE, WP:CITELEAD, etc.
3. ^ Once an editor has provided any source that he or she believes, in good faith, to be sufficient, then any editor who later removes the material has an obligation to articulate specific problems that would justify its exclusion from Wikipedia (e.g., why the source is unreliable; the source does not support the claim; undue emphasis; unencyclopedic content; etc.). If necessary, all editors are then expected to help achieve consensus, and any problems with the text or sourcing should be fixed before the material is added back.
4. ^ It may be that the article contains so few citations that it is impractical to add specific citation needed tags, in which case consider tagging a section with {{unreferencedsection}}, or the article with {{refimprove}} or {{unreferenced}}. In the case of a disputed category or on a disambiguation page, consider asking for a citation on the talk page.
5. ^ When tagging or removing such material, please keep in mind that such edits can be easily misunderstood. Some editors object to others making chronic, frequent, and large-scale deletions of unsourced information, especially if unaccompanied by other efforts to improve the material. Do not concentrate only on material of a particular POV, as that may result in accusations that you are in violation of WP:NPOV. Also check to see whether the material is sourced to a citation elsewhere on the page. For all of these reasons, it is advisable to communicate clearly that you have a considered reason to believe that the material in question cannot be verified.
6. ^ Wales, Jimmy. "Zero information is preferred to misleading or false information", WikiEN-l, May 16, 2006: "I can NOT emphasize this enough. There seems to be a terrible bias among some editors that some sort of random speculative 'I heard it somewhere' pseudo information is to be tagged with a 'needs a cite' tag. Wrong. It should be removed, aggressively, unless it can be sourced. This is true of all information, but it is particularly true of negative information about living persons."
7. ^ This includes material such as documents in publicly accessible archives, inscriptions on monuments, gravestones, etc., that are available for anyone to see.
8. ^ [a] [b] Please do note that any exceptional claim would require exceptional sources.
9. ^ [a] [b] Sources that may have interests other than professional considerations in the matter being reported are considered to be conflicted sources. Further examples of sources with conflicts of interest include but are not limited to articles by any media group that promote the holding company of the media group or discredit its competitors; news reports by journalists having financial interests in the companies being reported or in their competitors; material (including but not limited to news reports, books, articles and other publications) involved in or struck down by litigation in any country, or released by parties involved in litigation against other involved parties, during, before or after the litigation; and promotional material released through media in the form of paid news reports. For definitions of sources with conflict of interest:
   - The Columbia Center for New Media Teaching and Learning, Columbia University mentions: "A conflict of interest involves the abuse – actual, apparent, or potential – of the trust that people have in professionals. The simplest working definition states: A conflict of interest is a situation in which financial or other personal considerations have the potential to compromise or bias professional judgment and objectivity. An apparent conflict of interest is one in which a reasonable person would think that the professional's judgment is likely to be compromised. A potential conflict of interest involves a situation that may develop into an actual conflict of interest. It is important to note that a conflict of interest exists whether or not decisions are affected by a personal interest; a conflict of interest implies only the potential for bias, not a likelihood. It is also important to note that a conflict of interest is not considered misconduct in research, since the definition for misconduct is currently limited to fabrication, falsification, and plagiarism."
   - The New York Times Company forwards this understanding: "Conflicts of interest, real or apparent, may come up in many areas. They may involve the relationships of staff members with readers, news sources, advocacy groups, advertisers, or competitors; with one another, or with the newspaper or its parent company. And at a time when two-career families are the norm, the civic and professional activities of spouses, family and companions can create conflicts or the appearance of conflicts."
10. ^ Self-published material is characterized by the *lack of independent reviewers* (those without a conflict of interest) validating the reliability of content. Further examples of self-published sources include press releases, material contained within company websites, advertising campaigns, material published in media by the owner(s)/publisher(s) of the media group, self-released music albums and electoral manifestos:
    - The University of California, Berkeley library states: "Most pages found in general search engines for the web are self-published or published by businesses small and large with motives to get you to buy something or believe a point of view. Even within university and library web sites, there can be many pages that the institution does not try to oversee."
    - Princeton University offers this understanding in its publication, *Academic Integrity at Princeton (2011)*: "Unlike most books and journal articles, which undergo strict editorial review before publication, much of the information on the Web is self-published. To be sure, there are many websites in which you can have confidence: mainstream newspapers, refereed electronic journals, and university, library, and government collections of data. But for vast amounts of Web-based information, no impartial reviewers have evaluated the accuracy or fairness of such material before it's made instantly available across the globe."
    - The Chicago Manual of Style, 16th Edition [*dead link*] states, "any Internet site that does not have a specific publisher or sponsoring body should be treated as unpublished or self-published work."
11. ^ Rekdal, Ole Bjørn (1 August 2014). "Academic urban legends". *Social Studies of Science*. **44** (4): 638–654. doi:10.1177/0306312714535679. ISSN 0306-3127. Retrieved 30 April 2016.
12. ^ [a] [b] When there is dispute about whether a piece of text is fully supported by a given source, direct quotes and other relevant details from the source should be provided to other editors as a courtesy. Do not violate the source's copyright when doing so.
13. ^ Hume, David. *An Enquiry concerning Human Understanding*, Forgotten Books, 1984, pp. 82, 86; first published in 1748 as *Philosophical enquiries concerning human Understanding*, (or the Oxford 1894 edition OL 7067396M at para. 91) "A wise man ... proportions his belief to the evidence. ... That no testimony is sufficient to establish a miracle, unless the testimony be of such a kind, that its falsehood would be more miraculous, than the fact, which it endeavours to establish; and even in that case there is a mutual destruction of arguments, and the superior only gives us an assurance suitable to that degree of force, which remains, after deducting the inferior." In the 18th century, Pierre-Simon Laplace reformulated the idea as "The weight of evidence for an extraordinary claim must be proportioned to its strangeness." Marcello Truzzi recast it again, in 1978, as "An extraordinary claim requires extraordinary proof." Carl Sagan, finally, popularized the concept broadly as "Extraordinary claims require extraordinary evidence" in 1980 on *Cosmos: A Personal Voyage*; this wa the formulation originally used on Wikipedia.

Further reading

- Wales, Jimmy. "Insist on sources", WikiEN-l, July 19, 2006: "I really want to encourage a much stronger culture which says: it is better to have no information, than to have information like this, with no sources."—referring to a rather unlikely statement about the founders of Google throwing pies at each other.

v · t · e	Wikipedia referencing
Policies and guidelines	**Verifiability** · Biographies of living persons · Reliable sources (Medicine) · Citing sources · Scientific citations
General advice	Citation needed · Find sources · Combining sources · Offline sources · Referencing styles
Citing sources	Citation Style 1 · Citation Style 2 · Citation Style Vancouver · LSA · Comics · Citation templates · Reflist template
Inline citations	Footnotes · Parenthetical referencing · Punctuation and footnotes · Shortened footnotes · Nesting footnotes
Help for beginners	Reference-tags · Citations quick reference · Introduction to referencing · Referencing with citation templates · Referencing without using templates · Referencing dos and don'ts · Citing Wikipedia
Advanced help	Cite link labels · Citation tools · Cite errors · Cite messages · Converting between references formats · Reference display customization · References and page numbers
Template documentation	{{Edit refs}} · {{Refref}} · {{Refref2}} · {{Refstart}}
Tools	Wikipedia Library

v · t · e		Wikipedia key policies and guidelines
		Five pillars (What Wikipedia is not · Ignore all rules)
Content	✓	**Verifiability** · No original research · Neutral point of view · What Wikipedia is not · Biographies of living persons · Image use · Article titles
	✓	Notability · Autobiography · Citing sources · Reliable sources (medicine) · Do not include copies of lengthy primary sources · Plagiarism · Don't create hoaxes · Fringe theories · Patent nonsense · External links · Portal namespace
Conduct	✓	Civility · Consensus · Editing policy · Harassment · Vandalism · Ignore all rules · No personal attacks · Ownership of content · Edit warring · Dispute resolution · Sock puppetry · No legal threats · Child protection · Paid-contribution disclosure
	✓	Assume good faith · Conflict of interest · Disruptive editing · Do not disrupt Wikipedia to illustrate a point · Etiquette · Gaming the system · Please do not bite the newcomers · Courtesy vanishing · Responding to threats of harm
Deletion	✓	Deletion policy · Proposed deletion (Biographies · Books) · Criteria for speedy deletion · Attack page · Oversight · Revision deletion
Enforcement	✓	Administrators · Banning · Blocking · Page protection

Editing	✓	Article size · Be bold · Disambiguation · Hatnotes · Talk page guidelines (Signatures) · Broad-concept article
		Style: Manual of Style (Contents) · Accessibility (Understandability) · Dates and numbers · Images · Layout · Lead section · Linking · Lists
		Classification: Categories, lists, and navigation templates · Categorization · Template namespace
Project content	✓	Project namespace (WikiProjects) · User pages (User boxes) · Shortcuts · Subpages
WMF	✓	List of policies · Friendly space policy · Licensing and copyright · Privacy policy · Values · FAQ
List of all policies and guidelines (✓ List of policies · ✓ List of guidelines) · Lists of attempts in creating fundamental principles		

Categories: Wikipedia policies | Wikipedia content policies | Wikipedia verifiability

This page was last edited on 25 February 2019, at 09:55 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Cookie statement Mobile view

Wikipedia:Content assessment

From Wikipedia, the free encyclopedia

"WP:CLASSES" redirects here. For the catalogue of CSS classes, see WP:CLASS.
"WP:ASSESSMENT" redirects here. For a more general overview of assessment at Wikipedia, see Wikipedia:Assessment. For the original, superseded assessment process, see Wikipedia:Article assessment (historical).

This page documents an English Wikipedia editing guideline.
It is a generally accepted standard that editors should attempt to follow, though it is best treated with common sense, and occasional exceptions may apply. Any substantive edit to this page should reflect consensus. When in doubt, discuss first on the talk page.

Shortcuts
WP:ASSESS
WP:1.0/A

It has been suggested that *Wikipedia:WikiProject Council/Assessment FAQ* be merged into this page. (Discuss) *Proposed since June 2018.*

The following system is used by the Wikipedia:Version 1.0 Editorial Team for assessing how close we are to a distribution-quality article on a particular topic. The system is based on a letter scheme which reflects principally how factually complete the article is, though language quality and layout are also factors. Once an article reaches the **A-Class**, it is considered "complete", although edits will continue to be made.

The quality assessments are mainly performed by members of WikiProjects, who tag talk pages of articles. These tags are then collected by a bot, which generates output such as a log and statistics. For more information, see *Using the bot*. (Note that when more than one WikiProject has rated an article, the bot will take the best rating as the rating of the overall article.) The WP:1.0 team is now setting up to use a *second* bot to select articles, based on the assessments performed by WikiProjects.

Two levels, GA (Good Article) and FA (Featured Article), are assessments made by independent editors, rather than by WikiProjects. GAs are generally reviewed by a single editor, and FA by a panel. Candidates are nominated by listing them at *WP:Good article nominations* and *WP:Featured article candidates*. Judgments are made according to the criteria at *WP:Good article criteria* and *WP:Featured article criteria*, and the results are listed at *WP:Good articles* and *WP:Featured articles*.

It is vital that editors not take these assessments of their contributions personally. It is understood that we each have our own opinions of the priorities of the objective criteria for a perfect article. Generally an active project will develop a consensus, though be aware that different projects may use their own variation of the criteria more tuned for the subject area, such as this. More active WikiProjects have an assessment team. If you contribute a lot of content to an article you may request an independent assessment.

At present this assessment system is in use in the Wikipedia 1.0 project, and in several hundred WikiProjects on the English Wikipedia. As of May 2017, over 5.1 million articles have been assessed. Several other languages are also using this assessment system or a derivative thereof.

Contents [hide]

Grades [edit]

WikiProject article quality grading scheme v · t · e

Class	Criteria	Reader's experience	Editing suggestions	Example
	The article has attained featured article status by passing an in-depth examination by impartial reviewers from WP:Featured article candidates. **More detailed criteria** The article meets the **featured article criteria**: A featured article exemplifies our very best work and is distinguished by professional			

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Български
Dansk
Ελληνικά
Euskara
فارسی
Français
Galego
□□□
Italiano
Македонски
नेपाली
Norsk
پنجابی
Português
Română
Русский
Suomi
Svenska
தமிழ்
Türkçe
□□
Edit links

	★ FA	standards of writing, presentation, and sourcing. In addition to meeting the policies regarding content for all Wikipedia articles, it has the following attributes. 1. It is: a. **well-written**: its prose is engaging and of a professional standard; b. **comprehensive**: it neglects no major facts or details and places the subject in context; c. **well-researched**: it is a thorough and representative survey of the relevant literature; claims are verifiable against high-quality reliable sources and are supported by inline citations where appropriate; d. **neutral**: it presents views fairly and without bias; and e. **stable**: it is not subject to ongoing edit wars and its content does not change significantly from day to day, except in response to the featured article process. 2. It follows the style guidelines, including the provision of: a. **a lead**: a concise lead section that summarizes the topic and prepares the reader for the detail in the subsequent sections; b. **appropriate structure**: a substantial but not overwhelming system of hierarchical section headings; and c. **consistent citations**: where required by criterion 1c, consistently formatted inline citations using either footnotes (<ref>Smith 2007, p. 1</ref>) or Harvard referencing (Smith 2007, p. 1)—see citing sources for suggestions on formatting references. Citation templates are not required. 3. **Media.** It has images and other media, where appropriate, with succinct captions and acceptable copyright status. Images follow the image use policy. Non-free images or media must satisfy the criteria for inclusion of non-free content and be labeled accordingly. 4. **Length.** It stays focused on the main topic without going into unnecessary detail and uses summary style.	Professional, outstanding, and thorough; a definitive source for encyclopedic information.	No further content additions should be necessary unless new information becomes available; further improvements to the prose quality are often possible.	Cleopatra (as of June 2018)
		The article is well organized and essentially complete, having been examined by impartial reviewers from a WikiProject or elsewhere.			

A	Good article status is not a requirement for A-Class. **More detailed criteria** The article meets the **A-Class criteria**: Provides a well-written, clear and complete description of the topic, as described in Wikipedia:Article development. It should be of a length suitable for the subject, appropriately structured, and be well referenced by a broad array of reliable sources. It should be well illustrated, with no copyright problems. Only minor style issues and other details need to be addressed before submission as a featured article candidate. See the A-Class assessment departments of some of the larger WikiProjects (e.g. WikiProject Military history).	Very useful to readers. A fairly complete treatment of the subject. A non-expert in the subject would typically find nothing wanting.	Expert knowledge may be needed to tweak the article, and style problems may need solving. WP:Peer review may help.	Battle of Nam River (as of June 2014)
GA	The article has attained good article status having been examined by one or more impartial reviewers from WP:Good article nominations. **More detailed criteria** The article meets the **good article criteria**: A good article is: 1. **Well written**: a. the prose is clear and concise, and the spelling and grammar are correct; and b. it complies with the manual of style guidelines for lead sections, layout, words to watch, fiction, and list incorporation. 2. **Verifiable** with **no original research**: a. it contains a list of all references (sources of information), presented in accordance with the layout style guideline; b. all inline citations are from reliable sources, including those for direct quotations, statistics, published opinion, counter-intuitive or controversial statements that are challenged or likely to be challenged, and contentious material relating to living persons—science-based articles should follow the scientific citation guidelines; c. it contains no original research; and d. it contains no copyright violations nor plagiarism. 3. **Broad in its coverage**: a. it addresses the main aspects of the topic; and b. it stays focused on the topic without going into unnecessary detail (see summary style). 4. **Neutral**: it represents viewpoints fairly and without editorial bias, giving due	Useful to nearly all readers, with no obvious problems; approaching (but not equalling) the quality of a professional encyclopedia.	Some editing by subject and style experts is helpful; comparison with an existing featured article on a similar topic may highlight areas where content is weak or missing.	Discovery of the neutron (as of December 2017)

	weight to each. 5. **Stable**: it does not change significantly from day to day because of an ongoing edit war or content dispute. 6. **Illustrated**, if possible, by **media** such as images, video, or audio: a. media are tagged with their copyright statuses, and valid fair use rationales are provided for non-free content; and b. media are relevant to the topic, and have suitable captions.			
B	The article is mostly complete and without major problems, but requires some further work to reach good article standards. **More detailed criteria** The article meets the **six B-Class criteria**: 1. **The article is suitably referenced, with inline citations.** It has reliable sources, and any important or controversial material which is likely to be challenged is cited. Any format of inline citation is acceptable: the use of <ref> tags and citation templates such as {{cite web}} is optional. 2. **The article reasonably covers the topic, and does not contain obvious omissions or inaccuracies.** It contains a large proportion of the material necessary for an A-Class article, although some sections may need expansion, and some less important topics may be missing. 3. **The article has a defined structure.** Content should be organized into groups of related material, including a lead section and all the sections that can reasonably be included in an article of its kind. 4. **The article is reasonably well-written.** The prose contains no major grammatical errors and flows sensibly, but it does not need to be "brilliant". The Manual of Style does not need to be followed rigorously. 5. **The article contains supporting materials where appropriate.** Illustrations are encouraged, though not required. Diagrams, an infobox etc. should be included where they are relevant and useful to the content. 6. **The article presents its content in an appropriately understandable way.** It is written with as broad an audience in mind as possible. Although Wikipedia is more than just a general encyclopedia, the article should not assume unnecessary technical background and technical terms should be explained or avoided where possible.	Readers are not left wanting, although the content may not be complete enough to satisfy a serious student or researcher.	A few aspects of content and style need to be addressed. Expert knowledge may be needed. The inclusion of supporting materials should be considered if practical, and the article checked for general compliance with the Manual of Style and related style guidelines.	The Hague (as of June 2018)
	The article is substantial, but is still missing important content or contains much irrelevant			

C	material. The article should have some references to reliable sources, but may still have significant problems or require substantial cleanup. **More detailed criteria** The article cites more than one reliable source and is better developed in style, structure, and quality than Start-Class, but it fails one or more of the criteria for B-Class. It may have some gaps or missing elements; need editing for clarity, balance, or flow; or contain policy violations, such as bias or original research. Articles on fictional topics are likely to be marked as C-Class if they are written from an in-universe perspective. It is most likely that C-Class articles have a reasonable encyclopedic style.	Useful to a casual reader, but would not provide a complete picture for even a moderately detailed study.	Considerable editing is needed to close gaps in content and solve cleanup problems.	Wing (as of June 2018)
Start	An article that is developing, but which is quite incomplete. It might or might not cite adequate reliable sources. **More detailed criteria** The article has a usable amount of good content but is weak in many areas. Quality of the prose may be distinctly unencyclopedic, and Wikipedia:Manual of Style compliance non-existent. The article should satisfy fundamental content policies, such as Wikipedia:Biographies of living persons. Frequently, the referencing is inadequate, although enough sources are usually provided to establish verifiability. No Start-Class article should be in any danger of being speedily deleted.	Provides some meaningful content, but most readers will need more.	Providing references to reliable sources should come first; the article also needs substantial improvement in content and organisation. Also improve the grammar, spelling, writing style and improve the jargon use.	Ring-tailed cardinalfish (as of June 2018)
Stub	A very basic description of the topic. However, all very-bad-quality articles will fall into this category. **More detailed criteria** The article is either a very short article or a rough collection of information that will need much work to become a meaningful article. It is usually very short; but, if the material is irrelevant or incomprehensible, an article of any length falls into this category. Although Stub-class articles are the lowest class of the normal classes, they are adequate enough to be an accepted article, though they do have risks of being dropped from being an article altogether.	Provides very little meaningful content; may be little more than a dictionary definition. Readers probably see insufficiently developed features of the topic and may not see how the features of the topic are significant.	Any editing or additional material can be helpful. The provision of meaningful content should be a priority. The best solution for a Stub-class Article to step up to a Start-class Article is to add in referenced reasons of why the topic is significant.	Crescent Falls (as of June 2018)
	The article has attained featured list status. **More detailed criteria** The article meets the **featured list criteria**: 1. **Prose.** It features professional standards of writing. 2. **Lead.** It has an engaging lead that introduces the subject and defines the scope and inclusion criteria. 3. **Comprehensiveness.** • (a) It comprehensively covers the defined scope, providing at least all of the major items and, where practical, a complete set of items; where appropriate, it has annotations that provide useful and			

FL	appropriate information about the items. • (b) In length and/or topic, it meets all of the requirements for stand-alone lists; does not violate the content-forking guideline, does not largely duplicate material from another article, and could not reasonably be included as part of a related article. 4. **Structure.** It is easy to navigate and includes, where helpful, section headings and table sort facilities. 5. **Style.** It complies with the Manual of Style and its supplementary pages. • (a) *Visual appeal.* It makes suitable use of text layout, formatting, tables, and colour; and a minimal proportion of items are redlinked. • (b) *Media files.* It has images and other media, if appropriate to the topic, that follow Wikipedia's usage policies, with succinct captions. Non-free images and other media satisfy the criteria for the inclusion of non-free content and are labeled accordingly 6. **Stability.** It is not the subject of ongoing edit wars and its content does not change significantly from day to day, except in response to the featured list process.	Professional standard; it comprehensively covers the defined scope, usually providing a complete set of items, and has annotations that provide useful and appropriate information about those items.	No further content additions should be necessary unless new information becomes available.	List of dates predicted for apocalyptic events (as of May 2018)
List	Meets the criteria of a stand-alone list, which is an article that contains primarily a list, usually consisting of links to articles in a particular subject area.	There is no set format for a list, but its organization should be logical and useful to the reader.	Lists should be lists of live links to Wikipedia articles, appropriately named and organized.	List of Guggenheim Fellowships awarded in 1947 (as of June 2018)

Note: Some WikiProjects omit some of the standard classes, most often A-class, especially when they lack an assessment team.

Non-standard grades [edit]

Some WikiProjects use a few other assessments for the main namespace that do not fit into the scale. The more popular assessments, in no particular order:

Shortcut
WP:NONSTANDARDCLASS

Other WikiProject assessments

Label	Criteria	Reader's experience	Editor's experience	Example
Future {{Future-Class}}	A topic where details are subject to change often. The article covers a future topic of which no broadcasted version exists so far and all information is subject to change when new information arises from reliable sources. With multiple reliable sources there might be information that contradicts other information in the same or other articles.	Amount of meaningful content varies over time as the projected event draws near.	Material added might be speculation and should be carefully sourced.	Preston City Council election, 2014 (as of September 2013)
Current {{Current-Class}}	A topic where details are subject to change often. The article covers an event or topic that is currently going on, such as a football game or a sports team's season.	Amount of meaningful content varies over time as the projected event goes on.	Material added might quickly become obsolete.	2017 Atlantic hurricane season (as of June 2017)
			Additions	

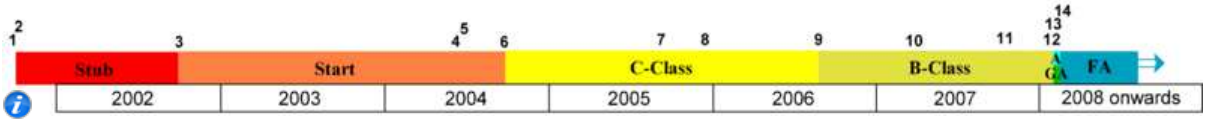
Disambig {{Disambig-Class}}	Any disambiguation page falls under this class.	The page directs the reader to other pages of the same title.	should be made as new articles of that name are created.	Aa River (as of June 2008)
NA {{NA-Class}}	Any non-article page that does not fit into any other category.	The page does not have article content.	May or may not apply, depending on the type of page.	any WikiProject's internal resources
Redirect {{Redirect-Class}}	Any redirect falls under this class.			Collapse Into Now
Book {{Book-Class}}	Any Wikipedia book falls under this class.			Book:Canada
Template {{Template-Class}}	Any template falls under this class.			Template:Magnapop
Category {{Category-Class}}	Any category falls under this class.			Category:George Orwell
Draft {{Draft-Class}}	Any draft falls under this class.			Draft:Example

Some WikiProjects use additional grades not listed above, such as those used at WP:Comics. Most common are "Image" and "Needed". See the relevant Assessment page for the WikiProject, at Category:WikiProject assessments.

Evolution of an article – an example [edit]

See also: outreach:Life of an Article

This clickable imagemap, using the article "Atom" as an example, demonstrates the typical profile for an article's development through the levels. Hold the mouse over a number to see key events, and **click on a number** to see that version of the article. Please note that until 2008, a C-class rating did not exist on the project, and as such this grading is retroactive. Also, in 2006 references were much less used, and inline references were quite rare; a barely-B-Class article today would typically have many more references than this article did in late 2006.



Importance assessment [edit]

There is a separate scale for rating articles for *importance* or *priority*, which is unrelated to the *quality* scale outlined here. Unlike the quality scale, the priority scale varies based on the project scope. See also a template at {{Importance scheme}}.

Statistics [edit]

The WP 1.0 bot tracks assessment data (article quality and importance data for individual WikiProjects) assigned via talk page banners. If you would like to add a new WikiProject to the bot's list, please read the instructions at Wikipedia:Version 1.0 Editorial Team/Using the bot.

The global summary table below is computed by taking the highest quality and importance rating for each assessed article in the main namespace.

All rated articles by quality and importance						
Quality	Importance					
	Top	High	Mid	Low	???	Total
FA	1,291	1,967	1,902	1,267	182	**6,609**
FL	141	539	609	522	109	**1,920**
A	251	472	631	419	90	**1,863**
GA	2,383	5,377	10,632	12,109	1,813	**32,314**

B	13,010	24,877	38,313	33,297	15,559	**125,056**
C	11,762	35,233	79,925	118,698	52,486	**298,104**
Start	18,093	82,348	337,118	961,718	346,298	**1,745,575**
Stub	4,336	31,917	245,528	2,138,849	886,422	**3,307,052**
List	3,456	12,749	39,732	115,462	74,368	**245,767**
Assessed	54,723	195,479	754,390	3,382,341	1,377,327	**5,764,260**
Unassessed	130	575	1,770	18,324	516,296	**537,095**
Total	**54,853**	**196,054**	**756,160**	**3,400,665**	**1,893,623**	**6,301,355**

About this table

See also [edit]

- Wikipedia:Article assessment, the previous version superseded by this version.
- Wikipedia:Assessing articles, an essay on the criteria and purpose of article assessments
- Wikipedia:Metadata gadget, a script (and gadget) that finds articles' assessment information from the talk page and puts it in the article's header.
- User:Kephir/gadgets/rater, a script for tagging articles' talk pages with assessment information
- User:N8wilson/AQFetcher, a script that stylizes links on Wikipedia according to the assessed quality of the target article.
- mw:Article feedback, an initiative of the Wikimedia Foundation to engage Wikimedia readers in the assessment of article quality, one of the five priorities defined in the strategic plan
- Wikipedia:Data mining Wikipedia, a potential use of WikiProject assessments

Categories: Wikipedia editing guidelines | Wikipedia 1.0 assessments | Wikipedia project content guidelines | Wikipedia article assessment

This page was last edited on 19 February 2019, at 00:25 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Cookie statement Mobile view

Wikipedia: Your first article

From Wikipedia, the free encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item

Print/export

Create a book
Download as PDF
Printable version

Languages

Afrikaans
العربية
বাংলা
Bân-lâm-gú
भोजपुरी
Български
Català
Čeština
Chavacano de Zamboanga
Cymraeg
Dansk
Ελληνικά
Español
Esperanto

Writing an article

Learn how you can create an article.

This page is not a sandbox. It should not be used for test editing. To experiment, you can use the shared **sandbox** – or if you're logged in, **your personal sandbox**.

This page in a nutshell: Wikipedia articles follow certain guidelines: the topic should be notable and be covered in detail in good references from independent sources. Wikipedia is an encyclopedia – it is not a personal home page or a business list. Do not copy-paste content from other websites even if you, your school, or your boss owns them. If you choose to create the article with only a limited knowledge of the standards here, you should be aware that other editors may delete it if it's not considered appropriate. To create full articles (as opposed to draft pages), your account must be at least 4 days (96 hours) old, and you must have made more than ten edits. For information on how to request a new article that can be created by someone else, see Wikipedia:Requested articles. To create an article, you can try the Article Wizard.

Welcome to Wikipedia! You have probably already edited blogs or social media sites. You have made edits that improved existing articles and now you want to start a new

Shortcuts
WP:YFA
WP:FIRSTARTICLE

فارسی
Hausa
Հայերեն
हिन्दी
Hrvatski
Bahasa Indonesia
עברית
Basa Jawa
■■■■■
ქართული
Ladino
Magyar
मैथिली
Bahasa Melayu
■ □■ □■■■■■■
□□□
Norsk
ଓଡ଼ିଆ
ਪੰਜਾਬੀ
پښتو
Ripoarisch
Română
Русиньскый
Sicilianu
සිංහල
سنڌي
Slovenčina
Soomaaliga
کوردی
Српски / srpski
□□□■ □□■ □■
Basa Sunda
Suomi
Svenska
தமிழ்
ไทย
Türkçe
اردو
Tiếng Việt
□□
Беларуская (тарашкевіца)
Edit links

article from scratch. Now, if you have not done any Wikipedia editing, you cannot directly make a new article in mainspace. This permission is only for autoconfirmed users, which means those whose accounts are more than four days old and who have done at least ten edits. Non-confirmed users and non-registered users can submit a proposed article through the Articles for Creation process, where it will eventually be reviewed and considered for publication.

Contents [hide]

Wikipedia:Processes

Article creation

Introductory
Getting started with Wikipedia
Article wizard
Your first article

Suggested articles
Most-wanted articles
Requested articles
Images needing articles

Concepts and guidelines
Standard layout
Lead section
Sections
Stub articles
Categorization

Development processes
Article development
Moving a page
Merging articles
Featured article criteria
The perfect article

Meta tools and groups
WikiProject: Articles for creation
Special:NewPages
New pages patrol
New articles by topic
Recent additions (DYK)

V • T • E

Introduction

First, please be aware that Wikipedia is an **encyclopedia**, and our mission is to share accepted knowledge to benefit people who want to learn. We are not social media or a place to promote a company or product or person, or a place to advocate for or against anyone or anything. Please keep this in mind, always. (This is described in our "mission statement", "What Wikipedia is not".)

We find "accepted knowledge" in high quality, published sources. By "high quality" we mean books by reputable publishers, high-quality newspapers like *The New York Times*, or literature reviews in the scientific literature. We *summarize* such sources here. That is all we do! Please make sure that anything you write in Wikipedia is based on such sources - not what is in your head.

Here are some tips that can help you with your first article:

- Register an account. All you need is to choose a username and password. This will give you various powers. After a few days of editing articles, it will give you the power to create a new one.
- Biographies of living people are among the most difficult articles to get right. Consider starting with something easier.
- Search Wikipedia first in case an article *already exists* on the subject, perhaps under a different title. If the article already exists, feel free to make any constructive edits to improve it.

- Nothing? OK, now you need to try to determine if the subject you want to write about is what we call "notable" in Wikipedia. The question we ask is - **does this topic belong in an *encyclopedia*?**
  - We generally judge this by asking if there are at least three high-quality sources that a) have substantial discussion of the subject (not just a mention) *and* b) are written and published independently of the subject (so, a company's website or press releases are not OK). Everything here is based on high-quality independent sources, and without them, we generally just cannot write an article. If you are not sure if the subject you want to write about is "notable", you can ask questions at the Wikipedia Teahouse.
  - Please be mindful of conflict of interest. If you have one, you will probably have a hard time writing a good enough Wikipedia article (this is not about you, it is just human nature). However, if you insist on trying, you need to disclose your conflict of interest, and you need to try very hard not to allow your "external interest" to drive you to abuse Wikipedia. And you need to try hard to *hear* the feedback from independent people who review the draft before it is published and made available in the main encyclopedia. Your conflict of interest might lead you to believe something is "notable" when it isn't and to argue too hard for it to be published there.
- **Practice first.** Before starting, try editing existing articles to get a feel for writing and for using Wikipedia's mark-up language—we recommend that you first take a tour through the Wikipedia tutorial or review contributing to Wikipedia to learn editing basics.
- The **Article Wizard** will help you create your article in Draft space, and will put some useful templates into your draft, including the button to click when you are ready to submit the draft for review.

Article Wizard
An easy way to create articles.

These points are explained in further detail below.

If you are logged in, and your account is autoconfirmed, you can also use this box below to create an article, by entering the article name in the box below and then

clicking "Create page".

Create page

Search for an existing article

The English Wikipedia already has 5,810,846 articles. Before creating an article, try to make sure there is not already an article on the same topic, perhaps under a slightly different name. Search for the article, and review Wikipedia's article titling policy before creating your first article. If an article on your topic already exists, but you think people might look for it under some different name or spelling, learn how to create *redirects* to alternative titles; adding needed redirects is a good way to help Wikipedia. If you're not already autoconfirmed, you can request a redirect to be created at Wikipedia:Articles for creation/Redirects, where a volunteer will review the request, and if it seems like a plausible search term, accept the redirect request. Also, remember to check the article's deletion log in order to avoid creating an article that has already been deleted. (In some cases, the topic may be suitable even if deleted in the past; the past deletion may have been because it was a copyright violation, did not explain the importance of the topic, or on other grounds addressed to the writing rather that the topic's suitability.)

If a search does not find the topic, consider broadening your search to find existing articles that might include the subject of your article. For example, if you want to write an article about a band member, you might search for the band and then add information about your subject as a section within that broader article.

Gathering references

Gather sources for the information you will be writing about. To be worthy of inclusion in an encyclopedia, a subject must be sufficiently notable, and that notability must be verifiable through citations to reliable sources.

As noted, the sources you use must be reliable; that is, they must be sources that exercise some form of editorial control and have some reputation for fact-checking and accuracy. Print sources (and web-based versions of those sources) tend to be the most reliable, though some web-only sources may also be reliable. Examples might include (but are not limited to) books published by major publishing houses, newspapers, magazines, peer-reviewed scholarly journals, websites of any of the above, and other websites that meet the same requirements as a reputable print-based source.

Notability

Subject-specific guidelines

Academics
Astronomical objects
Books · Events
Films · Fringe theories ·
Geographic features · Music ·
Numbers
Organizations and companies
People · Sports and athletes
Web content

See also

Wikipedia essays
Guide to deletion
Common deletion outcomes
Why was my article deleted?

V · T · E

In general, sources with **no** editorial control are not reliable. These include (but are not limited to) books published by vanity presses, self-published 'zines', blogs, web forums, usenet discussions, personal social media, fan sites, vanity websites that permit the creation of self-promotional articles, and other similar venues. If anyone at all can post information without anyone else checking that information, it is probably not reliable.

To put it simply, if there are reliable sources (such as newspapers, journals, or books) with extensive information published over an extended period about a subject, then that subject is notable and you must cite such sources as part of the process of creating (or expanding) the Wikipedia article. If you cannot find such *reliable sources* that provide extensive and comprehensive information about your proposed subject, then the subject is not notable or verifiable and almost certainly will be deleted. So your first job is to **go find references** to cite.

There are many places to find reliable sources, including your local library, but if internet-based sources are to be used, start with books and news archive searches rather than a web search.

Once you have references for your article, you can learn to place the references into the article by reading Help:Referencing for beginners and Wikipedia:Citing sources. Do not worry too much about formatting citations properly. It would be great if you did that, but the main thing is to **get references into the article**, even if they are not perfectly formatted.

Things to avoid

Main pages: Wikipedia:What Wikipedia is not and Wikipedia:Avoiding common mistakes

Articles about yourself, your family or friends, your website, a band you're in, your teacher, a word you made up, or a story you wrote

If you *are* worthy of inclusion in the encyclopedia, let someone else add an article for you. Putting your friends in an encyclopedia may seem like a nice surprise or an amusing joke, but articles like this are likely to be removed. In this process, feelings may be hurt and you may be blocked from editing if you repeatedly make attempts to re-create the article. These things can be avoided by a little forethought on your part. The article may remain if you have enough humility to make it neutral and you really are notable, but even then it's best to submit a draft for approval and consensus of the community instead of just posting it up, since unconscious biases may still exist of which you may not be aware.

Advertising

Please do not try to promote your product or business. Please do not insert external links to your commercial website unless a neutral party would judge that the link truly belongs in the article; we do have articles about products like Kleenex or Sharpies, or notable businesses such as McDonald's, but if you are writing about a product or business be sure you write from a neutral point of view, that you have no conflict of interest, and that you are able to find references in reliable sources that are independent from the subject you are

writing about. For a business, make sure it meets the specific notability guidelines for businesses.

Attacks on a person or organization

Material that violates our biographies of living persons policy or is intended to threaten, defame, or harass its subject or another entity is not permitted. Unsourced negative information, especially in articles about living people, is quickly removed, and attack pages may be deleted immediately.

Personal essays or original research

Wikipedia surveys *existing* human knowledge; it is not a place to publish new work. Do not write articles that present your own original theories, opinions, or insights, *even if* you can support them by reference to accepted work. A common mistake is to present a novel synthesis of ideas in an article. Remember, just because both Fact A and Fact B are true does not mean that A caused B, or vice versa (fallacies) or (Post hoc ergo propter hoc). If the synthesis or causation is true, locate and cite reliable sources that report the connection.

Non-notable topics

People frequently add pages to Wikipedia without considering whether the topic is really notable enough to go into an encyclopedia. Because Wikipedia does not have the space limitations of paper-based encyclopedias, our notability policies and guidelines allow a wide range of articles – however, they do not allow *every* topic to be included. A particularly common special case of this is pages about people, companies, or groups of people, that do not substantiate the notability or importance of their subject with reliable sources, so we have decided that such pages may be speedily deleted under our speedy deletion policy. This can offend – so *please* consider whether your chosen topic is notable enough for Wikipedia, and then substantiate the notability or importance of your subject by citing those reliable sources in the process of creating your article. Wikipedia is not a directory of everything in existence.

A single sentence or only a website link

Articles need to have real content of their own.

See also:

- List of bad article ideas

And be careful about...

Copyright

As a general rule, **do not copy-paste text from other websites**. (There are a few limited exceptions, and a few words as part of a properly cited and clearly attributed quotation is OK.)

– *Wikipedia:Copy-paste*

Copying things. Do not violate copyrights

Never copy and paste text into a Wikipedia article unless it is a relatively short quotation, placed in quotation marks, and cited using an inline citation. Even

material that you are *sure* is in the public domain must be attributed to the source, or the result, while not a copyright violation, is plagiarism. Also, note that most web pages **are not** in the public domain and most song lyrics **are not** either. In fact, most things published after 1923, and almost all works written since January 1, 1978, are automatically protected by copyright under the Copyright Act of 1976 *even if they have no copyright notice or © symbol.* If you think what you are contributing is in the public domain, *say where you got it,* either in the article or on the discussion page, and on the discussion page give the reason why you think it is in the public domain (e.g. "It was published in 1895..."). For more information, see Wikipedia:Copyrights (which includes instructions for verifying permission to copy previously published text) and our non-free content guidelines for text. Finally, please note that superficial modification of material, such as minor rewording, is insufficient to avoid plagiarism and copyright violations. See Wikipedia:Close paraphrasing.

Good sources

1. have a reputation for reliability: they are reliable sources
2. are independent of the subject
3. are verifiable by other editors

Good research and citing your sources

Articles written out of thin air may be better than nothing, but they are hard to verify, which is an important part of building a trusted reference work. Please research with the best sources available and cite them properly. Doing this, along with not copying text, will help avoid any possibility of plagiarism. We welcome *good* short articles, called "stubs", that can serve as launching pads from which others can take off – stubs can be relatively short, a few sentences, but should provide some useful information. If you do not have enough material to write a good stub, you probably should not create an article. At the end of a stub, you should include a "stub template" like this: {{stub}}. (Other Wikipedians will appreciate it if you use a more specific stub template, like {{art-stub}}. See the list of stub types for a list of all specific stub templates.) Stubs help track articles that need expansion.

Articles about living persons

Articles written about living persons must be referenced so that they can be verified. Biographies about living subjects that lack sources may be deleted.

Advocacy and controversial material

Please do not write articles that advocate one particular viewpoint on politics, religion, or anything else. Understand what we mean by a neutral point of view before tackling this sort of topic.

Articles that contain different definitions of the topic

Articles are primarily about what something **is**, *not* any term(s). If the article is just about a word or phrase and especially if there are very different ways that a term is used, it usually belongs in Wiktionary. Instead, try to write a good short first paragraph that defines one *subject* as well as some more material to go with

it.

Organization

Make sure there are incoming links to the new article from other Wikipedia articles (click "What links here" in the toolbox) and that the new article is included in at least one appropriate category (see help:category). Otherwise, it will be difficult for readers to find the article.

Local-interest articles

These are articles about places like schools, or streets that are of interest to a relatively small number of people such as alumni or people who live nearby. *There is no consensus* about such articles, but some will challenge them if they include nothing that shows how the place is special and different from tens of thousands of similar places. Photographs add interest. Try to give local-interest articles local colour. Third-party sources are the only way to prove that the subject you are writing about is notable.

Breaking news events

While Wikipedia accepts articles about notable recent events, articles about breaking news events with no enduring notability are not appropriate for our project. Consider writing such articles on our sister project Wikinews. See Wikipedia:Notability (events) for further information.

Editing on the wrong page

If you're trying to create a new page, you'll start with a completely empty edit box. If you see text in the editing box that is filled with words you didn't write (for example, the contents of this page), you're accidentally editing a pre-existing page. Don't "Publish changes" your additions. See Wikipedia:How to create a page, and start over.

Are you closely connected to the article topic?

Wikipedia is the encyclopedia that anyone can edit, but there are special guidelines for editors who are paid or sponsored. These guidelines are intended to prevent biased articles and maintain the public's trust that content in Wikipedia is impartial and has been added in good faith. (See Wikipedia's conflict of interest (COI) guideline.)

The official guidelines are that editors **must be volunteers.** That means Wikipedia discourages editing articles about individuals, companies, organizations, products/services, or political causes that pay you directly or indirectly. This includes in-house PR departments and marketing departments, other company employees, public relations firms and publicists, social media consultants, and online reputation management consultants. However, Wikipedia recognizes the large volume of good faith contributions by people who have some affiliation to the articles they work on.

Here are some ground rules. If you break these rules, your edits are likely to be reverted, and the article(s) and your other edits may get extra scrutiny from other Wikipedia editors. Your account may also be blocked.

Things to avoid	Things to be careful	Great ways to contribute

	about	
• Don't add promotional language • Don't remove negative/critical text from an article • Don't make a "group" account for multiple people to share • Don't neglect to disclose your affiliation on the article's talk page	• Maintain a neutral, objective tone in any content you add or edit • Cite independent, reliable sources (e.g., a major media article) for any new statements you add – even if you are confident a statement is true (e.g., it is about your work), say it only if it has been already published in a reliable source.	• Make minor edits/corrections to articles (e.g., typos, fixing links, adding references to reliable sources) • If you are biased, suggest new article text or edits on the *article talk page* (not on the main article page). • Disclose your relationship to the client/topic. • Edit using personal accounts. • Recruit help: Seek out a sponsor (volunteer editor) who has worked on similar articles, or submit ideas for article topics via Requested articles.

Note that this has to do only with conflict of interest. Editors are encouraged to write on topics related to their expertise: e.g., a NASA staffperson might write about planets, or an academic researcher might write about their field. Also, Wikipedians-in-residence or other interns who are paid, hosted or otherwise sponsored by a scientific or cultural institution can upload content and write articles in partnership with curators, indirectly providing positive branding for their hosts.

Create your draft

Click here: Article wizard, read the brief introduction, and then click the big blue button to get started creating your draft.

And then what?

Now that you have created the page, there are still several things you can do.

Keep making improvements

Wikipedia is not finished. Generally, an article is nowhere near being completed the moment it is created. There is a long way to go. In fact, it may take you several edits just to get it started.

If you have so much interest in the article you just created, you may learn more about it in the future, and accordingly, have more to add. This may be later today, tomorrow, or several months from now. Any time – go ahead.

Improve formatting

To format your article correctly (and expand it, and possibly even make it featured!),

see

- Wikipedia:Tutorial to learn how to format your article
- Wikipedia:Writing better articles
- Wikipedia:The perfect article
- Wikipedia:Lead section

Others can freely contribute to the article when it has been saved. The creator does not have special rights to control the later content. See Wikipedia:Ownership of articles.

Also, before you get frustrated or offended about the way others modify or remove your contributions, see Wikipedia:Don't be ashamed.

Avoid orphans

An orphaned article is an article that has few or no other articles linking to it. The main problem with an orphan is that it'll be unknown to others, and may get fewer readers if it is not de-orphaned.

Most new articles are orphans from the moment they are created, but you can work to change that. This will involve editing one or more *other* articles. Try searching Wikipedia for other pages referring to the subject of your article, then turn those references into links by adding double brackets to either side: "[[" and "]]". If another article has a word or phrase that has the same meaning as your new article, but not expressed in the same words as the title, you can link that word or phrase as follows: "[[title of your new article|word or phrase found in other article]]." Or in certain cases, you could create that word or phrase as a redirect to your new article.

One of the first things you want to do after creating a new article is to provide links to it so it will not be an orphan. You can do that right away, or if you find that exhausting, you can wait a while, provided that you keep the task in mind.

See Wikipedia:Drawing attention to new pages to learn how to get others to see your new articles.

Add to a disambiguation page

If the term is ambiguous (meaning there are multiple pages using that or a similar title), see if there is a disambiguation page for articles bearing that title. If so, add it to that page.

Still need help?

Help desk

- For a list of informative, instructional and supportive pages, see Help directory.
- The best places to ask for assistance are at the Teahouse and at the main Help desk.
- Click here to ask for help on your talk page, a volunteer will visit you there shortly!
- For a list of the services and assistance that can be requested on Wikipedia, see Request departments.
- Alternately you can ask a question through the Wikipedia #wikipedia-en-help

connect on IRC chat.

Read a traditional encyclopedia

Try to read traditional paper encyclopedia articles (or good or featured articles on Wikipedia) to get the layout, style, tone, and other elements of encyclopedic content. It is suggested that if you plan to write articles for an encyclopedia, you have some background knowledge in formal writing as well as about the topic at hand. A composition class in your high school or college is recommended before you start writing encyclopedia articles.

The *World Book* is a good place to start. The goal of Wikipedia is to create an up-to-the-moment encyclopedia on every notable subject imaginable. Pretend that your article will be published in a paper encyclopedia.

V · T · E	**Basic information on Wikipedia**
Main Help (**directory** · **menu**) · **FAQs** · **Reference desk** · **Help desk**	
About Wikipedia	Administration (Purpose · Who writes Wikipedia? · Organization) · Censorship · Introduction · Why create an account? · In brief · General disclaimer · What Wikipedia is not
Readers' FAQ	Parental advice · Navigation (Introduction) · Searching · Viewing media (Help) · Mobile access · Other languages · Researching with Wikipedia (Citing Wikipedia) · Students help · Readers' index · Copyright · Book creation
Contributing to Wikipedia	Main tutorial (Tutorials and introductions) · The answer · Dos and don'ts · Learning the ropes · Common mistakes · Newcomer primer · Plain and simple · **Your first article** (Wizard) · Young Wikipedians (The Adventure)
Protocols and conventions	Five pillars (Introduction · Simplified ruleset · Simplified MoS) · Etiquette (Expectations · Oversight) · Principles · Ignore all rules (The rules are principles) · Core content policies · Policies and guidelines (Policies · Guidelines) · Vandalism · Appealing blocks
Getting assistance	Requests for help (Request editor assistance · Disputes resolution requests) · IRC live chat (Tutorial) · Village pump · Contact us
Wikipedia community	Community portal · Dashboard (Noticeboards) · Departments · Maintenance (Task Center) · Essays · Meetups · WikiProjects
Sourcing and referencing	Finding sources · Combining sources · Referencing (Introduction) · Citations (Citation Style 1 · Citation templates · Footnotes · Page numbers · Cite errors)
Information	Editing (Toolbar · Conflict) · VisualEditor (User guide) · Category · Diffs · Email confirmation · Infoboxes · Linking (Link color) · Manual of Style (Introduction · Simplified) · Namespaces · Page name · URLs · User contribution pages · Using talk pages (Introduction · Archiving) · Image and media files (Images · Media files)
How-to	Guide to page deletion (Image deletion) · Logging in · Merging pages · Page renaming (Requests) · Redirecting · Reset passwords · Reverting · Uploading images (Introduction)
Wiki markup	Wiki markup (Cheatsheet) · Barcharts · Calculations · Columns · HTML · Lists · Magic words (For beginners) · Music symbols · Sections · Sounds · Special Characters · Tables (Introduction) ·

	Templates (Documentation · Messages) · Tools · Transclusion · Visual file markup (Tutorial)
Directories	Abbreviations · Contents · Edit summaries · Essays · Glossary · Index · *The Missing Manual* · Shortcuts · Tips (Tip of the day) · Wikis
Teahouse (interactive help for new editors) · **Ask for help on your talk page** (a volunteer will visit you there)	

Categories: Wikipedia basic information | Wikipedia how-to | Wikipedia page help | Wikipedia tips | Wikipedia new articles

This page was last edited on 22 February 2019, at 12:46 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Developers Cookie statement

Mobile view

PBS.org

OUR SPONSORS
DRAPER

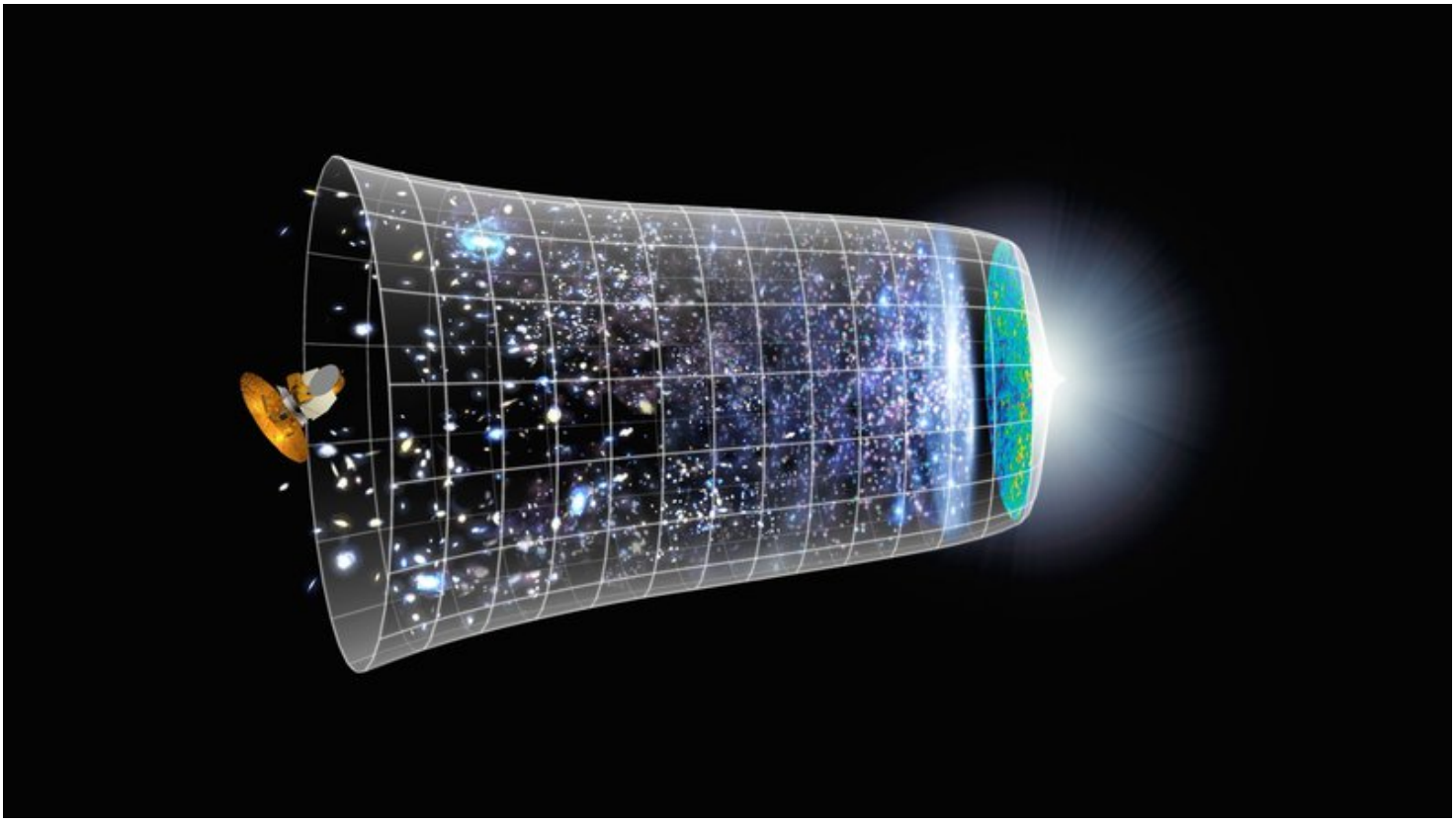
SUPPORT PROVIDED BY LEARN MORE

NOVA

PHYSICS + MATH

Big Bang May Have Created a Mirror Universe Where Time Runs Backwards

BY TIM DE CHANT MONDAY, DECEMBER 8, 2014 NOVA NEXT



In a mirror universe, from our perspective, time may run backwards from the Big Bang. Image credit: NASA / WMAP Science Team

SHARE

Why does time seem to move forward? It's a riddle that's puzzled physicists for well over a century, and they've come up with numerous theories to explain time's arrow. The latest, though, suggests that while time moves forward in our universe, it may run backwards in another, mirror universe that was created on the "other side" of the Big Bang.

Related

Tiny Black Holes

The Many Worlds Theory Today

Elements in the Ocean

Two leading theories propose to explain the direction of time by way of the relatively uniform conditions of the Big Bang. At the very start, what is now the universe was homogeneously hot, so much so that matter didn't really exist. It was all just a superheated soup. But as the universe expanded and cooled, stars, galaxies, planets, and other celestial bodies formed, birthing the universe's irregular structure and raising its entropy.

One theory, proposed in 2004 by Sean Carroll, now a professor at Caltech, and Jennifer Chen, then his graduate student, says that time moves forward because of the contrast in entropy between then and now, with an emphasis on the fact that the future universe will so much more disordered than the past. That movement toward high entropy gives time its direction.

SUPPORT PROVIDED BY LEARN MORE

The new theory says a low entropy early universe is inevitable because of gravity, and ultimately that's what gives time its arrow. To test the idea, the theory's proponents assembled a simple model with nothing more than 1,000 particles and the physics of Newtonian gravity. Here's Lee Billings, reporting for Scientific American:

> *The system's complexity is at its lowest when all the particles come together in a densely packed cloud, a state of minimum size and maximum uniformity roughly analogous to the big bang. The team's analysis showed that essentially every configuration of particles, regardless of their number and scale, would evolve into this low-complexity state. Thus, the sheer force of gravity sets the stage for the system's expansion and the origin of time's arrow, all without any delicate fine-tuning to first establish a low-entropy initial condition.*

But here's the twist: The expansion after the simulated Big Bang didn't just happen in one direction, but two. The simple Big Bang they modeled produced two universes, one a mirror of the other. In one universe, time appears to run forwards. In the other, time runs backwards, at least from our perspective.

Recieve emails about upcoming NOVA programs and related content, as well as featured reporting about current events through a science lens.

Email Address

Zip Code

SUBSCRIBE

Here's Billings again, interviewing lead author Julian Barbour from the University of Oxford:

> *"If they were complicated enough, both sides could sustain observers who would perceive time going in opposite directions. Any intelligent beings there would define their arrow of time as moving away from this central state. They would think we now live in their deepest past."*

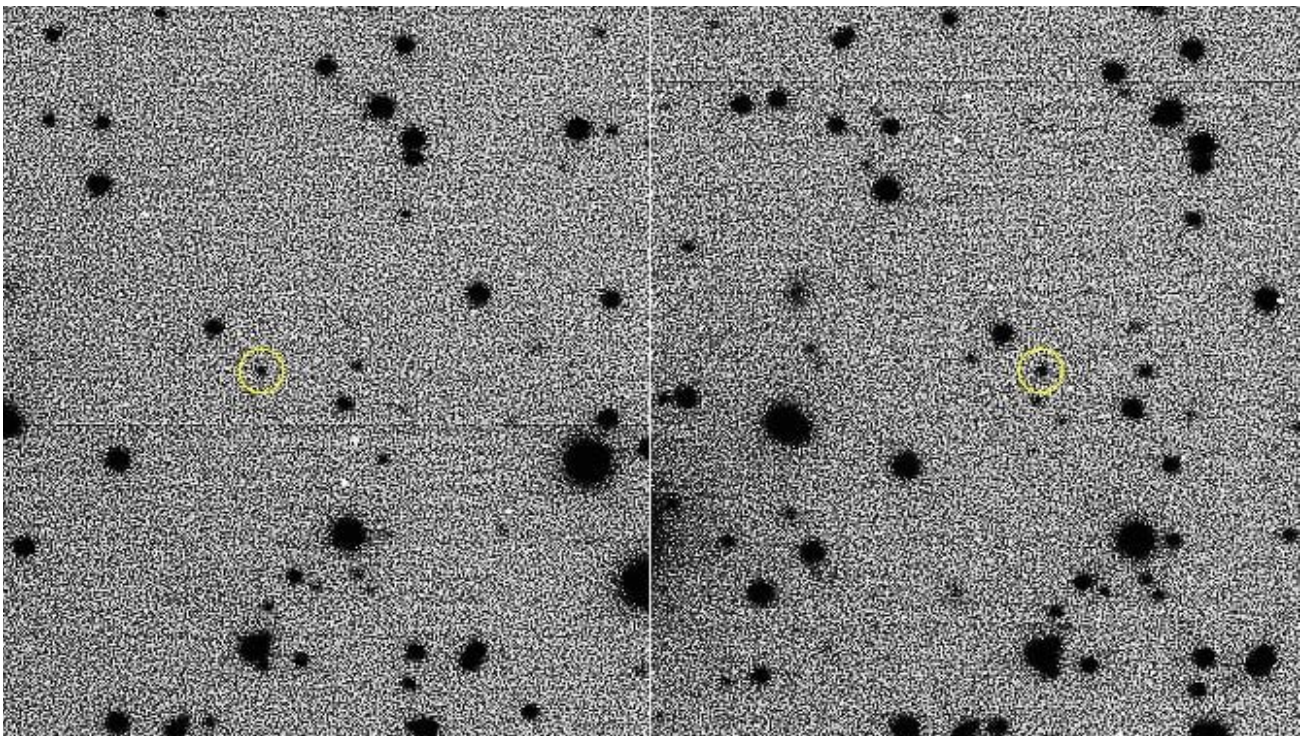
From that perspective, maybe George Lucas's *Star Wars* didn't take place a long time ago in a galaxy far, far away, but in the far future—our deepest past—of our mirror universe.

Image credit: NASA / WMAP Science Team

Tim De Chant

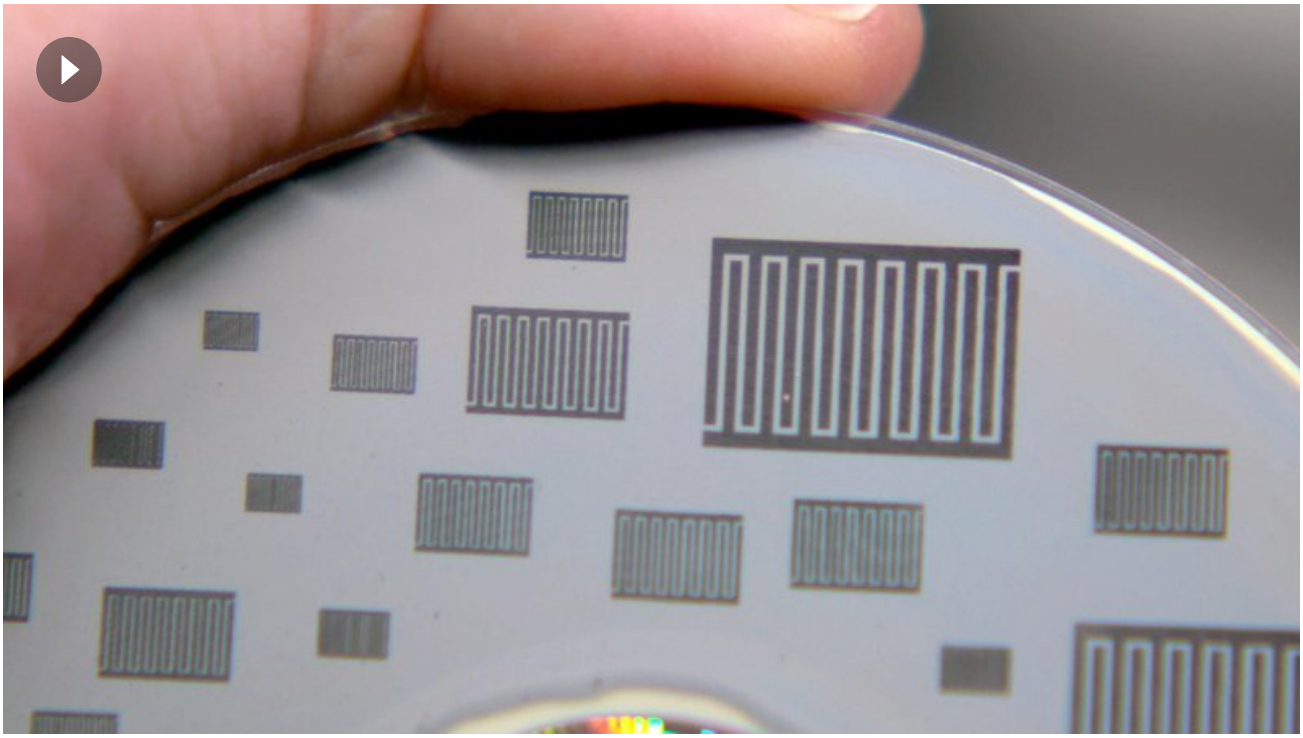
POSTS BY THIS CONTRIBUTOR ›

Most Popular



READ | ARTICLE

Rebel Asteroid Found Flying Backwards Near Jupiter

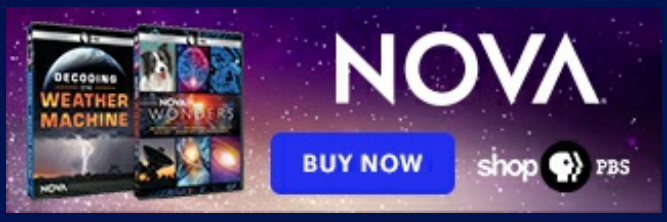


WATCH | SHORT

Supercharged Supercapacitors

VISIT | EXTERNAL LINK

NOVA Wonders Collection



This website was produced for PBS Online by WGBH. PBS is a 501(c)(3) not-for-profit organization.

© 1996–2019 WGBH Educational Foundation

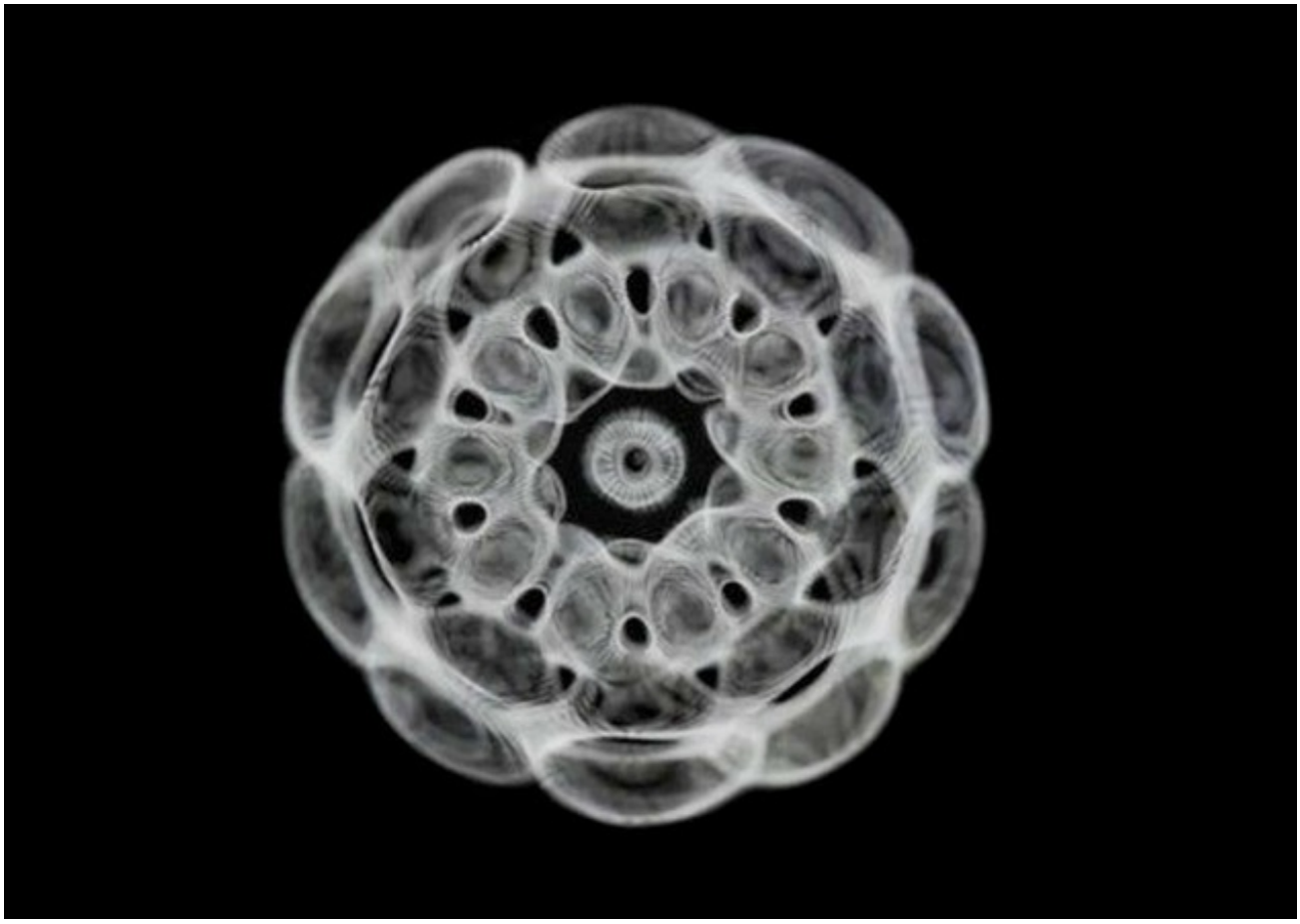
Funding for NOVA Next is provided by the Eleanor and Howard Morgan Family Foundation.

National corporate funding for NOVA is provided by Draper. Major funding for NOVA is provided by the David H. Koch Fund for Science, the Corporation for Public Broadcasting, and PBS viewers.

LATEST DIALOGUES

It's Time to Allow Vibration to Have its Way

by **Greg Sipp** ▪ in **Biology**, **Cognitive Sciences**, **Dialogues** ▪ 0 **Comments**



By Greg Sipp

"The Science of the future will b

~ Rudolf Steiner

For well over a century, scientis
revealing to us about fundamenta
rectify how our macroscopic view
It's been an impossible problem,
real, and who we truly our as hum

and returning to nothing. These v
contracting forces are balanced w
molecules and all the "solid" obj

Because everything is vibration,
great Dutch scientist and clock m
pendulums in clocks that were rur
that, Pythagoras observed sympat
chaos theory, self-organizing sys
resonance is kCeoyu.l dl nC anrsecc ioerutsnaersts
*Vib*Tra*a*m*t* *e* HPunt writes the following:

"*Something interesting happens wh*
after a little while, "$ɡ*n"*scio bonrpea,t tiem et s*
can seem mysteriouss.p oTmh tia n ieso üdse ss

Everything in your body resonates
cell that these atoms make up pul
movement. Your breath is an osci

There are countless resonators in
sympathetic resonance with one a
interference *d*p*e*a*c*t *c*t *k*e a*e*nm d*e* t*n* wh*c* i*a*elt l tke en ed ps tth
happening to its full*n* *e*wsh*t*oe *x*y toeun *t*a.re
y*t*o h*u*y*i* o*n* u*k* are. This interference pat
of the body, preventing optimum sy

The biggest resonator in your bod
veins and arteries. Your brain al
Bringing the resonators in the he
dissolving the interference patte
ego. This is what Heartmath is he
area of t*thaen* ib*t*no*i* d*e*Ty*n*a oc iaslml e d W th he en eithe
sympathetic relationship with the
reduced. Key to these relationshi

"We have met the enemy, and they
ego that just about everyone is c
systems and prevents us from kno
Source. The ego quite often make
evidenced in myriad ways — in the
throughout the world, and in the e
population, including many non-hu

Clearly, the ego of the human bei
happens when one is not connecte
many ways, is not helpful. This i
could be simply and gently dissol
senses. Unfortunately, the ego in
the Christian crucifixion. The cr
resonance. This crucifixion can b
deeply the ego is willing to let g
greatly reduced.

The issues we each face in our in
usually exacerbated through stres
ego experiences itself as separat

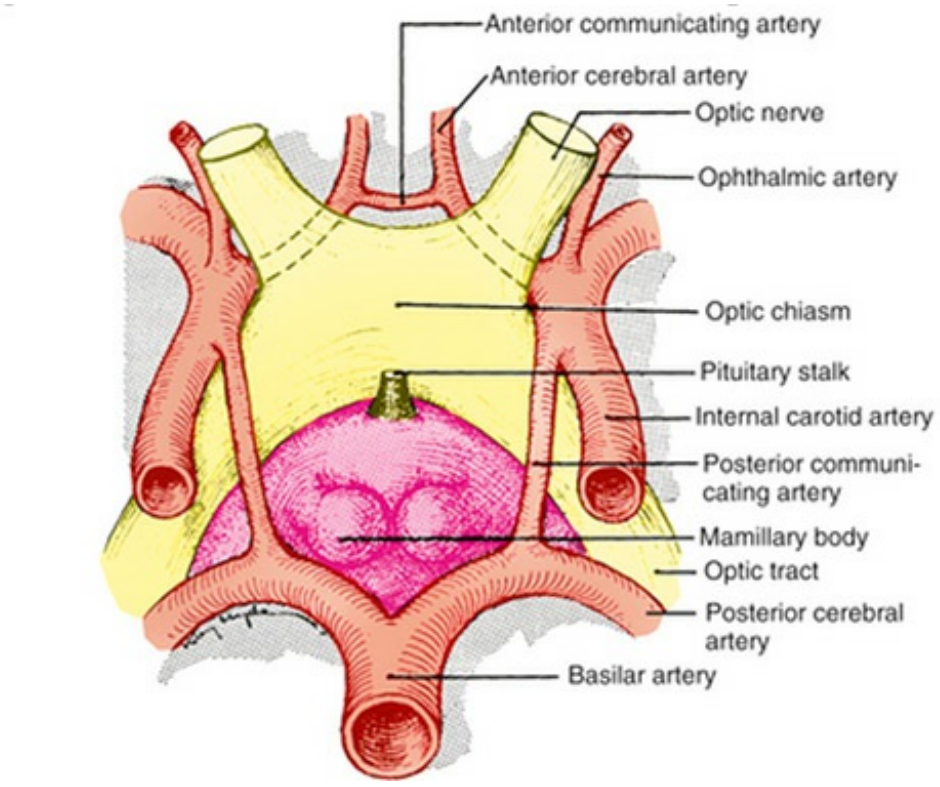
with fear and pain and, ultimatel
in and pray to a higher being, mo
which it can never fully do. The
with the images of who you were i
future. We store these images in
constructed of images and though
was never real to begin with.

Once we really come to understan
up of resonators, the ego can be
with? It is interfering with the n
sympathetic resonance. The ego's
life, known in Christianity as the

What scientists have been grappli
made up of particles or waves, de
observe them. It seems in this ma
seems we're stuck in this particl
is connected, exists only on the
see and the ears to hear."

So how do we change our percepti
invisible wave parad*ai*l*g*s*l*m ? i b*l* ro*a*w t*i*d*o*o n ?

In regard to seeing *a*, *l* i t*i* s t*i* s*v* *i* b*b* i*y*r t*a* at*y*l*i* d*o* u
eyes. There is a point behind and
cross that starts the process of
ever created is pierced and let g
circulation of the bloodstream. T
This area is called the optic chi
ventricle.



This resonator wants to fall into
ventricle, also known as the cave
node or still point of the sensor

the sensorium of the brain is bec
the vibratory role it plays with t
is right between two of the major
a relay station, anterior to the p
integrating center of seeing. The
and pineal glands start to fall in
cerebral spinal fluid of the third
thousands of years as the music o
savage beast." In this case the "
the root of the separative ego. W
sympathetic vibration.

"*The cerebral spinal fluid may abs*
experience'I 'oA~uMMra ubreoi nZg anpepsast, ertrhæ, M D

Allowing the resonators that make
what brings forth this inward-dir
"two-eyed" point of view that we

And how do we develop "ears to h
of connectedness? If we are to co
but didn't have the ability to rec
sympathetic resonance in the ear.
Tomatis, creator of the Tomatis M
One antenna, the cochlea, is the o
with bodily proprioception. When
of feeling and hearing are one. I
when "seeing" wasn't available an
experiencing your environment. T
bot*ahmde*ea el the breath, and the hear
her organs would make. Most impo
umbilicus, giving you two types o
eternal nourishment, the pulsatio
pulsations at the naval as the "p
pulsatory continuum is what we hu
gaze and touch, in Being.

In the liquid environment of the
of feeling and hearing become inc
primary sense we use to know the
hooks up with seeing to create th
thousands of images that we store
attached to each one of these ima
reality, who we are in time/space
creates our sense of time. In con
moment in pure space.

It is time for us to move in our p
"original sin," to the other way o
constant, unconscious need to "co
collapsing the wave function is t
there is a predator or a bus comi
is" in every moment, you wouldn't
particle point of view outer worl
know from quantum mechanics, tha
in how we expT*ehi*ensicse owuhra ti nwheercean lt
Discovering the flow is discoveri

penetrate your senses – from see even you!

“I Am” an activity, not a thing.

– – – – – – – – – – – –

Greg has studied cranial sacral t with Joseph Heller and Continuum which endeavors to hear the soun directs one’s attention to the “I attended numerous “Towards a Sci

Email:8750 1@aol.com

You can visit www.thoughtexplorer.com

Related Dialogues

What Can Cognitive Neuroscience Tell Us About Perc...

Everything Only Looks Like a Thing: Neil Theise

Deleting the App, The New Ritual of Commitment: Es...

Sex Chromosomes Aren’t the Whole Story

Share + | Appreciate | Tags | About the author

Leave a Reply

You must be logged in to post a comment.

WHEN

2019 Italy

Retreat July 02nd- 08th

WHERE

Titignano Castle in Orvieto Terni
Italy

WHEN

2019 US

Pre-conference Workshops begin Oct. 23rd

Main Conference Oct. 24th- 27th

WHERE

Dolce Hayes Mansion
San Jose, CA

SUPPORT SAND

Science and Nonduality is a not-for-profit organization and your donation goes directly towards continuing the development and expansion of the SAND vision and our community. Thank you for your support!

DONATE

Dialogue Categories

Select Category

Dialogue Archives

Select Month

STAY CONNECTED

1. SAND community

Username or Email

Password

LOG IN

LOST PASSWORD

2. Get the SAND newsletter

SUBSCRIBE

3. Connect with us

Like 593K people like this. Sign Up to see what your friends like.

Connect on Facebook Follow us on Twitter Youtube

SAND is a wonderful place for you to offer their thoughts and ideas to others and to get feedback to help you further hone and articulate what you are trying to bring forward into the world. It is a place where ideas can be exchanged and a place where the cross pollination between science and spirit offers new perspectives to those working in both areas. It is also a place of learning and growing - a place where you can think about things you never thought about before.

– Isa

THANKS TO OUR SPONSORS

ConsciousLifestyle magazine

newharbingerpublications
NON-DUALITY PRESS SAHAJA

MAPS
MULTIDISCIPLINARY ASSOCIATION FOR PSYCHEDELIC STUDIES

© 2018 Science and Nonduality . All rights reserved.

ISSN 1551-3483

https://scale.qihardware.org