Shared Ownership of Scarce Bitcoin with Multi Signatures in the Lightning Network

Bachelor's Thesis

by

Max Hillebrand

Declaration of Honor

I hereby certify that I have written my thesis with the topic: *Shared Ownership of Scarce Bitcoin in the Lightning Network* independently and did not use any sources or tools other than those specified. The thesis is published under the Creative Commons Attribution-Share Alike 4.0 International License.

I declare that the commits of the account @MaxHillebrand to the git research repository https://github.com/MaxHillebrand/LightningMultiSig/ were done only by myself, this can be verified by the GPG signatures of the commits.

I also declare that the electronic version *SharedOwnershipOfScarceBitcoinWithMultisignaturesInTheLightningNetwork_HillebrandMax.pdf* matches this printed version. SHA256 of the pdf:

_____

A .txt.asc file containing my GPG signature over this declaration of honor, including the SHA256 of the pdf and the most recent Bitcoin block hash, is time stamped in the Bitcoin time chain. The Open Time Stamp proof is available together with the pdf. This is a cryptographically valid Jurat proof of the time window the thesis was finished.

Villingen-Schwenningen, 29.06.2019
At Bitcoin block hight: 583035
Block hash: 000000000000000000005f9d1f93a39fa5528cf54735dbd29c87b42b7dcb2574b

_____

Max Hillebrand
GPG: E900 5F66 A86B B816 BD7D 967E BEDC D95C 42AC 3C57

Abstract

This thesis praxeologically defines the universal axiom of scarcity as the attribute of exclusivity and rarity of a good. Humans act in order to remove uneasiness by allocating scarce resources throughout time. An individual can acquire the just property right in a scarce good by either homesteading or exchanging it. Yet abundant non-scarce goods without exclusivity do not justify the need for resource allocation through property rights, as they are in the realm of mind and cyberspace. Bitcoin applies cryptography in order to emerge the scarcity of the unspent transaction output. Its scripting capability can define the property rights of non-simulated shared ownership between several individuals. This fundamental economic breakthrough can be applied in countless use cases to increase the security, efficiency and usability of libre sound money Bitcoin.

Acknowledgement

I thank the many peers who have supported me in my path towards understanding this complex topic. Especially I am thankful for the countless hours **Jonas Nick** has sacrificed in order to initiate myself in the magic of cryptography. I greatly appreciate the teachings of **Stephan Kinsella** who has fundamentally realigned my axioms in order to grow a mindset with scarcity at it's core.

6102 Bitcoin, **Adam Back**, **Adam Fiscor**, **Adam Gibson**, **Andreas Antonopolous**, **Andrew Poelstra** for your brilliant explanation of advanced magic, **Alex Brosworth**, **Candle Lover**, **Chris Belcher**, **David Harding** for the invaluable knowledge you share through OpTec, **Edmond Belliveau**, **Eric Voskuil** for challenging myself to always be consistent, **Giacomo Zucco**, **Gregory Maxwell**, **Hans Hermann Hoppe**, **Jeff Gallas**, **Justin Carter**, **John Newbery**, **Joseph Poon**, **Konstantin Nick**, **KZen Research**, **Lucas Ontivero**, **Ludwig von Mises**, **Manfred Karrer**, **Manuel Polavieja**, **Mark Erhardt**, **Marek Palatinus**, **Martin Habovstiak**, **Max Keidum**, **Murray Rothbard**, **Nicolas Dorier**, **Pavol Rusnak**, **Peter Gray**, **Pieter Wullie**, **Rene Pickhardt**, **Rudolfo Novak**, **Olaoluwa Osuntokun**, **Omer Shlomovits**, **Sjors Provoost**, **Stepan Snigirev**.

These are only a few peers who have directly supported the writing of this thesis, be it through high level conversations, detailed write-ups or fixing typos and structure. I thank the entire Bitcoin tribe for fostering the brightest and kindest hodlers of last resort.

Vires in numeris.

After two rounds of rigorous peer review of several drafts of this thesis, any errors in the published version are entirely the fault of the author.

List of Abbreviations and Definitions

AggPubK X́	Aggregated public key
Bitcoin [upper case B]	The open source software and network
bitcoin [lower case b]	The sound money unit of accoutn
c = H(X,R,m)	The hash of the public key, nonce commitment and message.
CSRNG	Cryptographically secure randon number generator
G	Cyclic group over a finite field
g	Generator point of G
H	Hash function to fingerprint data
IP	Intellectual Property
m	Threshold required for valid multisig
MAST	Merkelized Abstract Syntaxt Trees
m-of-n	m out of n threshold signatures required for a valid script
Multisig	Multi signatures, at least 2 signatures needed for a valid script
MuSig	Multisig based on the Schnorr signature scheme
n	Number of total keys in multisig
n-of-n	n out of n signatures required for a valid script
r	Random number generated with a CSRNG

R = H(r)	Hash commitment of a random number
UTXO	Unspent transaction output, the tip of chain of digital signatures
Vbytes	Virtual bytes used to calculate total block weight
VSS	Verifiable secret sharing
p	Prime number
P2PKH	Pay to Public Key Hash
P2WPKH	Pay to Witness Public Key Hash
P2SH	Pay to Script Hash
P2WSH	Pay to Witness Script Hash
PubK X = g^t	Public key
PrK t	Private key
PSBT	Partially signed Bitcoin transaction
Singlesig	Single signature required for a valid script
SSSS	Shamir's secret sharing scheme
s = r + cx	Part of the signature
σ = (R,s)	Signature is the tuple of R and s
X́	Aggregated public key

Table of Contents

PART II: SHARED OWNERSHIP OF SCARCE BITCOIN.....22

Introduction

In order to achieve wisdom, one must accumulate knowledge from a vast eclectic array of sources, internally seek to understand the subject by rigorous logic and reason, and apply in ones life what is understood to be truthful. This trivium process of education is based on some fundamental axioms that underly the universe humans manifest. It is of utmost importance to have a thorough grasp of the foundation on which more complex thought processes are build. If a theory is based on a wrong or imprecise axiom, then whatever is deduced above must be fallacious, regardless it's internal justification. But when the starting point is an uncontested fact that is properly defined, then whatever is deduced logically and with consistent reason, must undoubtedly be the truth.

Praxeology is the applied science of purposeful human action to allocate scarce resources throughout time. It's the study of individuals with subjective preferences of possible ends, and the ability to analyze and utilize the scarce means available in order to achieve these ends. This thesis seeks to explain scarcity as the fundamental precondition to human action, as the attribute of exclusivity and rarity of goods and as the justification for the need of property rights in order to allocate these scarce resources.

Yet in the same line of reasoning is the attribute of non-scarcity of goods that are either non-exclusive or abundant in nature. Any individual can utilize this knowledge without the need of another's sacrifice, thus there is no rivalry of who may use it. This thesis explains the axiomatic reason why there is no justification whatsoever for introducing artificial scarcity of naturally non-scarce goods through intellectual property rights.

Rather than to attempt to defy the universal truth of scarcity, cryptography embraces the attributes of non-scarce knowledge and libre open source technologies. Although information cannot be owned, it can be exclusively known by one individual, and cryptography allows for an efficient proof of existence without revealing the secret. One novel contribution of this research is the explanation of how Bitcoin utilizes non-scarce cryptography in order to emerge non-simulated ownership in a scarce good of

cyberspace. Full nodes define, verify and enforce the property rights boundaries of the bitcoin in an unspent transaction output.

The main contribution is the analysis of how the non-scarce cryptography of the Bitcoin protocol can grant several individuals shared ownership of a bitcoin. This is a groundbreaking achievement, since so far a scarce good could ultimately only be used by one individual for one task at one time. There are several different approaches and technologies available to achieve similar ends, and they are carefully analyzed and defined in part two of the thesis.

To ensure that the theories elaborated in part one, and the tools explained in part two are not just theoretical constructs without any relevance to human action, part three is dedicated to exploring several applications and use cases of shared ownership of a scarce money. This tool is valuable to advance the security, enable the collaboration and increase the efficiency of the libre sound money Bitcoin.

PART I: ON SCARCITY

To understand a non-trivial subject fundamentally, we start with an analysis of its axioms. The basis of all human action is scarcity, and out of this binary attribute we can deduct a logical framework of individual liberty, property rights and prosperity.[1]

Part one of this thesis is exploring the fundamental axiom of scarcity, and how this conflict of exclusion leads to the need for property rights to manage resource allocation and mutually beneficial exchange amongst individuals. The opposite polarity of non-scarcity is evident in the dealings of the mind and cyberspace where knowledge can be shared freely amongst peers.[2] Cryptography in general, and Bitcoin specifically, are concepts that utilize both scarce and non-scarce goods to create fascinating tools to remove uneasiness. Although Bitcoin, the network, is in non-scarce cyberspace, the individual bitcoin, the unspent transaction output, is fundamentally scarce, thus enabling non-simulated property rights in this libre sound money.

1 Mises, Human Action. 1949.

2 Hillebrand, Anarchy in Money and the chapter on Scarcity. 2018.

1. Human Action

Individuals act in order to remove uneasiness by allocating scarce resources throughout time. There are means available, and one can utilize them in order to achieve some desired end. Action is purposeful behavior believed to solve problems. This necessarily must take place in time. There is an a priori value judgment of the possible ends to seek out, and following the action to employ the means necessary to succeed in the plan. Time is the concept that something has changed, and since action is change it therefore implies the passing of time. *"Time is scarce for man only because whichever ends he chooses to satisfy, there are others that must remain unsatisfied."*[3]

Action implies a preference hierarchy of scarce goods. Alice has the choice to consume either an apple, coffee or pizza, yet she can at any time only use one of each of them. Although she would like to take all three of them at once, her means, time, is insufficient. When Alice chooses to eat the apple, she clearly ranks this as her most favorable satisfaction of the ends, the apple is better than the pizza or coffee. From the axiom of action, it can be deduced that the all means are scarce resources that are allocated to the most satisfactory end.

3 Rothbard. Man Economy and State, 1. The Fundamentals of Human Action, 2. First Implications of the Concept. 1962

2. Exclusivity of Scarce Goods

The first attribute of scarcity is the exclusivity of a good. When Alice has a wood log, she has full control over it and can use it throughout time and space, she can for example build a house with it. At the same time, Bob cannot have the same log of wood and applying it to a different task, like building a boat. It's either Alice, or Bob, who can enjoy the current possession and future utility at their full disposal, but not both of them at the same time. Only one can use this good to solve problems, and even though the other might need the good, he cannot utilize it at all. Alice must sacrifice the usage of the wood, in order that Bob can have it. Scarcity applies to everything that cannot be simultaneously owned, where one's ownership excludes that of others. Scarcity is an unwavering constraint on these goods, yet it is neither negative nor positive, but neutral and natural.

The exclusivity of a good does not in it of itself make it an economic good. Although there is the *potential* of conflict over who can use the good, there needs to be sufficient demand by individuals to utilize this scarce good. This scarce good needs to be the means used to achieve certain ends with purposeful human action. Only when humans actually use a scarce good in order to remove uneasiness then it is an economic good. In the case where a scarce good has not yet been occupied by anyone, then it is so far not of value to anyone. There is no need to allocate a non-economical good, since it is unclaimed for now, and in the case one would need it, he can simply homestead it.

3. Rarity of Scarce Goods

There are scarce goods, which are valuable and used by individuals as means to achieve ends, and they are limited in supply, so that only some individuals can use them, while others are excluded. Although many would like to own these rare goods, there is not enough for everyone. For example a small water reservoir in the dessert with otherwise no water source. Clearly, this water is a scarce good, that is desired by individuals, yet limited in supply not sufficient enough to satisfy everyone. The demand is larger than the supply, thus the buyers of this rare good will be willing to sacrifice a higher price in exchange.

A good is abundant when its supply is vastly larger than its demand. For example river water has potential of conflict of use. The water is also the means of many individuals who utilize this economic good in order to satisfy their ends. Yet the supply of the water at a river is more than sufficient to satiate any demand by any individual, thus there is no urgent need to allocate this super-abundant non-scarce good. Any individual who needs water for whatever reason can go to the river and take some, without taking from another individual. Only when there is enough demand so that not everyone can draw from the available supply is there the actual need to allocate these resources efficiently.

"Only because scarcity exists is there even a problem of formulating moral laws; insofar as goods are superabundant ('free' [non-scarce] goods), no conflict over the use of goods is possible and no action-coordination is needed"[4] Since only one individual alone can at any time use a limited scarce good, there needs to be an economic and moral principle for individuals to allocate the scarce means so to achieve their ends. There are three possible systems that might be applied: Individual Liberty with private property; Slavery with a master and slave class; or Communism with collective ownership.

4 Hoppe, Theory of Socialism and Capitalism, p.158, n.120. 1989.

4. Liberty

4.1. Private Property

The function of property rights is to mitigate interpersonal conflict by defining one specific and exclusive owner of each scarce resource. With property rights, it is always clear who owns which good and which goods are not allowed to be used, specifically the property of someone else. These property rights must be both visible and just, since otherwise they could not effectively mitigate conflict. One individual can only know that a specific good is owned by another if it has been advertised clearly. Thus, the definition of property rights must be universal, objective and unambiguous. This is specifically the Law of Non-Aggression: do not murder, do not assault, do not steal, do not trespass, do not rape, do not lie. Do not initiate aggressive force against the person or property of another peaceful individual.[5]

4.2. Homesteading

Resources are abundant in nature and the first individual to occupy a previously unused good and intermingle his human ingenuity to create higher order scarce goods has acquired the just property rights. *"Whatsoever, then, he removes out of the state that Nature hath provided and left it in, he hath mixed his labor with it, and joined to it something that is his own, and thereby makes it his property."*[6] Homesteading is the establishment of a link between a particular person and an particular scarce good before anybody else had done so.[7] Since natural resources are unowned and not used by anyone, the homesteader does not harm anyone else by occupying these goods. Contrarily, with these goods at his disposal, the homesteader has the means to remove problems in his uncertain future, thus marginally increasing his subjective values. Because nobody loses, and the homesteader gains, this act is in accordance to the Natural Law of Non-Aggression.

5 Hillebrand, Anarchy in Money, Chapter 1 on Natural Law. 2018.

6 Locke, The Two Treatises of Government. 1689.

7 See Hoppe, The Economics and Ethics of Private Property: Chapter 13 On the Ultimate Justification of the Ethics of Private Property. 1993.

4.3. Mutually Beneficial Exchange

Since property rights mean full and ultimate individual control over a scarce good, he can exchange the right to use these goods with another peer.[8] When two peers are exchanging scarce resources, Alice is sacrificing her property in that good, she is giving up the current and future enjoyment and prosperity of it. In order to compensate her for this loss, she will want to have something in return, a good or service being offered by Bob. And only when she values Bobs exchanged good marginally higher than she values her previous good, will she voluntarily agree to the trade. The same is true for Bob, but in reverse order. He will only trade if Alice's good is of higher marginal subjective value than the good he is sacrificing. Only because of this asymmetry of needs and desires will the two peers voluntarily choose to trade their goods, and only then it is mutually beneficial. Although both have sacrificed and lost access to a good, they have both gained marginally more, and are thus better off. Since nobody was harmed, there was no aggression and no violation to Natural Law.

Homesteading and mutually beneficial voluntary exchange are the only ways of justly acquiring the property rights in a scarce good. Anyone who is aggressively and without consent taking away a justly acquired good from another is politically breaking private property rights in that scarce good.

“The argument for property rights is based [...] on the need of individuals to employ means to achieve ends, and to avoid interpersonal conflict over such means. This scarcity is [...] a necessary background condition that must obtain before property rights can emerge."[9]

8 See Rothbard, Man Economy and State With Power and Markets, Chapter 2 Part 4 Terms of Exchange. 1964.

9 Kinsella, Against Intellectual Property, p. 40. 2008.

5. Slavery

In a system of slavery there exist two distinct classes of individuals: the master class who has property rights, and the slave class who has not. The tyrant has the legal right to exclusive ownership for all his homesteaded and exchanged scarce goods as well as his human slaves. He has the full benefit of their human ingenuity to remove problems, and can deploy them in various tasks. For fundamental praxeological analysis, it is not of concern how the master came to rule, through inherited monarchy, elected democracy or military dictatorship. The tyrant can affect others through autistic, binary or triangular intervention.

5.1. Autistic Intervention

In autistic intervention[10] the master denies the slave the peaceful use of a scarce good. For example during prohibition the tyrant aggresses against a slave who is himself consuming a good like alcohol without permission. The master claims a higher right in the slaves body and action, than the right of the individual. The master gains in the satisfaction of obedience and by the inaction of his slave, he only loses the opportunity cost of not enforcing the intervention. The slave loses due to the lack of problem solving, as well as through the direct and indirect aggression of the master, yet he gains nothing. He loses regardless his intention to adhere to the man made law, since even if he would voluntarily choose to comply, he is still subject to the implied coercion by the master. This is contrary to a free market of voluntary interaction, where the individual would be suggested and persuaded to stop the action non coercively without the threat of violence. Yet the free individual can, of course, deny the advice and continue allocating his scarce resources in the way he intended.

10 See Rothbard, Power and Markets, Chapter 2 Fundamentals of Intervention. 1962.

5.2. Binary intervention

A tyrant can enforce binary intervention[11] by coercing an individual to interact with the state. This can be the involuntary expropriation of scarce goods or scarce labor. The master gains in the products and services stolen, and only sacrifices the opportunity cost of not enforcing the taxation. The slave loses the scarce good, the opportunity cost of acquiring this good, and the implied coercion. In any case, the master gains at the expense of the slave, he constantly applies aggression and coercion and claims property rights in the person and goods of the individual slave. Since this action is not voluntary, it is praxeological proof that it is not mutually beneficial. Even though the state might use some of the stolen property to sustain the slaves, like food, shelter or health care, these goods are not desired marginally most, since they have not been voluntarily chosen. The tyrant breaks the legs of the slave, then hands him a crutch and demands gratitude for the support. This is, of course, in stark contrast to a free market with only mutual beneficial exchange.

5.3. Triangular intervention

In triangular intervention[12] the tyrant aggresses against two slaves who do not ask for permission to engage in trade amongst them. Through some form of licensing, registration or forced contract covenants, the state prescribes the way in which two peaceful individuals must act, and failure to comply results in punishment. The attack is either in form of price control, or product control. A minimum price artificially excludes both clients and entrepreneurs who would each willingly trade at a level below the minimum price. Since this is inherently not mutually beneficial, it destroys capital and prosperity, while benefiting the master only. When manipulating the nature of production directly, rather than the terms of exchange, the creation or sale of certain scarce goods is prohibited. Again, both client and entrepreneur are artificially prohibited from removing their problems and increasing their subjective marginal value scale, yet the master gains in the pleasure of submission.

11 See Rothbard, Power and Markets, Chapter 4 Binary Intervention: Taxation and 5 Binary Intervention: Government Expenditures. 1962.

12 See Rothbard, Power and Markets, Chapter 3 Triangular Intervention. 1962.

6. Communism

Communism is the utopia of shared ownership of scarce means of production. The workers shall collectively use the tools at their disposal to remove problems, from each according to his ability, to each according to his needs.[13] Yet this disregards the universal truth of exclusivity and rarity of scarce goods. The workers cannot collectively own the scarce means of production, since they are fundamentally scarce, and thus only one individual at one time can use them. In any case, ultimately the choice of what to do with a given scarce resource has do be done by one individual, it cannot be done by "the collective" because only individuals act. Thus even when striving for the end of common ownership of the means of production, ultimately there has to be one individual to actually manifest the action. Without clear individual property rights, the only other option is tyranny by a master class who has full control over the means of production. Communism neglects the need to allocate scarce resources throughout time, and thus inevitably will lead to aggression against individual, the destruction of capital and the overall decrease of individual subjective value scales.[14] The ideal of sharing goods with others, although logically infeasible in the realm of scarcity, is not just doable, but desirable with non-scarce goods.

13 See Marx, Critique of the Gotha Program. 1875.

14 See Mises, Socialism: An Economic and Sociological Analysis. 1951.

7. Non-Scarcity

The polar opposite manifestation of scarcity is non-scarcity, goods that can be copied ad infinitum without degrading the quality of the original.[15] Due to this super-abundance these good are not subject to desire and choice, as they exist in superfluity, they gratify and also satisfy all desires which depend on their use.[16] They are non-exclusive, thus anyone who desires access and use, can gain this without taking it from another.

When based on her previous experiences Alice formulates a thought, and speaks this into existence to Bob, this information forces itself into Bob's possession. He can now contemplate what Alice is trying to convey and act upon the information. Bob has clearly gained possession and usage of a good, this can help him remove uneasiness and thus increase his subjective valuation. He only sacrificed the opportunity cost of listening to Alice, yet he may gain tremendously by this new knowledge. Alice maintains her "original" idea in mind, she can further think on it and share it with others and nothing has been taken away from her. Rather, she can now accumulate new information based on Bob's reaction and use this input to advance and refine the idea. *"When speaking words, they can be taken all to oneself, yet leave all to others and unless the memory fades away, everyone who can hear those words, can take them all and go on each separate way."*[17]

15 See Kinsella, Against Intellectual Property. 2008.
16 See Fetter, Economic Principles, Chapter 1, §3. 1915.
17 Wills, St. Augustine, p. 145. 1999.

8. Libre Open Source Software

Early cypherpunks had a thorough understanding of the axiom of scarcity, and they strived to create a realm in cyberspace where non-scarce knowledge can be shared freely and without permission. Nobody could possibly own the 0's and 1's that make up the computer code, just as nobody could exclusively own a certain pattern of words or a specific number. Especially digital data can be copied at high speed with negligible computing costs, and shared across a global network of nodes. Anyone desiring access to a certain set of data can request a copy without decreasing the quality of the dataset of the originator. Should they add to this data the original author is free to request a copy of the expanded data set, which if granted results in a marginal benefit for the original author at little to no cost.

The rational conclusion of the non-exclusivity of software is that the user shall have full access to the source code, and that nobody has the right to aggress against him for copying and adapting the code. *"'Free Software' means that the user has the freedom to run, copy, distribute, study, change and improve the software."*[18] This implies the ability to run the program in whatever way possible and for whatever purpose, since regardless how the software is used, this is of no concern to the author. Changing the individual implementation of the software does not meddle with the copy of other peers, thus there is no harm in forking the code to solve a different task to that originally intended. The source code must be open and accessible in order for the user to study and verify what the computation is doing. This includes the ability to share the knowledge with whoever may request it, both the original, and the forked version. Only when the improvements to the code are made public can everyone benefit from them by updating their code, this is the right to learn and share what one has learned with others.

18 Gnu Project, What is Free Software.

9. Fallacy of Intellectual Property

As there is no potential conflict of control, there is no need to organize the structure of production with these non-scarce goods, because any entrepreneur who would need the good to advance the process could simply copy it. There does not need to be a direct exchange, because the original creator does not give up anything, he still retains his version without sacrifice. *"But sharing isn't immoral — it's a moral imperative. Only those blinded by greed would refuse to let a friend make a copy."*[19] *“These designs – the recipes, the formulas, the ideologies – are the primary thing; they transform the original factors – both human and nonhuman – into means.”*[20]

Because there is no need to ration the allocation of non-scarce goods, property rights are not necessary. There is no individual ownership of ideas, recipes or music, rather, anyone who is interested can acquire and use this information without taking it from someone else. Information belongs in the universal field of knowledge from which any individual can draw everything needed to understand the truth and apply it in one's life.

There are currently several different types of "intellectual property" [patents, licenses, non-disclosure agreements, ...]. They all claim that the "creator" of a specific idea, recipe or thought is to be the sole beneficiary of it. Anyone who is using this idea on its own, without the explicit consent of the "original thinker" is breaking their property rights and thus punishment is justified and desired.

This line of reasoning is flawed on a fundamental level; all forms of intellectual creation are per definition non scarce, that is, when the information is shared with others, the "original" producer does not sacrifice the enjoyment of the thought. Precisely because there is no need for resource allocation, there is no need for property rights. IP thus attempts to introduce artificial scarcity in a place where nature has granted us non scarcity! It is aggression against the possibility of sharing new knowledge with anyone who needs it, without taking anything from anyone else. The knowledge differential in the

19 Swartz, Guerilla Open Access Manifesto. 2008.
20 Mises, Human Action, p. 142. 1949.

hierarchy of peers increases and is thus more prone to attacks. This is an unnecessary limit on the prosperity which humans can achieve.

Furthermore, following this bogus claim to establish property rights where we do not need them, innocent individuals legitimate property rights get violated. Suppose Alice formulates the idea to bake an apple cake, and she registered her IP claim. When later Bob independently formulates the same recipe he has not taken anything from Alice. Because Alice has the power of the State, she can enforce her IP claim and steal the cake from Bob. She has no property right whatsoever in the goods and services that Bob has produced on his own, yet with IP, she can justify her aggressions against a peaceful individual.

"Natural scarcity is that which follows from the relationship between man and nature. Scarcity is natural when it is possible to conceive of it before any human, institutional, contractual arrangement. Artificial scarcity, on the other hand, is the outcome of such arrangements. Artificial scarcity can hardly serve as a justification for the legal framework that causes the scarcity. Such an argument would be completely circular. On the contrary, artificial scarcity itself needs a justification."[21] Thus, any form of "intellectual property", be it patents, copyrights or trademarks are completely unjustifiable monopolies of state aggression, privilege and censorship. They are evil to its fundamental core, since it introduces an artificial limit to the potential prosperity humankind might achieve, at the benefit of only a few, but at the expense of many. Those that violently enforce unjust intellectual property, assert control and ownership over someone else's property in scarce resources.

21 Bouckaert, What is Property? p. 793.

10. Non-Scarcity of Cryptography

A cryptographic private key **t** is a very large random number, a piece of non-scarce information which can be copied endlessly without degrading the original. Anyone has the opportunity to independently discover this particular number, it is impossible to exclude others from doing so. Nobody can exclusively use, and thus own, a specific number, thus there are no property rights whatsoever in private keys. In cryptography a sufficiently random number can be gained by throwing dice, picking random pages of a book, or utilizing a cryptographically secure random number generator. Because the number field of 2^256 is so large, when one sufficiently random number is picked it can be assumed that nobody else has knowledge of this specific information.

Whoever has the knowledge of this private key can easily compute a corresponding public key **X** by using a cyclic group **G**, and a generator **g** of **G** to calculate **X = g^t**. Yet with knowledge of only the public key, it is computationally infeasible to reverse this operation and calculate the private key. Thus the public key can be shared with others, without revealing any part of the private key itself. In the Pretty Good Privacy protocol[22], a static master public keys is used as a long term identity of the key holder. In Bitcoin however, the public key should be used only one time for one payment, and never reused across transactions.[23]

To sign a message **m**, a random number **r** and a corresponding nonce **R = g^r**, as well as a hash **c = H(X,R,m)** are computed. The signature **σ** is the tuple **(R,s)** with the nonce **R** and **s = r + cx**.[24] A signature can thus only be produced with knowledge of the private key **t** and the random nonce **r**, which are both generated at random in a huge number field. The verifier of the signature need only have knowledge of the public key **X**, the nonce commitment **R** and the part of the signature **s**. Only when the calculation of **g^s = RX^c** returns valid, can the signer have provable knowledge of the private key.

22 Zimmermann. Pretty Good Privacy Freeware Software. 1991.
23 Belcher. Bitcoin Wiki: Privacy, Address Reuse. 2018.
24 Schnorr. Efficient Signature Generation by Smart Cards. J. Cryptology, 4(3):161–174, 1991.

Asymmetric cryptography assumes that the creator of the private key can keep these bits hidden and occulted from anyone else. Only when this knowledge is exclusively available to the original creator is the signature a conclusive proof of the identity and intent of the original signer (the creator). However, when some other party copies the non-scarce private keys, he can easily compute an absolutely valid signature, that was not made by the original creator of the keys. It is extremely difficult to keep a private key in the exclusive control of one individual, thus in order to ensure a stable and working protocol, the secure storage and management of private keys is of utmost importance.

11. Scarcity of UTXOs

A Bitcoin unspent transaction output [UTXO] can only be spend when the corresponding redeem script is returned valid, these conditions are expressed in the non-Turing-complete Bitcoin script language. At any time, a UTXO has only one script which commits to the spending conditions, the property boundry definitions of that bitcoin. Thus, there is a potential conflict over who can use this UTXO, it's either the script of Alice, or that of Bob. For example a pay-to-witness-public-key-hash [P2WPKH] UTXO can only be spent by he who has the knowledge of the committed private key and proves this with a valid signature. If a transaction is proposed with a wrong signature, then the script computes invalid, and the UTXO is thus not advanced to the next script. Possession of the non-scarce information is sufficient to use the absolutely scarce bitcoin. Although nobody owns information of the private key, its knowledge grants the right to own and use this specific coin. This excludes all those without the private key from using the UTXO, creating a potential conflict of control. Thus there is a need for resource allocation of the coins, which is done with the property rights defined in Bitcoin script.

Because the private key can be shared with others without taking the knowledge from someone else, the access rights to the bitcoin can also be shared. Multiple individuals can have knowledge of the same secret, and thus they have the means to provide a valid signature proof. However, with the single key P2WPKH script, only the first individual to broadcast a valid transaction (and have it committed to the time chain) has ultimate control over the bitcoin on chain. Many peers have potential control over the coin, yet only the first to act has the ultimate ownership of it. Thus, sharing the same private key with others is only a weak simulation of shared ownership.

Pay-to-witness-script-hash [P2WSH] transactions commit to more advanced scripts that can add complexity to the conditions that the spender needs to prove. Such a script could be a multi signature scheme, where **n** private keys are generated individually by different peers. Each peer has exclusive knowlege of their specific private key, and they

compute and share the corresponding public key with their peers. A multi signature redeem script includes all of the **n** public keys, as well as the threshold number of **m** signatures required in order to spend the coin. **N** individuals can create their own unique private keys, however this piece of information alone is worthless, as it cannot create a valid signature script by itself. Only with the coordination of **m** individuals can the chain of digital signatures be advanced. This is non-simulated shared ownership of the scarce bitcoin which is cryptographically proven and cannot be broken.

11.1. Double Spending is Non-Scarcity

A double spend is the aspect of a digital asset to be able to be sent several times to different individuals. First, Alice initiates a transaction to Bob, and later, she sends the same asset to Charlie; this is an asset that can be double spent, a non-scarce good that is non-exclusive. The main issue is to find a common state of the most recent property right definitions, this challenge can be seen to be a narrow version of the Byzantines Generals Problem.[25] Bitcoin solves this computer science problem in a decentralized trustless manner.

The ability to spend the same good twice means that Alice can give a good to Bob, without sacrificing the possession and usage of that good. Thus, this good is non scarce and does not require property rights to allocate resources. Any non scarce good can thus be double spent, while this is impossible for any scarce asset.

11.2. Bitcoin Halving and Scarcity

Every 210 000 blocks, the issuance rate of new bitcoin in the coin base transaction is halved from the original 50 bitcoin reward. Full nodes will not allow any block that has a coin base reward larger than the halved amount. This means that over time, the stock to flow ratio increases exponentially, until it reaches infinity in the year 2141. It is important to differentiate that this does not at all affect the exclusiveness of bitcoin. Regardless the quantity of total money supply, one UTXO can only be spend by the one defined script. This is true in the case of a total money supply of 50, 21 million or 84 billion bitcoin. The

25 Lamport, Shostak, Pease. The Byzantine Generals Problem. In ACM Transactions on Programming Languages and Systems, July 1982. pp.382-401.

axiomatic importance is the fact that one UTXO can only be spend by one script, and not two different scripts at the same time. In order to enable holistic scarcity of bitcoin, there needs to be both the exclusivity of UTXOs as well as a limited supply of bitcoin to justify the need for resource allocation through property rights. Scarcity is what is needed in order to make a monetary asset possible in the first place.

However, since money is neither consumed [that is the direct satisfaction of desires] and nor used in production [that is the building of future consumption good], a larger supply does not mean a higher satisfaction. For a medium of exchange, the total supply of money is completely irrelevant, since prices will simply adjust to reflect the market demand of holding money in percentage to the total money supply.[26]

11.3. Full Nodes Define, Verify and Enforce Scarcity

The scarcity of rare bitcoin is defined as the attribute that one UTXO can exclusively be spend by one pre-defined script. In other words, one UTXO must only be spend once, and only in the way specified by the creator of the UTXO. In previous cryptosystems this was enabled by relying on one trusted central server to authoritatively define, verify and enforce this scarcity. Yet a slave node has no opportunity to trustlessly audit and enforce the dealings of the central master server.

On the contrary in Bitcoin, even before the genesis block there was libre and open source software that clearly defined the Nakamoto consensus rules. Any peer can see, verify and edit a local copy of this particular set of rules. With the software installation every individual proves they voluntarily agree to use this consensus protocol. Further, any node will only communicate with another node who runs exactly the same consensus rules, thus everyone in the network is voluntarily in agreement of the property rights definition of the scarce UTXOs.

Every time a full node receives a new transaction or block, it runs several checks[27] to determine if this transfer of ownership of the UTXO is valid. For example whether the input coin of the transaction is actually in the UTXO set of the chain with the most

26 Hillebrand. Anarchy in Money: Money Supply and Inflation. 2018.
27 See file /bitcoin/src/consensus/validation.h in the Bitcoin Core software.

accumulated proof of work. In the case that this particular coin has already been included in a valid confirmed transaction earlier in the time chain, then the sender is trying to double spend this coin. This is an attempt to use the same coin twice for different transactions, to break the scarcity of this UTXO. Every full node will detect and block any such double spend transaction. Notice that even if the witness program would be valid, an already spent coin will never be allowed to be spend again.

Although this is scarcity defined and enforced by imperfect software, written and run by imperfect humans, there are several reasons why this simulation of scarcity is so incredibly strong, that it might even be considered non-simulated. There has never been an invalid transaction confirmed in the most accumulated proof of work chain, the scarcity was never broken. The protocol is currently running at 99.984% uptime[28], with only two downtime events in 2010[29] and 2013[30]. Currently it seems there is no active exploitation of critical bugs. Because the Bitcoin Core software has a very rigorous peer review process for any change to the code base, new bugs will hopefully be found and fixed before merge. Of course software, as any other form of speech, is never perfect and the possibility of critical exploits is ever present. Yet Bitcoin is a high assurance protocol with a close to perfect uptime and no invalid transaction confirmation, thus it can be said that the property rights are not simulated, but actually enforceable without any counter party risk.

28 Calculated by (the number of actual blocks since Jan 3 2009 02:54:25 GMT) divided by (the number of expected blocks given a 10 minute block time since genesis).

29 8 hours, 27 minutes downtime due to CVE-2010-5139

30 6 hours, 20 minutes downtime due to CVE-2013-32220

PART II: SHARED OWNERSHIP OF SCARCE BITCOIN

1. Basics of Bitcoin Script

A bitcoin UTXO is the tip of the chain of digital signatures which originates in the coinbase issuance transaction. A transaction dedicates inputs, which are previously unspent transaction outputs with a valid witness to the ScriptPubkey; and outputs, which "lock up" the bitcoin with a new ScriptPubKey. The Bitcoin script is the property rights definition of how a UTXO can be spend. At any time, there is only one script for one UTXO, the spending conditions are defined at the point where the UTXO is created.

1.1 Pay to Public Key Hash

A ScriptPubKey is a short script that details the conditions which must be met in order to claim control of the bitcoin. In a P2PKH transaction, this script contains **`DUP HASH160 <PubKey> EQUALVERIFY CHECKSIG`**, the UTXO can only be spend by the proof of knowledge of the private key corresponding to the committed public key hash. The signature script is a prefixed to the ScriptPubKey and contains an secp256k1 signature and the full public key: **`<Sig> <PubKey> DUP HASH160 <PubkeyHash> EQUALVERIFY CHECKSIG`**. To verify the correctness of the transaction, all operators from both the Alice's signature and Bob's pubkey script are executed. The message to sign is the hash of certain parts of the transaction it self. So not only is the signature proof of knowledge of the private key, it also gives the explicit consent which input is consumed, and which outputs are created.

Multi Signature in P2PKH

ScriptPubKey: **`M <Public Key 1> <Public Key 2> ... <Public Key N> N CHECKMULTISIG`**

SignatureScript: **`<Signature 1> <Signature 2> ... <Signature M>`**

Validation Script: **`<Signature 1> <Signature M> M <Public Key 1> <Public Key 2> <Public Key N> N CHECKMULTISIG`**

1.2. Pay to Script Hash

The ScriptPubKey of a P2SH transaction contains a hash of a redeem script, which can be any script whatsoever.[31] At the time of UTXO creation there is only the hash of the script on the time chain, but not the script itself. It is thus unclear what the spending condition of that particular coin is. These conditions to redeem the bitcoin are with the receiver, not the sender of the transaction. To spend the UTXO, Bob needs to provide both the full redeem script and a valid signature script. It is only valid, when the redeem script hashes to the same value as the script hash Alice committed to in the output. The redeem script has the same attributes as a pubkey hash, thus the follow up process is identical to a P2PKH script. This gives the benefit of removing complex scripts from block space and shifting the burden of key storage to the future spending time. The majority of fees are paid not when receiving to a complex P2SH, but when spending the P2SH UTXO where the redeem script and signatures are revealed.

Multi Signature in P2SH

Redeem Script: **`M <Public Key 1> <Public Key 2> ... <Public Key N> N CHECKMULTISIG`**

ScriptPubKey: **`HASH160 <20-byte hash of redeem script> EQUAL`**

SignatureScript: **`<Signature 1> ... <Signature M> <redeem script>`**

Address: **`Base58Check(HASH160(<ScriptPubKey>))`**

31 Andreson. BIP 16: Pay to Script Hash. 2012.

1.3. Pay to Witness Public Key Hash

Segregated witness[32] is a protocol upgrade[33] intended to fix transaction malleability and to increase the block size through different weight calculations.[34] P2WPKH transactions remove the signature in the input field, and relocate it in the witness structure of the scriptSig. The script PubKey thus contains only **`0 <PubKeyHash>`** and the signature is verified as **`<signature> <pubkey> CHECKSIG`**.

Single Signature in P2WPKH

Redeem Script: **`<Public Key> CHECKSIG`**

Witness Script: **`0 <Public Key Hash>`**

ScriptPubKey: **`0 SHA256(<witness Script>)`**

Address: **`RIPEMD160(SHA256(<witnessScript>))`**

1.4. Pay to Witness Script Hash

Similar to P2WPKH, P2WSH removes the witness part from the transaction merke tree of the redeem script. The main difference to P2PKH is thus the location of the signature part. SegWit also increases the security for multisig compared to P2SH, which uses 160-bit HASH160 algorithm, a double hash with RIPEMD of SHA256. In the case where one signer of the multisig is malicious and wants to spend the money without the consent of other signers, he can brute force a collision between a valid P2SH multisig address and a script that pays him all the money. This requires only 80-bits, that is **`2^80`** worth of work, which is already possible for a well funded attacker. SegWit fixes this issue by using HASH160 only for P2WPKH, but 256-bit SHA256 for P2WSH.

Multi Signature in P2WSH

Redeem Script: **`M <Public Key 1> <Public Key 2> ... <Public Key N> N CHECKMULTISIG`**

Witness Script: **`0 <signature 1> ... <signature M> M <Public Key 1> ... <Public Key M> N CHECKMULTISIG`**

ScriptPubKey: **`0 SHA256(<witnessScript>)`**

Address: **`checksum base32`**

32 Lombrozo, Lau, Wuille. BIP 141: Segregated Witness. 2017.

33 Fry. BIP 148: Mandatory Activation of SegWit Deployment. 2017.

34 See Bitcoin Wiki. Weight units.

2. Script Multi Signature

In the incumbent implementation of multi signatures, the redeem script contains a list of `n` public keys, as well as the threshold of `m` necessary signatures to validate a transaction. This redeem script is hashed to the script public key and committed to the time chain at the creation of the UTXO. The hash commitment decreases the P2WSH UTXO data size at the time of funding, yet the clear text script needs to be revealed for verification at the time of spending.

In order to spend this coin, the committed redeem script and a valid witness script has to be provided, containing any `m` signatures corresponding to the committed public keys. If the proposed signatures are not valid for the public keys, or if less then `m` signatures are provided, then the transaction and block are rejected by every full node. This enforces the shared ownership of bitcoin on the level of full nodes. This verification is done both at the time the transaction is initially broadcasted, or during the initial block download of a new full node.

However, this system is fundamentally based on the trust in the thorough verification of the consensus rules as they have been defined at the point of funding. Full nodes could adopt a hard fork that changes the validity rules so that the coins can be spend in another way as originally defined. This is in contrast to aggregated Schnorr MuSig[35] where nodes verify only one signature, not knowing that this is actually a aggregated signature of several individual private keys. Since the verifier of a script multi signature has more information available, there is a higher possibility to censor this particular transaction. An adversary could define an additional rule that all script based multi signature scripts are to be dropped, yet he cannot do this for aggregated multi signatures. Script multi signatures have a smaller anonymity set.

35 See chapter on Schnorr MuSig.

However, if this concept of full node verification is working properly and reliably, as is the case in Bitcoin[36], then the redeem script can be considered a non-simulated property right definition, there is a very slim and unprecedented plausibility of breaking the spending conditions. This is one of the many reasons why the adherence to Nakamoto consensus is so vitally important, and why any change to the set of rules, including soft forks but especially hard forks, carry a tremendous risk.

36 See Chapter on Full Nodes Define, Verify and Enforce Scarcity.

3. Schnorr Signatures

The Schnorr signature scheme[37] is a conceptually simple, data size and computationally efficient and secure under the discrete logarithm assumption. This protocol is currently proposed as s soft fork replacement of Bitcoin's incumbent elliptical curve digital signature algorithm [ECDSA].

Schnorr is in nearly all regards superior to ECDSA, other than the fact that it's use requires a change to Nakamoto consensus as well as the lacking documentation and implementation.

[i] The signature is constant-size, regardless the number of participants in the multi signature, only one aggregated public key and signature needs to be verified. This allows for a large group of peers to define and enforce secure, private and efficient shared ownership of scarce bitcoin. The nuances of aggregated MuSig is explained in detail below.

[ii] Since the data size of redeem and witness script are overall smaller, this also means that the bandwidth usage of the peer to peer gossiping protocol is reduced. Every transaction is shared with a default of 8 peers, thus the bandwidth constrains are more pressing than the computation or storage of the data.

[iii] Due to the linearity of Schnorr, there can be entire spending conditions and policies included in the public key and signature itself, obscured and indistinguishable from regular single public key and signature. This means that a single signature spend, a MuSig spend, a taproot cooperative spend, a lightning network payment channel closing, or an adapter signature coin swap, all of them require the same form of information to validate - only one single public key and signature. The signature can be valid if and only if all the spending conditions are met, the details of the property rights definition don't necessarily matter, rather it's validity is essential.

37 Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. J. Cryptology, 4(3):161–174, 1991.

[iv] A verifier can leverage the linearity of public keys and signatures in order to batch validate them, it is computationally easier to batch and verify many signatures at once, rather than to verify them individually one after another. For example all Schnorr signatures in a Bitcoin block can be batched together and if the batched single check is valid, than all the individual signatures must be valid as well. Yet in the case that one signature is wrong, the batch verification will also be invalid, and the block will be discarded.

[v] In an advanced implementation of Schnorr signatures to the Bitcoin protocol, interactive cross-input signature aggregation can drastically reduce the size of transactions with many inputs. This transaction doesn't need to have one signature for each input, but one single signature for all inputs. This aggregated signature is only valid if all the signatures of each individual input are valid. This would allow for example for a large coin join transaction with only one signature, which would be much more efficient than a one input - one output transaction.

The Schnorr signature scheme uses a cyclic group **G** of prime order **p**, a generator **g** of **G**, and a hash function **H**. It uses a random number private key **t**, and public key **X**, with **(t, X) ∈ {0, …, p-1} * G** where **X = g^t**. To sign a message **m**, the signer generates a random number integer **r** in **Zp** and computes the nonce **R = g^r, c = H(X,R,m)**[38], as well as **s = r + cx**. The signature **σ** is the tuple **(R,s)** and this can be verified by **g^s = RX^c**.

Just like ECDSA, the Schnorr signature scheme is proven secure under the hardness assumption of the discrete logarithm, defined as followed. Let **(G,p,g)** be group parameters. An algorithm **A** is said to **(t,ε)**-solve the DL problem w.r.t. (**G,p,g**) if on input a random group element **X**, it runs in time at most **t** and returns **x ∈ {0, ..., p − 1}** such that **X = g^x** with probability at least **ε**, where the probability is taken over the random draw of **X** and the random coins of **A**.[39]

38 The key-prefix method with the hash of _R_ and _m_ as described by Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-Speed High-Security Signatures. In Bart Preneel and Tsuyoshi Takagi, editors, Cryptographic Hardware and Embedded Systems – CHES 2011, volume 6917 of LNCS, pages 124–142. Springer, 2011.

39 See Maxwell, Poelstra, Seurin, Wuille. Simple Schnorr Multi-Signatures with Applications to Bitcoin. [From now, just MuSig] 2018. Chapter 2.1. Notation and Definitions. Notations for this definition are

4. MuSig

The MuSig paper[40] describes a simple and efficient multi-signature scheme based on Schnorr. Some of the benefits are key aggregation, signature aggregation and batch verification. The paper includes a security prove in the plain public-key model[41], which is omitted in this paper.

MuSig is parameterized by group parameters **(G,p,g)** where **p** is a **k**-bit integer, **G** is a cyclic group of order **p**, and **g** is a generator of **G**, and by three hash functions.[42] The total signature size is **|G|+|p|**; the public key size **|G|**; and the private key size **|p|**.

4.1. Interactive Key Generation and Aggregation

Individual private keys **t_i** are generated by each cosigner with a true random number generator and the public keys **X_i** are computed with **X_i = g^t_i**. The **X_1** and **t_1** are individual keys of a specific signer; **X_2, …, X_n** are the public keys of the cosigners; and **L = {pubk_1 = X_1, …, pubk_n = X_n}** is a multiset of all public keys. In the first round of communication, all cosigners share their public keys, any verifier can build the multiset **L** and for all **i ∈ {1, …, n}**, the signer computes **a_i = Hagg(L,X_i)** and then aggregates all the individual public keys into the single "aggregated" public key **X́ = X_i for 1 <= X_i = n, X́ = product of X_i^a_i**. The the hash pre-image of **a_1** contains all the public keys once, but **X_1** twice.

The aggregated public key **X́ for 1 <= X́_i = 1 <= n, X́ = product of X_i^a_i** is indistinguishable from any other Schnorr public key. If only **X́** is known, then the individual public keys **X_i** cannot be computed. Thus, the on-chain commitment to this MuSig is the exact same virtual size as any other public key commitment. Therefore, MuSig is both a privacy and scalability improvement. Further, anyone with knowledge of

adapted from the paper, and do not match the usual notations of this thesis.

40 See MuSig 2018.

41 See MuSig, Chapter 4. Security of the New Multi-Signature Scheme. 2018.

42 See MuSig. Chapter 3. Our New Multi-Signature Scheme. 2018.

all the public keys **X_i** can compute [and thus send bitcoin to] this aggregated public key **X́**, without collaboration from the peers.

Each individual signer has sole knowledge of the non-scarce information of the private key. Assuming that this secret is not shared with others and generated with a cryptographically secure random number generator, then only this individual can produce a signature that is valid for the given public key.

4.2. Interactive Signing

The signer has knowledge of aggregated **X́**; the message **m** (in the context of Bitcoin **m** is the transaction according to the SIGHASH flag); and the multiset **L**. He generates another random integer **r_1** and computes the nonce of **R_i for 1 <= R_i <= n, R = product of all R_i**, and the commitment to that nonce **t_1 = H_com(R_1)**. The commitment **t_1** is shared with all cosigners, then in the next round the nonce **R_1**, and we proceed with the protocol only if all **R** have been correctly committed for all **t_i = H_com(R_i) with i ∈ {2, …, n}**.

The signer computes **R for 1 <= R_i <= n, R = product of all R_i**, **c = H_sig(X́,R,m)**, and **s_1 = r_1 + ca_1x_1 mod p**, **s_1** is send to all cosigners. After all **s_2, …, sn** have been received, the signer computes let **s for 1 <= s_i <= n, s = sum of all (s_i mod p)**. The signature is **σ = (R,s)**.

Although there is one aggregated public key **X́**, there is no "aggregated private key". In order to produce a valid signature while defending against the rogue key attack[43] [44], all cosigners have to collaborate in a three step[45] signing ceremony. First, sharing a nonce commitment **t_i**, then the nonce **R_i**, and finally the partial signatures **s_i**. Only when all **i** partial signatures are available can the coordinator produce the valid signature **σ** which contains the aggregated nonce **R** and **s** part of the signature. If one cosigner is unavailable to communicate the signature, then there can not be a valid signature.

43 Ristenpart, Yilek. The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks. In Moni Naor, editor, Advances in Cryptology - EUROCRYPT 2007, volume 4515 of LNCS, pages 228–245. Springer, 2007.

44 See MuSig. Chapter 5.3. Cross-Input Multi-Signatures. 2018.

45 whilst a two-step round would be possible, it is larger in signature size and computational cost of signing and verification.

Only those who have securely generated the individual private key can produce a valid individual signature over a message with very little effort. Without the knowledge of the private key, it is computationally infeasible to produce a correct signature. Once the signing algorithm is calculated, it cannot be undone, as the specific information of the signature is manifested. However, when the signature is not shared with others, nobody can verify it.

4.3. Verification

The verifier has a multiset of public keys **L**, a message **m**, and a signature **σ**. With this public information, the verifier computes **a_i**, **X́** and **c**. The signature is valid only if **g^s = R <= R = 1 <= n, R (product of X_i^(a_i c)) = R X́^c**. Due to key aggregation, the verification is similar to the standard Schnorr scheme, and secure variants of the MuSig scheme are discussed in the original paper[46].

When given a Bitcoin transaction as a message as well as a signature, then any full node can verify conclusively that the signer had knowledge of the private key and the intend to send this specific transaction. According to Nakamoto Consensus, this means that an existing UTXO can be spend and a new UTXO is created. The transaction will be included in a block of the time chain.

Since the aggregated public key and signature look identical to an individual public key, the verifier knows only that [all of] the signer[s] has [have] agreed and collaborated with that signature and thus the spending of the bitcoin, but he does not know whether this is only one single key pair, or several key pairs in aggregation. Further, this single public key and signature could be a collaborative taproot[47] or graftroot[48] transaction, a collaborative lightning network channel close, or a scriptless script atomic coin swap[49]. This plausible deniablity is a great enhancement to the fungibility of UTXOs and strengthening Bitcoins overall privacy aspects. Although lots of the spending logic is abstracted from the time chain, yet every full node can still verify absolutely if the

46 See MuSig. Chapter 4.3 Discussion. 2018.
47 Maxwell. Taproot: Privacy preserving switchable scripting. Bitcoin-dev mailing list. Jan 23 2018.
48 Maxwell. Graftroot: Private and efficient surrogate scripts under the taproot assumption. Bitcoin-dev mailing list. Feb 05 2018.
49 Poelstra. Scriptless scripting and deniable swaps. Mimblewimble team mailing list. Feb 03 2017.

spending condition, whatever it might be, is completely valid. There no false positives or negatives, a UTXO can only be spend with a valid witness script.

Contrarily to the script based multi signature, in Schnorr MuSig only one aggregated public key is committed to the time chain, and a valid signature can only be computed when all **m** signers collaborate on the shared message. Without any further detail than the aggregated public key and signature, any full node can verify if the spending attempt is valid or not. There are no additional security and node verification assumptions compared to any other single signature transaction.

4.4. Non-Simulated Shared Ownership

In a Schnorr **3-of-3** MuSig ceremony, Alice Bob and Charlie each generate an individual non-scarce private key, which only they have the knowledge of. They compute and exchange public keys and concatenate them into one single aggregated public key. Although each individual can produce a valid individual signature with their individual private key, a valid aggregated signature to the aggregated public key can only be produced by all three individual signatures over the same message. Thus one aggregated signature is cryptographic proof, that all **n-of-n** individual private keys have been known and have given active consent to the transaction.

Assuming the discrete log problem, there is no computationally feasible way to fake a signature without the knowledge of the private key. When a full node receives a valid transaction with a valid Schnorr signature, it has cryptographic proof that the committed script is computed valid. Thus the transaction is included in the time chain with the most accumulated proof of work, the chain of digital signatures is advanced and a new UTXO with a new spending condition is created. The transfer of the UTXO is thus irrefutable and censorship resistant, it is a true ownership exchange. And since the MuSig transaction is only valid when all **n-of-n** peers agree, this is non-simulated shared ownership over a scarce bitcoin.

5. Taproot

Taproot is a proposed variation on the current script language to add a BIP-Taproot[50] Merkel spend. Taproot is a clever usage of aggregated Schnorr signatures and Merkelized abstract syntax tree [MAST][51]. This enables a drastic increase in the complexity of potential spending conditions, since only the one script that is actually used to move the coins is revealed to full nodes on the time chain. This allows the writing of very complex scripts while still minimizing their data size for efficient and private usage of block space. A taproot bech32 address contains the public key directly, and not the hash of the public key as in incumbent P2WPKH addresses. Therefore a taproot spend does not require to reveal the public key when the UTXO is consumed. A valid transaction needs to contain a Schnorr signature [64 bytes / 16 vbytes] according to BIP-Schnorr[52]. In total, the cost of creating a taproot UTXO is roughly similar to sending to a P2WSH, yet spending a single-key taproot is 40% cheaper than P2WPKH.

[in Vbytes]	P2PKH	P2WPKH	Taproot
scriptPubKey	25	22	35
scriptSig	107	0	0
witness	0	26.75	16.25
total	132	48.75	51.25

Data size of different scripts, by David Harding for the Bitcoin OpTech Group[53]

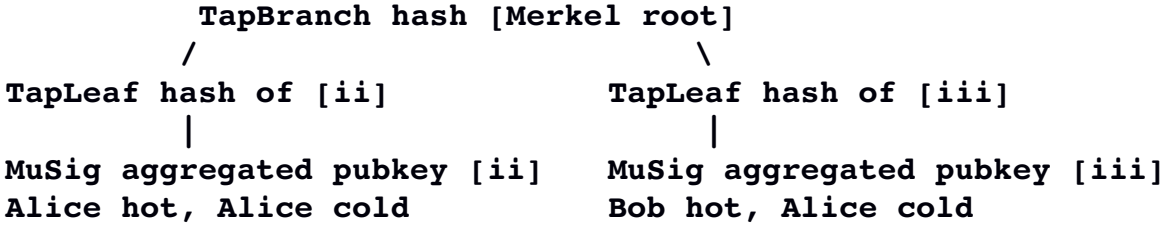
5.1. m-of-n Threshold signatures using Taproot

Schnorr MuSig aggregation is very efficient and private for **n-of-n** interactive signers, but the taproot concept can be used to add more complexity into the spending condition script, while retaining some privacy and efficiency. For example, a **2-of-3** multi signature security hot wallet, where Alice has two keys, one hot and one cold storage, and Bob as a second factor security expert knows the third hot key. The most common use is [i] the combined signature of the hot keys of both Alice and Bob. In case

50 Wuille, Nick, Petukhow. BIP-Taproot: SegWit version 1 output spending rules. 2019.
51 Rubin, Naik, Subramanian. Merkelized Abstract Syntax Trees. 2016.
52 Wuille. BIP Schnorr: Schnorr Signatures for secp256k1. 2019.
53 Harding, Single-sig spending using Taproot. Bitcoin Optech Newsletter #46. 2019.

[ii] Bob is malicious, Alice retrieves her cold storage key and now has two signatures to spend the money. But in case [iii] where Alice's hot wallet key is compromised, she can use the cold storage wallet and Bob as second factor to spend the coins.

For incumbent script multi signature, each full node would verify in parallel that at least two valid signatures from any of three public keys are provided. Schnorr MuSig will generate a valid signature only if **3-of-3** individual signatures have been made. Schnorr Threshold signatures can actually be used to create a **2-of-3** condition. Yet we can achieve the same result with taproot, by utilizing a different intuition. Instead of a spending condition of **2-of-3**, we build three individual scripts of each a **2-of-2** multi signature. Incumbent script multisig would work for these internal spending conditions, but for efficiency, let's work with three independent aggregated Schnorr public keys, that can only generate a valid signature if **2-of-2** individual private keys sign. The three pairs are [i] Alice hot and Bob hot [the most common case], [ii] Alice hot and Alice cold [Bob is malicious], or [iii] Bob hot and Alice cold [Alice hot key compromised]. The uncommon cases [ii] and [iii] are hashed and put in lexicographic order as the TapLeafs of the MAST. These two hashes are then hashed again to calculate the TapBranch, the Merkel root of the tree.



For the cooperative common case [i], Alice and Bob create another Schnorr MuSig aggregated public key, the taproot internal key. Then, TapBranch and the taproot internal key are hashed together, resulting in a tweak private key, used to calculate the tweak public key. The tweak public key is added to the taproot internal key which generates the taproot output key and used in the bech32 address committed in the time chain. This taproot output key has two spending options, the cooperative key path, or the advanced script path. In the cooperative case all peers can calculate individual and aggregated signatures that validate to this taproot output pubkey. But the output key also commits to a the TapBranch Merkel root, and in the advanced case, it can be verified that the

proposed script was part in that MAST, and thus a valid spending condition defined at the time of funding the UTXO.

```
                Merkel root [hash]   =\ Tweak Hash
Alice pubkey =\ Taproot internal key =/
Bob pubkey   =/ [Aggregated MuSig pubkey]


Tweak Hash => Tweak prkey [32-byte integer] => Tweak pubkey


Tweak pubkey           =\ Taproot output key
Taproot internal key   =/ [pubkey on time chain]
```

For spending this taproot UTXO in the cooperative case [i], Alice and Bob calculate a valid signature aggregated with the tweak private key [including the Merkel root of the unused spending conditions [ii] and [iii]] and taproot internal key. Full nodes will only see the committed taproot output key and a valid signature for it, they do not know that this was a MuSig, or even a taproot. When using spending condition [ii] or [iii], then the spending transaction includes the script they want to use, the data needed by it [in our case only the aggregated public key and aggregated signature], the taproot internal key and the hash of the TapLeaf script not used. In the sub-optimal case, it has to be revealed that the script in fact is a taproot, yet only the spending condition actually used is revealed, not the many other scripts that could have potentially been used to spend the UTXO. The maximum depth of the tree is 32 rows, which would allow for over four billion possible scripts, yet only one has to be revealed and verified. But for any **`m-of-n`** there need to be **`n!/((m!(n-m)!)`** TapLeafs specified to express all the possible combinations of m signatures.

6. Shamir's Secret Sharing Scheme

Shamir's Secret Sharing [SSSS][54] is an algorithm used to divide a given master secret **MS** into **n** parts, such that **m** parts are required in order to compute the original master secret. If only **m-1** parts are available, no information about the master secret is revealed. If the **m-of-n** threshold scheme is **n = 2m-1** then we can still recover **MS** even if **n/2 = m-1** of the **n** pieces are destroyed. However, an adversary cannot reconstruct **MS** even when he has compromised **n/2 = m-1** parts.

SSSS is based on polynomial interpolation: given **m** points in the 2-dimensional plane **(x_1, y_1) … (x_m, y_m)** there is only one function **q(x)** of degree **m-1** such that **q(x_i) = y_i** for all **i**. In order to protect against the attacker acquiring information about **MS** with every additional **D_i**, we use finite field arithmetic with a field of size **p ∈ P: p > MS, p > n**. Prime number **p** must be close enough to the desired security level, because a too large **p** leads to long cypher text, but a too small **p** leads to compromised security.

6.1. Preparation

After specifying **MS**, **m** and **n**, we generate **m-1** random numbers **a_1, … a_[m-1]** and build a polynomial with the secret as **a_0**. The polynomial is thus **q(x) = a_0 + a_1*x + a_2*x^2 + … + a_[m-1]*x^[m-1]**.

Then we construct **n** points **D_[x-1] = (x, q(x) mod p)** from the polynomial and each party gets a different point [both **x** and **q(x)**], the **MS** is **q(0)**. Each sub-secret is a point **n** on the constructed polynomial curve.

6.2. Reconstruction

To reconstruct **MS**, any **m** of **n** will be enough to compute the entire polynomial **q(x)** with the Lagrange interpolation formula[55].

54 Shamir. How to Share a Secret. Communications of the ACM, Volume 22, November 1979.
55 Hazewinkel, Michiel. Lagrange interpolation formula. Encyclopedia of Mathematics, Springer Science+Business Media B.V. 1994

6.3. Simulated shared ownership

SSSS can distribute the knowledge of a secret across several different sub-secret, where each of the holders has full knowledge of his individual part, yet alone he has no knowledge of the master secret. However, the dealer first generates a master secret, which he has full knowledge off. Thus the dealer has full access and property rights in the funds locked up by the master secret. The sub-secret holders have a simulated shared ownership, however, they rely on the good will of the dealer to not spend the funds on his own accord. The use case for SSSS is thus more to backup a private key among semi-trusted peers, but where the dealer and owner of the bitcoin has always full control himself.[56] This is a vitally important differentiation compared to secure key and signature aggregation[57], which generates non-simulated shared ownership.

6.4. Verifiable Secret Sharing Scheme

Verifiable Secret Sharing Scheme [VSS] is used to prevent the dealer from cheating, every peer can verify his own share and will detect when the dealer has distributed inconsistent shares.[58] This removes some, but not all of the trust in the central dealer.

The dealer specifies **MS ∈ Z** and a random number **MS' ∈ Z** and commits to them by publicly releasing **C_0 = MS*G + MS'*H**. Then he chooses at random the polynomials **f(u) = MS + f_1 u + ... + f_x+1 u^x-1** and **f'(u) = MS' + f'_1 u + ... + f'_x+1 u^x-1** to compute **(s_i, s'_i) = (f(i), f'(i)) for i ∈ {1, ..., n}**. The tuple **(s_i, s'_i)** is send secretly to player **P_i** for **1 < i < n**. Now the master dealer publicly commits the values **C_j = f_j*G + f'_j*H for 1 =< j =< x-1**.

Then each player **P_i** verifies that **s_i*G + s'_i*H = for x-1 <= j = 0 <= sum of i^j*C_j**, if this is false, the dealer is publicly accused and he can defend himself by revealing the value **(f(i), f'(i)**. The dealer is rejected if there are more than **m** complaints, or if his defense does not validate the equation.

56 See Rusnak, Kozlik, Vejpustek, Susanka, Palatinus, Hoenicke. SLIP 0039: Shamir's Secret-Sharing for Mnemonic Codes. 2017.

57 Refer to chapter II.3. on Schnorr.

58 Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. Lecture Notes in Computer Science (Crypto '90), 473:331-238, 1991.

7. Schnorr Threshold Signatures

A threshold signature scheme[59] is setup by n individual public keys, and it computes valid only with proof of knowledge of m private keys. This protocol uses in part MuSig and Verifiable Secret Sharing Scheme.

7.1. Key Generation

All **n** signers compute their individual private public key pairs, and they use a **m-of-n** verifiable secret sharing scheme[60] to generate **n** shares of their individual private key, so that given **m** shares the individual private key can be calculated. Each of the **n** participants then give each peer one specific share, so that all peers have one share each of all the private keys of all participants. Due to the linearity of the Schnorr signature scheme, these shares can be added, that is tweaked, to the individual private key. All participants broadcast their individual public keys, so that an **n-of-n** aggregated public key **X́** can be calculated and used as the locking script of a UTXO.[61]

7.2. Signing

In order to produce a valid signature, at least **m** participants need to collaborate. Each of them signs a spending transaction with the individual tweaked private key, which is the sum of their individual private key and all **n-1** shares of the other individual private keys. All **m** individual signatures are then aggregated to the final signature. This includes the **m** "full" signatures of each active signer, and **m** shares of the signature of the **n-m** non-signing private keys. Because **m** shares are enough to produce the full signature for the non-signing keys, this final signature is a fully **n-of-n**, and thus valid according to regular MuSig.

59 Stinson, Strobl. Provably Secure Distributed Schnorr Signatures and a (t,n) Threshold Scheme for Implicit Certificates. Certicom Corporation, 2001.
60 Refer to chapter II.6. on Shamir's Secret Sharing Scheme.
61 Refer to chapter II.4. on MuSig.

8. Lightning Network

The lightning network consists of individual peers communicating information and trustlessly exchanging bitcoin between each other without requesting the verification of all Bitcoin full nodes. An additional piece of software, a lightning network node, has to be installed and run by the user. Each node can communicate with different peers to gain necessary information about the state of the network. In order to send bitcoin between lightning peers, two nodes collaboratively open, update and close a payment channel. This limits full node verification to two on-chain transactions in the life cycle of a payment channel, which can conduct potentially unlimited amounts of updated commitment transactions. These are based on the Bitcoin scripts multi signatures and hashed time locked contracts, as well as per-signed revocation transactions. A payment can be routed through many independent channels, so even peers without a direct channel can send and receive bitcoin. The following chapters focus on the shared ownership of payment channel, and not on the routing between the channels since this is independent from the channel update mechanism.

8.1. 2-of-2 Lightning Network Payment Channel

The basic implementation of the current payment channels is based on a **`2-of-2`** script multisig. Two peers collaborate long term to send payments in between each other, and to route payments through the lightning network. The life cycle of a channel consists of the on-chain opening transaction, off-chain commitment transaction, different cases for collaborative or forced closing transactions, and the defense against theft with revocation transactions. Each of them will be analyzed in this chapter.

8.1.1 Funding Transaction

Two parties create a transaction funded by individual inputs, for example Alice provides a **`10 bitcoin`** UTXO as input. This funding transaction creates a **`2-of-2`** multi signature with the redeem script **`2 <PubKeyAlice> <PubKeyBob> 2 CHECKMULTISIG`**.[62] Alice and Bob can only spend this UTXO with both signatures. If one

62 BOLT 3, Funding Transaction Output.

of them is malicious, the funds are locked and irredeemable. Alice wants to protect herself against the case that malicious Bob goes off-line, so she requests Bob's signature over a commitment transaction, as described below, that send all **10 bitcoin** to a new script of Alice. Alice stores this transaction, yet she doesn't yet broadcast it. Now Alice will sign the funding transaction, knowing that at any time she could broadcast the initial commitment transaction with Bob's signature. The funding transaction is verified by every full node and confirmed in the time chain. Now the payment channel is open, it has a unique identifier of the transaction and channel ID. Alice and Bob can choose to announce this channel publicly to the lightning network and offer to route payments up to the capacity of the multisig.

8.1.2. Commitment Transaction

Subsequently, Alice and Bob can exchange signed commitment transactions[63] which change the value of the outputs dedicated to Alice and Bob. This transaction consumes the output of the funding transaction, and creates four new outputs, one back to Alice's single signature private key, the other back to Bob's, and one for each offered and received HTLC.[64] Initially, only Alice partially signs[65] the transaction and sends it to Bob, who completes the **2-of-2** multi signature and sends the fully signed transaction back to Alice. The next commitment transaction consumes the same founding transaction output, but changes the amount dedicated to the newly created outputs of Alice and Bob. These valid transactions could be broadcasted to the network and added to the time chain, but for now they are kept occulted by Alice and Bob.

Offered HTLC output P2WSH:

```
DUP HASH160 <RIPEMD160(SHA256(revocationpubkey))> EQUAL
 IF CHECKSIG
 ELSE <remote_htlcpubkey> SWAO SIZE 32 EQUAL
   NOTIF DROP 2 SWAP <local_htlcpubkey> 2 CHECKMULTISIG
   ELSE HASH160 <RIPEMD160(payment_hash)> EQUALVERIFY CHECKSIG
   ENDIF
 ENDIF
```

Witness Script: **<remotehtlcsig> <payment_preimage>**

63 BOLT 3, Commitment Transaction.

64 BOLT 3, Commitment Transaction Outputs.

65 Chow. BIP 174: Partially Signed Bitcoin Transactions. 2017.

If a revoked commitment transaction is published, the Witness Script **<revocation_sig> <payment_preimage>** can spend the output immediately. For every commitment transaction, the receiver requests the revocation private key before accepting the money. Thus for any received payment there is proof of payment, and that can be used to punish a malicious actor trying to settle an old state. It has to be ensured, that a old state of the channel is invalidated with the most current commitment transaction. There needs to exist enforceable proof in the case that a old state is closed on chain. There are several different update and revocation mechanisms with according thread models and security assumptions:

Transaction held by Alice:

```
[i] 2-of-2 funding output, signed by Bob
[o] <nValueAlice>: <PubKeyAlice>
[o] <nValueBob>: IF <Revocation Public Key> ELSE
    <delay in blockst> CHECKSEQUENCEVERIFY DROP <PubKeyBob>
    CHECKSIG
```

Transaction held by Bob:

```
[i] 2-of-2 funding output, signed by Alice
[o] <nValueBob>: <PubKeyBob>
[o] <nValueAlice>: IF <Revocation Public Key> ELSE
    <delay in blocks> CHECKSEQUENCEVERIFY DROP <PubKeyAlice>
    ENDIF CHECKSIG
```

The revocable key is split in two secrets, similar to **2-of-2** multi signature based on elliptical curve arithmetics. When Bob sends funds to Alice, Bob has to revoke the old commitment transaction by revealing her secret to Alice, before Alice agrees to sign the commitment transaction of the new state. If Bob would try to cheat and broadcast an old state, Alice can become active and use both paths of the revocation key to redeem Bob's delayed output. Alice only has one half of the revocation key and can only redeem her output after the block delay timeout. With each state update, both exchange the new commitment transactions, and the revocation secret of the previous one.

8.1.3 Closing Transaction

After Alice and Bob have done several off-chain payments, they can cooperatively close this payment channel, by broadcasting the final multisig settlement transaction to the network. This cooperative closing transaction has a witness script **`0 <signature1> <signature2> 2 <PubKey1> <PubKey2> 2 CHECKMULTISIG`** and can specify any new ScriptPubKey in the output. In this case where both signatures are available, the transaction does not include any timelocks and thus can be spend again without any timeouts.

In the case where one peer is uncooperative the other party can do a one-sided closing transaction. This is the script with the revocation key and HTLC in order to protect against the closing with an old state. In this time-out window the uncooperative party has the opportunity to come back online and to check if this closing attempt is actually valid. If not, then the revocation key is used as proof that this is an old state.

8.2. n-of-n Multi Party Channel Factories

Channel factories are payment channels where every commitment transaction opens more payment channels. The off-chain update transaction closes the previous payment channel and opens the new one atomically. This script enables the secure opening and closing of a new payment channel without committing any extra transaction to the time chain. A 10 peer channel factory has 90% transaction size savings compared to individual channel opening.[66]

8.2.1. Hook Transaction

Two, or preferably more peers create a channel factory deposit transaction that is verified by all nodes and committed to the time chain. All peers provide their individual UTXOs with witness proofs in the input of the hook transaction, and they create several individual change outputs, as well as the channel factory script UTXO. This is the transaction to collect all the bitcoin from the peers and fund the channel factory. This script has the regular payment channel conditions, the **n-of-n** cooperative case, as well as partially signed yet unbroadcasted backup transactions with time locks with single signatures for uncooperative spending. The hook transaction is signed only when all transactions of the initial state are signed to ensure the funds always return to their initial owner in case of the uncooperative case.

```
[i]     Alice     | [o]  10-of-10 channel factory
        Bob       |      Alice Change
        ...       |      Bob Change
        Justin    |      ...
                  |      Justin Change
```

66 Harding, What are Channel Factories and how do they work? 2018.

8.2.2. Replaceable Allocation Transaction

After the funding transaction has sufficient accumulated proof of work, the peers can collaboratively update the channel factory by creating an unbroadcasted commitment transaction. The input is the **`n-of-n`** cooperative multi signature of the allocation transaction, the outputs are the funding UTXO with a **`2-of-2`** direct payment channels between the individuals within the factory.

Every allocation transaction thus spends the hook UTXO to create individual payment channel funding transactions. Only peers of the same factory can connect, since only they verify and enforce the scarcity and double spending protection of these bitcoin. Because only the factory peers need to verify the unbroadcasted commitment transactions, the speed of opening and closing an individual payment channel is near instant, and without any on-chain transaction cost.

The goal of a channel factory is to have many off-chain allocation transactions that open many individual **`2-of-2`** payment channels. Because only the latest state of these allocation transactions must successfully be committed on the time chain, it is thus essential that old states are replaced by the new one. This can be done by building an invalidation tree with either decrementing timelocks started by a kickoff root transaction[67], exchanging revocation secrets[68] or utilizing the eltoo[69] updating mechanism. The leaves of the invalidation tree are the individual **`2-of-2`** multi signatures that open an individual 2 peer payment channel.

67 Decker, Wattenhofer. A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. 2016.
68 Poon, Dryja. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. 2016.
69 Decker, Russell, Osuntokun. eltoo: A Simple Layer 2 Protocol for Bitcoin. 2018.

8.2.3. Commitment Transaction

After the hook transaction has sufficient accumulated proof of work, and the allocation transaction is successfully signed and communicated, then the individual sub-channels are open. The commitment transaction spends the **`2-of-2`** multi signature of the allocation transaction, and creates UTXOs with the single public key of the individual owners of the bitcoin. The scripts and signing ceremony is identical to the ones of regular **`2-of-2`** payment channels.

8.2.4. Closing Transaction

In the case where all peers collaborate, they will close all the individual payment channels within the factory, this is done all off-chain and should be coordinated within seconds. Then they build and sign a factory closing transaction with the **`n-of-n`** multisig as the input, and individual UTXOs with the correct number of bitcoin according to the latest state of the factory. In this case there are no timelocks, and thus every individual can spend their own UTXO as soon as they desire. The individual script might even be the funding of a new channel or factory according to the splicing in and out process.

In the uncooperative case, the on-line peers can construct a one sided closing transaction, which is however limited by an additional time lock. During this period the off-line peers have the opportunity to come back on-line to check if this is the most recent state of the channel factory, and if not, to proof it and punish the thieves.

8.3. m-of-n HTLC

An early proposal for instant escrows was to use complex scripts with several HTLCs to enable threshold transactions inside a payment channel.[70] **m-of-n** hash preimages are required for the HTLC to be fulfilled. Timeouts are at the minimum the escrow timeouts and they increase with every additional **n** in the multisig scheme. Sender, escrow and receiver [or others] have the **n** preimages, yet only **m** of them are required in order to validate the redeem script. In most cases, sender and receiver will disclose preimage themselves without the need for escrow action. Although this does work theoretically, it commits a lot of information to the time chain and is thus rather inefficient.

Assume the order in the stack is Sender, Escrow, Recipient.

```
For PAID 2-of-3 Escrow+Recipient, the HTLC stack is:
        <BobSig> <0> <EscrowPreimageR> <RecipientPreimageR> <0>

If it's REFUND because 2-of-3 has not been redeemed in time:
        <AliceSig> <0> <1>

Bitcoin Script (Alice's, we use OP_1/OP_0 to distinctly show
computed
true/false. 0/1 is for supplied data as part of the
sigScript/redeemScript stack):


//Paid
IF
        <CSVDelay> DROP CSV //under rusty's CSV style

   //Stack: <BobSig> <0> <EscrowPreimageR> <RecipientPreimageR>
        //Recipient must agree to receive funds.
        HASH160 <RecipientHash> EQUALVERIFY

        //Stack: <BobSig> <0> <EscrowPreimageR>
        //Either the Sender or Escrow must consent for payment
        HASH160 <EscrowHash> EQUAL
        //Stack: <BobSig> <0> <OP_1>
        SWAP
        //Stack: <BobSig> <OP_1> <0>
        HASH160 <SenderHash> EQUAL
        //Stack: <BobSig> <OP_1> <OP_0>
```

70 Poon. 2-of-3 Instant Escrow, or How to Do "2-of-3 Multisig Contract" Equivalent on Lightning, Lightning-Dev-Mailinglist. 2016.

```
        BOOLOR
        //Stack: <BobSig> <OP_1>
        VERIFY

        <BobPubKey>
        //Stack: <BobSig> <BobPubKey>
//Refund
ELSE
        //Stack: <AliceSig> <0>
        HASH160 DUP
        <R-HASH> EQUAL
        NOTIF
                <CSVDelay> DROP CSV
        ENDIF

        <HTLCTimeout> DROP CLTV

        //Stack: <AliceSig>
        <AlicePubKey>
        //Stack: <AliceSig> <AlicePubKey>
ENDIF
CHECKSIG
```

PART III: Use Case

As described in the first part of this thesis, non-simulated shared ownership of any scarce asset, but especially of a libre sound money is a groundbreaking achievement. With this new technology at our disposal, there are countless new solutions to known problems, and probably even more solutions to unknown unknowns. The following list is mere an early compilation of interesting use cases, and not at all exhaustive. It is up to individual entrepreneurs to evaluate client problems, judge possible solution and build the services desired. These use cases might serve as inspiration for peers to evaluate the means available and which possible ends they could serve. As non-scarce knowledge they will be copied, adapted for unique needs and manifested through human action to remove uneasiness.

1. 1-of-2 Joint Spending

Alice and Bob have a joint account used for every day spending of petty cash. The signature of either one of them is sufficient to send the coin, thus no collaboration or communication between the two is necessary. Any one of them can at any time, for any reason, spend all the bitcoin locked in this script. This is very convenient for every day transactions, but it requires a large amount of trust, since both of them could turn rogue and run away with the money, and there is no defense against this. Further, if only one of them gets compromised by a malicious hacker, he can equally steal the funds without recourse.

2. 2-of-2 Joint Saving

Alice and Bob have a joint account used for long term savings. Both signatures are required to spend the funds, this requires coordination and communication for the signing ceremony. If either one of them is unavailable, the funds cannot be spend. This is thus true shared ownership of the bitcoin, since two separate individuals need to give their explicit consent to transact. This approval process might not be convenient for recurring payments, and thus it is more intended for long hodling of the coins. Since even the loss of one private key leads to indispensability, it is very much recommended to keep several backups in diverse secure locations. In the case of a malicious hacker gaining access to one of these keys, the funds are still secure, and there might be enough time to move them to a new script with fresh keys.

3. 2-of-2 Two-Factor Authentication

In order to protect against the leaking of one non-scarce private key and to increase the redundancy and security, Alice can require a second factor signature from another private key in a `2-of-2` multisig. Alice alone generates two private keys, she generates and stores one on her phone and the other on her hardware wallet. Now she requires access to both devices in order to spend the funds, thus an attacker needs to compromise both at the same time. In the case that a malicious actor gains undue access to only one of her devices containing the private key, this is not enough to spend the coins. The chance that a hacker is breaking both of her devices is several orders of magnitude more difficult. This also highly increases the defense against a malicious hardware provider, since two independent manufacturers need to collude. However, in the case that Alice looses only one of her devices and private key, she looses access to the bitcoin which would rightfully be hers. It is as impossible for her to spend the UTXO with only one key, as it is for a malicious actor. There is no redundancy to protect against key loss.

Thus, **n-of-n** second factor authentication is a valid defense against the leaking of private keys to unwanted malicious actors. They need to gain access to **n** private keys in order to have full control over the UTXO, the difficulty exponentially increases as **n** increases. Especially when Alice stores the private keys in different devices, in different location and with different protocols. However, the **n-of-n** script does not protect in the case where Alice herself looses access to even only one of the private keys. She can lock herself out of her own money, and this risk increases, as n increases. Although, this is also the case for a single signature script, once that one key is lost, the money is locked indefinitely. The trade off for the **n-of-n** scheme is thus the number of **n** in relation to the attackers sophistication to break all **n** security protocols, and the likelihood of Alice herself to loose only one of **n** keys.

4. 2-of-3 Child Saving

Alice and Bob are the parents of Charlie, and they have a child savings account where all three have one key. The parents would like to pass on with "warm hands" some of their wealth to their heir and to teach him the importance of decreased time preference through savings. Charlie alone cannot spend the funds, but he requires the approval of either one of his parents. In the same manner, one of the malicious parents or a hacker cannot alone spend the funds, but they always require approval of either their son or partner. The parents are protected against unwise spending of their son, and Charlie is protected against attempted theft of one of his parents. In the case of loss of one private key, the other two can still collaborate to produce a valid signature script and spend the coin into a fresh multisig with a new key setup.

5. 2-of-3 Buyer, Seller, Trust less Escrow

Alice wants to purchase a good from Bob, but they wanted to protect against the first-mover disadvantage. Charlie can be as the trust less escrow of a **`2-of-3`** multisig. Every one of them generates a unique private public key pair, one signature alone cannot validate the script, it always requires the collaboration of two signers. Alice sends the amount of bitcoin agreed upon previously to an UTXO with this redeem script. Bob will wait until these funds are confirmed and verify on his full node. He now has absolute proof that Alice has the bitcoin, and is willing to spend them. Even further, he now has shared ownership in this coin, requiring only the collaboration of one of the peers to gain the full control over it. Bob will now give Alice the good she desires and he will build and sign a transaction spending the multisig coin and generates his own singlesig UTXO. Only when Alice has a secured control over the good and verified its quality will she also sign that proposed transaction and broadcast it to the Bitcoin network. In this collaborative case, Charlie as trust-reduced escrow is not needed, he might not even know that he was part of the scheme. But when malicious Bob does not hand over the good, Alice will call on the judge and request his analysis of the case. Charlie will verification that Bob is indeed malicious, and then co-sign with Alice a transaction giving the bitcoin back to her. On the other hand, if Bob has given malicious Alice the good, but she refuses to pay, Bob can appeal to Charlie who will judge in his favor and co-sign a transaction paying Bob. In order to protect against denial-of-service attacks an upfront escrow from Alice and Bob might be added to the multisig address. In the case of attempted fraud, the judge will release that deposit to the victim. To ensure Charlie's honesty Alice and Bob might also require a security bond from him. Charlie himself alone cannot steal the funds. The decentralized and self-hosted bisq exchange[71] has been successfully using this smart contract to secure millions of trades.

71 Beams, Karrer. Phase Zero Protocol. 2017.

It would also be possible to include a time out condition that the bitcoin can be spend by the original owner, after a certain time window has passed. This can further reduce the need for collaboration with the third party arbitrator, because in the malicious off-line case, the victim can simply wait for the time out to pass, and then spend the funds without anyone else's interaction.

Redeem Script for time locked escrow:

```
IF
  IF
    2
  ELSE
    <1000 blocks> CHECKSEQUENCEVERIFY DROP
    <Public Key 1> CHECKSIGVERIFY
    1
  ENDIF
  <Public Key 2> <Public Key 3> <Public Key 4> 3 CHECKMULTISIG
ELSE
  <3000 blocks> CHECKSEQUENCEVERIFY DROP
  <Public Key 1> CHECKSIGVERIFY
ENDIF
```

6. 2-of-3 Hot Wallet Security

The main risk of **n-of-n** multisig is that the loss of only one key leads to a complete loss of funds. Although this is very secure against malicious actors trying to steal money, it does not protect Alice from herself loosing access to one single key. A **m-of-n** script can provide further benefits that neither single, nor **n-of-n** multisig have.

In order to secure hot wallet funds Alice can generate two private keys, generating and storing one on her hot hardware and the other one as a cold storage. The security specialist Bob will have the third key on a hot wallet. Alice can assign a **2-of-3** multisig script to a UTXO, such that she can only spend the coin with a total of **2** signatures. Every time Alice wants to make a payment she builds the transaction with the multisig as input and signs it with her hot key. Then she sends the partially signed Bitcoin transaction to Bob who will only sign the transaction if some predetermined conditions are met. This can be some two-factor authentication, or in-person verification, or a white listed and blacklisted addresses, or some maximum value in a given time period. Only when all of the agreements are met will Bob sign this transaction with his hot wallet key. If Bob realizes that Alice has been compromised then he will refuse to sign the transaction. It is important to note that the Bob's spending conditions are only simulated and rely on the trust in an honest Bob.

If Bob becomes unavailable, or fails to uphold his promise to co-sign, then Alice can get her second key out of cold storage and sign the transaction all by herself. She does not need to cooperate with Bob in order to spend her money, without any other party, she has full control over that coin. Bob alone cannot spend the money, he never has full control over the coin, and thus no property rights in it. A **2-of-3** multisig has the same anti theft protection as a **2-of-2** multisig. The attacker needs to gain access to both Alice's hot and cold wallet, or one of them in addition to Bob's hot storage. However, in the case that Alice looses one of her every day keys, let's assume her hot wallet, she can get her second key out of cold storage, and use it together with Bob's hot key. She can decrease the risk of loss of private keys drastically with such a script.

The **m-of-n** multisig script provides simultaneous protection against theft and loss of private keys. The malicious actor needs to gain access to **m**, not **n**, private keys in order to generate a valid spending transaction. This provides the same level of protection as a **m-of-m** multisig script would. However, contrarily to **n-of-n** scripts, Alice can afford to loose **n-m** private keys before she herself looses access to the UTXO. She can have **m** convenient and easy to use keys, like mobile and hardware wallets, which she can interact with for every day spending. For redundancy however, she also has **n-m** cold storage keys, which are difficult to access, for example a paper wallet hidden in a remote safe. She only needs to reveal these keys in the case where she looses access to some of the every day keys. However, the UTXO is locked when **m** private keys are lost, same as with a **m-of-m** multisig script.

7. 3-of-5 Low Trust Joint Funds

The five peers Alice Bob Charlie David and Eve work on the same project and have a low trust **3-of-5** multisig. Each of them holds one key and they need collaboration of any of the three peers in order to spend the bitcoin. This reduces the threat of embezzlement, hacking and loss of two keys. Since it is provable on the time chain which private keys have signed the transaction, there is accountability after the fact.[72] This is especially useful for decentralized and non-hierarchical projects where peers have the convenience of collaboration without full consensus for every transaction. This setup retains spend-ability for up to the loss of two keys, yet it is also secure for up to two malicious peers.

72 It is evident for script based multisig, yet not by default for MuSig, but there is a drop-in protocol to achieve the accountability.

8. n-of-n Transaction Output Commitments

Alice wants to pay 10 different peers, yet the current transaction fee level is high, and she estimates it will decrease in the future. She wants to commit to the payment now, yet pay the fee for the final transaction at a later point in time. Alice requests the public keys of all 10 receivers and builds a large **`10-of-10`** multi signature script. She builds an unsigned and unbroadcasted setup transaction, with her own UTXO in the input, and the **`10-of-10`** multisig and her change output. The value of the multisig coin is the total sum of all the 10 payments she is sending. Then, she creates a distribution transaction, which spends the multisig UTXO and creates the 10 different UTXOs with the individual public keys of the receivers. Alice then requests each of the receivers to sign the distribution transaction, and she ensures that each co-signer has a valid signature of all the peers. Only then does she sign and broadcast the setup transaction, to get confirmation with a relatively low fee. The receivers are now certain that at any time one of them can broadcast the signed distribution transaction to receive the money. Yet they can delay the distribution transaction until the fee for block space has decreased.

Setup transaction [signed after distribution tx]:

```
[i] Alice signature  |   [o] 10-of-10 multisig
                     |      Alice shange
```

Distribution transaction [signed first, broadcasted later]:

```
[i] 10 signatures    |   [o] Bob pubkey
                     |      Charlie pubkey
                     |      ...
                     |      Kai pubkey
```

Conclusion

The first part of this thesis explores with a rational praxeological analysis the axiom of scarcity and non-scarcity as the fundamental per-requisite of the realm of human action. Individuals live in a world filled with scarce and non-scarce goods and both can be used to satisfy desires. Yet human action only applies to the allocation of scarce resources throughout time. Scarce goods are exclusive, meaning that only one individual can at any time fully use the good, as well as rare, meaning that there is a limited supply of the means to satisfy only a limited number of ends. Without this universal limitation there would be no need and justification for individuals manifesting themselves in this world.

Scarcity is a solid foundation for the praxeological analysis to explore the two different forms of interaction, the voluntary mutually beneficial economic means, and the aggressive exploitative political means. On the one side is Liberty, where free individuals have the property rights in themselves, as well as their homesteaded and exchanged goods. The agora is based on the mutually beneficial trade of scarce goods so that both parties increase their subjective marginal preferences. The polar opposite is slavery, where a master initiates aggressive force against the private property of a slave. Since this is not a voluntary action, it is praxeological proof that only the tyrant is increasing his value scale, while the victim is suffering a reduction in his well-being.

The other side of this axiom is non-scarcity of goods. These are either non-exclusive, so that the use by one individual does not prohibit the use of another, or abundant, in the sense that there is enough for everyone to satisfy their desire. Non-scarcity is evident in the realm of mind and cyberspace, where knowledge can be shared freely without the need of sacrifice of the inventor. Information can be perfectly copied and instantly shared with whomever it might be valuable for only a minuscule amount of computational energy. This is especially the case for libre open source software, since the user has full access to run the code, he can copy and distribute it, as well as study, change and improve the non-scarce software. This is in stark contrast with the fallacious claims of intellectual property rights, where the creator utilizes state aggression in order to create artificial

scarcity for the knowledge he has created. Any form of this ownership claim in knowledge is not just logically infeasible, but especially destructive and an unjustifiable limitation to humankind.

Cryptography is a brilliant use of non-scarce information in order to create exclusive knowledge of a secret, with mathematical proof of existence without revealing the secret itself. Since a private key is simple a non-scarce large random number, there is no need for property rights to allocate it. A proper use of private public key cryptography requires that the knowledge of the secret is kept occulted by the individual who is using it. However, this is an artificial exclusivity and is easily broken by simply copying the private key, without taking it from the original creator. Bitcoin uses a vast array of different non-scarce protocols and software in order to emerge cryptographically proven scarcity of bitcoin. One exclusive UTXO is at any time always locked up by only one script, and only he who can produce a valid witness has the ability to spend the coin. The halvening limits the supply of new bitcoin being created, thus establishing the rarity of the libre sound money. Full nodes define, verify and enforce the Nakamoto consensus protocol, and thus full nodes create and secure the scarcity of UTXOs.

Part two details and summarizes the different cryptographically protocols and scripts that define and enforce the property rights of bitcoin. The Bitcoin scripting language is used to commit an exclusive locking script at the time of creating the coin. Only with a valid solution to this script, and nothing else, can the bitcoin be send to another script. This can be as basic as a single public key and signature, here the coin is in the exclusive ownership of the first individual to prove knowledge of the private key. Yet there are further capabilities of the scripting language to describe more precise property right definitions.

The command OP_CHECKMULTISIG specifies $\mathbf{n}$ public keys in the redeem script, and requires $\mathbf{m}$ signatures of these keys in the witness script. One individual alone can not spend the coin, but he always requires the collaboration of $\mathbf{m}$ signers. Thus one alone has no exclusive ownership of the coin, rather a shared ownership amongst any group of $\mathbf{m}$ individuals. This spending condition is verified and enforced explicitly by full nodes.

Another technology to achieve a similar end is the Schnorr signature scheme, which due to its linearity enables many possibilities to define shared ownership. **N** individual public keys are aggregated into one single public key which is committed to the time chain in the redeem script of a single signature UTXO. A valid signature can only be produced when **n** individual private keys sign the same message interactively in order to produce one single aggregated signature. Taproot is a new script verification protocol that enables the setup of several redeem scripts in a tree structure for a reduction of data size committed to the chain and increased privacy. One basic use of taproot is a brilliant setup of a **m-of-n** threshold scheme that is actually a tree with several individual **m-of-m** scripts, so that any **m-of-n** can produce a valid witness.

Shamir's Secret Sharing is a protocol used to divide a given master secret **MS** into **n** shares, where knowledge of **m** shares is required in order to calculate the master secret again. Although the **n** peers have each exclusive knowledge of their **n** shares, they alone have no knowledge whatsoever about the **MS**. Yet the master dealer has knowledge of **MS**, thus this scheme is not a true non-simulated shared ownership setup. Yet it can be used in conjunction with other protocols like Schnorr to enable non-simulated **m-of-n** threshold ownership of a UTXO.

The lightning network consists of several individual payment channels to update the current state of property rights of a UTXO off-chain without verification by all full nodes. The current implementation of these payment channels is a **2-of-2** multisig between the channel peers, with several partially signed commitment transactions that spend this multisig into the individual scripts. This is a complex setup of several transactions with signatures, hashed time locked contracts and revocation secrets. Yet this concept of a **2-of-2** payment channel can be expanded to create a **n-of-n** channel factory that creates many individual **2-of-2** payment channels off-chain. Lightning network is one of the well thought out implementations of non-simulated shared ownership of scarce bitcoin, yet there are countless more use cases.

Multi signatures can be used to solve for countless problem in the context of security, collaboration and exchange. In the case that one individual has control of all **n** keys, then this is a second-factor defense against the leaking of any **m-1** keys. This can be further straightened by giving **m-1** keys to a semi-trusted security specialist to verify specified spending conditions before co-signing a transaction. It can also be used to manage shared funds in a collaborative project, so that only a subset of participants needs to agree to a spend, and one individual alone can not go rogue and steal the funds. One of the most powerful use cases is the trustless execution of a **2-of-3** escrow for the exchange of other goods. Here the merchant gains a proof of fund before the good is exchanged, and the buyer gains the ability to verify the goods before ultimately settling the payment.

To conclude, bitcoin is libre sound money, defined, verified and enforced by individual sovereign full nodes. With a brilliant application of non-scarce cryptography and open source software, Bitcoin emerges cryptographically proven non-simulated shared ownership of a scarce UTXO. This technology can be used by individuals to support their desire to remove uneasiness by allocation scarce resources throughout time.

Future Research

The bulk of future research is in context of specifying the proper implementation details of the several multi signature schemes described in this thesis. Several peers are working on the protocol implementation[73] of Schnorr signatures, which will kick off a new era of shared ownership of bitcoin. Then independent wallet and back-end developers need to understand and implement this new signature scheme.[74] Hardware wallets need to be carefully checked for Schnorr MuSig capability, since the interactive signing ceremonies are more complex than independent singlesig operations.[75]

Lightning network uses multisig as a fundamental building block, and Schnorr threshold signatures can solve many design issues of the current implementation.[76] This includes upon others the construction of more efficient and private individual payment channels and large scale channel factories, yet also the use of adaptor signatures for more secure and private routing of payments[77]. Not covered in this thesis are possible approaches to enforce a multisig spending condition for one specific lightning payment, not for the entire channel.[78] Although there are several possible approaches, they need a lot further research, clarification and of course implementation.

This thesis also omits the use of multisig in federated side chains[79], which in the current script based multisig setup is rather cumbersome and inefficient. Alone Schnorr aggregated MuSig and Taproot will make this advanced script exponentially more efficient. Yet there is active research in different mechanisms for securing a side chain in general, specifically state chains shows great promise.[80]

---

73 See Wuille and others in pull request #212 of the bitcoin-core/secp256k1 repository.

74 For example Wasabi Wallet is already utilizing Schnorr blind signatures, and as soon as possible, SegWit v1 Schnorr.

75 Active research by Stepan Snigirev, Pavol Rusnak, Marek Palatinus, Peter Gray and others.

76 Active research by KZen

77 Active research by Jonas Nick and Andrew Poelstra

78 Although heavily discussed with Jonas Nick, currently not advanced enough for explanation.

79 See Back, Corallo, Dashjr, Friedenbach, Maxwell, Miller, Poelstra, Timón, Wuille. Enabling Blockchain Innovations with Pegged Sidechains. 2014.

80 See Somsen, Statechains: Non-custodial off-chain Bitcoin Transfer, 2018.

References

Andreson, Gavin	BIP 16: Pay to Script Hash, 2012
Beams, Chris; and Karrer, Manfred	Bisq: Phase Zero Protocol, 2017.
Belcher, Chris	Bitcoin Privacy, Bitcoin Wiki, most recent version 1st July 2019
Bernstein, Daniel J.;Duif, Niels; Lange, Tanja;Schwabe, Peter; and Yang, Bo-Yin.	High-Speed High-Security Signatures. In Bart Preneel and Tsuyoshi Takagi, editors, Cryptographic Hardware and Embedded Systems – CHES 2011, volume 6917 of LNCS, pages 124–142. Springer, 2011.
Bouckaert, Boudewijn	What is Property? In Symposium: Intellectual Property, Harvard Journal of Law & Public Policy 13, no. 3, 1990
Chow, Andrew	BIP 174: Partially Signed Bitcoin Transactions. 2017.
Decker, Christian; and Wattenhofer, Roger	A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. 2016.
Decker, Christian; Russell, Rusty; and Osuntokun, Olaoluwa	eltoo: A Simple Layer 2 Protocol for Bitcoin. 2018.
Fetter, Frank Albert	Economic Principles, 1915
Frey, Shaolin	BIP 148: Mandatory Activation of SegWit Deployment, 2017

Gnu Project	What is Free Software, most recent version 1st July 2019
Harding, David	Single-sig spending using Taproot. Bitcoin Optech Newsletter #46, 2019.
Harding, David	What are Channel Factories and how do they work? Bitcoin Stack Exchange, January 2nd 2018.
Hazewinkel, Michiel	Lagrange interpolation formula. Encyclopedia of Mathematics, Springer Science+Business Media B.V. 1994
Hillebrand, Max	Anarchy in Money, on the Ethical Economics of Bitcoin, 2018
Hoenicke, Jochen; Kozlik, Andrew; Palatinus, Marek; Rusnak, Pavol; Susanka, Tomas; and Vejpustek, Ondrej	SLIP 0039: Shamir's Secret-Sharing for Mnemonic Codes. 2017.
Hoppe, Hans Hermann	Theory of Socialism and Capitalism, 1989
Hoppe, Hans Hermann	The Economics and Ethics of Private Property, 1993
Kinsella, Stephan	Against Intellectual Property, 2008
Lamport, Leslie; Shostak, Robert; and Pease, Marshall	The Byzantine Generals Problem. In ACM Transactions on Programming Languages and Systems, July 1982. pp.382-401.
Locke, John	The Two Treatises of Government, 1689
Lombrozo, Eric; Lau, Johnson; and Wuille, Pieter	BIP 141: Segregated Witness. 2017
Marx, Karl	Critique of the Gotha Program

Maxwell, Gregory	Taproot: Privacy preserving switchable scripting. Bitcoin-dev mailing list, Jan 23 2018.
Maxwell, Gregory	Graftroot: Private and efficient surrogate scripts under the taproot assumption. Bitcoin-dev mailing list, Feb 05 2018.
Maxwell, Gregory; Poelstra, Andrew; Seurin, George;and Wuille, Pietre	Simple Schnorr Multi-Signatures with Applications to Bitcoin, 2018
Mises, Ludwig von	Human Action,1949
Mises, Ludwig von	Socialism: An Economic and Sociological Analysis, 1951
Nick, Jonas; Petukhow, Dimitri; andWuille, Peter	BIP-Taproot: SegWit version 1 output spending rules, 2019.
Pedersen, Torben Pryds	Non-interactive and information-theoretic secure verifiable secret sharing. Lecture Notes in Computer Science (Crypto '90), 473:331-238, 1991.
Poelstra, Andrew	Scriptless scripting and deniable swaps. Mimblewimble team mailing list, Feb 03 2017.
Poon Joseph	2-of-3 Instant Escrow, or How to Do "2-of-3 Multisig Contract" Equivalent on Lightning, Lightning-Dev-Mailinglist. 2016.
Poon, Joseph; and Dryja, Thaddeus	The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. 2016.

Ristenpart, Thomas; and Yilek, Scott	The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks. In Moni Naor, editor, Advances in Cryptology - EUROCRYPT 2007, volume 4515 of LNCS, pages 228–245. Springer, 2007.
Rothbard, Murray Newton	Man Economy and State with Power and Markets, 1962
Rubin, Jeremi; Naik, Manali; and Subramanian, Nitya	Merkelized Abstract Syntax Trees, 2016.
Schnorr, Claus-Peter	Efficient Signature Generation by Smart Cards, J. Cryptology, 4(3):161–174, 1991
Schwartz, Aaron	Guerilla Open Access Manifesto, 2008
Shamir, Adi	How to Share a Secret. Communications of the ACM, Volume 22, November 1979.
Stinson, Douglas R.; and Strobl, Reto	Provably Secure Distributed Schnorr Signatures and a (t,n) Threshold Scheme for Implicit Certificates. Certicom Corporation, 2001.
The Bitcoin Core Developers	Source code of Bitcoin Core
The Bitcoin Wiki Contributors	Weight Units, most recent version 1st July 2019
The Lightning Network Developers	Basics of Lightning Technology 3, most recent version 1st July 2019
Wills, Garry	St. Augustine, 1999
Wuille, Pieter	BIP Schnorr: Schnorr Signatures for secp256k1, 2019
Zimmermann, Phil	Pretty Good Privacy Freeware Software, 1991