National Technical University of Athens

School of Electrical and Computer Engineering

Department of Computer Science

Trust in decentralized anonymous marketplaces

Dionysis S. Zindros

November 2015

Advisor: Aris Pagourtzis, Assistant Professor

National Technical University of Athens

School of Electrical and Computer Engineering

Department of Computer Science

Trust in decentralized anonymous marketplaces

Dionysis S. Zindros

Dionysis S. Zindros,
Diploma in Electrical and Computer Engineering,
National Technical University of Athens.

Table of Contents

List of figures

List of tables

List of code listings

Abstract (Greek)

(An abstract in English follows)

Αυτή η εργασία εισάγει την έννοια της αποκεντρωμένης ανώνυμης αγοράς. Μελετάμε τη φύση τέτοιων αγορών και παρουσιάζουμε την υλοποίηση μίας συγκεκριμένης τέτοιας αγοράς την οποία ονομάζουμε OpenBazaar. Στη συνέχεια επικεντρώνουμε την προσοχή μας στη μελέτη των εννοιών της ασφάλειας και της εμπιστοσύνης μίας τέτοιας αγοράς.

Εξερευνούμε τις επιπτώσεις ασφάλειας που έχουν η αποκεντροποίηση και η ανωνυμία και προτείνουμε πολλούς μηχανισμούς που επιτυγχάνουν εμπιστοσύνη. Οι προτάσεις μας περιλαμβάνουν παραδοσιακές τεχνικές για ανάπτυξη ασφαλούς λογισμικού τις οποίες εφαρμόσαμε στην πράξη στην υλοποίησή μας. Προτείνουμε επίσης τη χρήση ενός δικτυακού επιπέδου που διατηρεί την ανωνυμία.

Πιο σημαντικά, εξερευνούμε πώς μηχανισμοί κρυπτονομισμάτων μπορούν να επιτύχουν εμπιστοσύνη χωρίς την ανάγκη κεντρικών έμπιστων αρχών, μέσα από τους συγκεκριμένους μηχανισμούς διαιτησίας, multisigs (πολλαπλών υπογραφών), και αμοιβαίως εγγυημένης καταστροφής. Οι κρυπτογραφικές και παιγνιοθεωρητικές ιδιότητες τέτοιων σχημάτων μελετώνται και μία υλοποίηση χρησιμοποιώντας το bitcoin παρουσιάζεται. Προτείνουμε επίσης νέες επιθέσεις σε τέτοια σχήματα.

Η συμβολή μας περιλαμβάνει πολλούς μηχανισμούς επιτυχίας εμπιστοσύνης. Εισάγουμε μία νέα μορφή web-of-trust στο οποίο πολλαπλασιαστικοί κανόνες χρησιμοποιούνται για την επίτευξη μεταβατικής εμπιστοσύνης, ενώ η αποκεντροποίηση, η ασφάλεια και η ανωνυμία διατηρούνται. Εξερευνούμε τις ιδιότητες ανωνυμίας και ασφάλειας τέτοιων webs-of-trust και συζητάμε επιθέσεις διαχωριστών. Η συμβολή μας επίσης περιλαμβάνει μηχανισμούς proof-of-burn για τη δημιουργία ταυτοτήτων για το οποίο παρέχουμε μία υλοποίηση. Τέλος, συμβάλλουμε επίσης με την ανώνυμη αποκεντρωμένη ονοματοδοσία μαγαζιών, για την οποία παρέχουμε μία Namecoin υλοποίηση.

Abstract

This work introduces a decentralized anonymous marketplace. We study the nature of such marketplaces, and we present our implementation of a practical such marketplace which we call OpenBazaar. We then focus our study on the security and trust aspect of such a marketplace.

We explore the security implications of decentralization and anonymity, and propose several mechanisms to achieve trust. Our proposals include traditional secure software development mechanisms, which we have applied in practice in our implementation. We also propose the use of an anonymity-preserving transport layer.

More importantly, we explore how cryptocurrency mechanisms can be used to achieve trust without the need of central third parties, through the particular mechanisms of arbitration, multisigs, and mutually assured destruction. The cryptographic and game theoretic properties of such schemes are explored, and an implementation using bitcoin is studied. We also propose novel attacks on these schemes.

Our contribution includes several mechanisms of achieving trust. We introduce a new form of web-of-trust in which multiplicative rules are used for trust transitivity, while decentralization, security, and anonymity are preserved. We explore the anonymity and security properties of such webs-of-trust and discuss separation attacks. Our contribution also includes the use of proof-of-burn mechanisms to build identities, for which we provide an implementation. Finally, we also contribute with anonymous decentralized naming of stores, for which we also provide an implementation using Namecoin.

Keywords

pseudonymous anonymous web-of-trust identity trust cryptocurrency bitcoin namecoin proof-of-burn timelock decentralized anonymous marketplace openbazaar

Acknowledgements

OpenBazaar is a collaborative work to which many researchers and volunteers around the world have contributed. The major portion of the invention and implementation was carried out by the other OpenBazaar co-inventors, Brian Hoffman, Sam Patterson, and Washington Sanchez. Washington in particular participated by helping with feedback and ideas around trust, Ricardian contracts, identities, and self-enforcing contracts; many of the ideas in this work are inspired by him or were invented by him where mentioned. Sam was a thought leader and helped us reinstate the cause we are building for and the political and social aspects of our work. Most importantly, Brian spent endless nights coding, recruiting, and leading the project.

But we couldn't get where we are without the continuous help of the OpenBazaar developers. Thank you all. Angel Leon, for building the important OpenBazaar networking infrastructure based on your FrostWire experience; and for being a peer for continuous discussions and ideas around decentralized systems after our meeting in Brussels. Nikolas Korasidis for all your hard work in code quality and refactoring; we are connected as fellow students in our school and through our work on OpenBazaar. Amir Taaki, for building the first quickly hacked but working version which inspired us all. Jonas David Nick who helped in the early stage of the OpenBazaar code and who was always willing to help me with my bitcoin questions in the Zürich Bitcoin Meetups at Kafi Schofell where we regularily paid in bitcoin. And Thomas Stilwell, John Adamopoulos, George Chatzisophroniou for their contributions.

This work would not be possible without the help of my dear friend and colleague Petros Angelatos who inspired me to create a decentralized marketplace. We first discussed about trust and identity systems together, long before OpenBazaar came to be.

I'd also like to thank my supervising professor Aris Pagourtzis who was helpful with his expertise in theoretical cryptography and algorithms. And my other teachers and mentors who taught me mathematics and programming. You know who you are, but I would like to mention three by name specifically: Nikos Papaspyrou, Andrew McCann, Dimitris Fotakis.

My students Chara Podimata and Kostis Lolos from the National Technical University of Athens helped with getting these ideas from theory to application.

My family, Spyros, Efi, and Natalia were there for me to help me during the writing of this work with their valuable support.

Thank you all.

Introduction

Marketplaces

A marketplace is a location where goods and services can be exchanged, swapped between each other directly, or for money. The participants in a marketplace are the buyer, who is interested in purchasing a product and the seller who is interested in offering a product to the market.

Marketplaces can exist offline or online. Examples of popular offline marketplaces are flea markets or shopping centers. Examples of popular online marketplaces are alibaba.com, a popular e-shop in Asia, amazon.com, an online bookstore, and ebay.com, an e-shop where buyers and sellers can transact between each other. Two other examples include Google's Play Store and Apple's AppStore, two online stores for mobile applications. We are concerned about online marketplaces.

Traditionally, marketplaces have exhibited various desirable qualities. We will explore the properties of decentralization, censorship-resistance, and anonymity.

Decentralization

Centralization and decentralization are critical characteristics of an online service. Traditional networking literature has concentrated on the networking portion of such systems. As decentralized systems are becoming prevalent and are having important political impact as they evolve, we feel it is time to redefine certain terms. We will make a distinction between the *network* structure of a system and the *technical ownership* structure of a system.

A *centralized network* service is a service which follows the client/server architecture. One server serves all clients, and the server is in this way a distinguished privileged node. This server could be owned and operated by a company, and in addition constitutes a single point of failure of the system; if the server is shut down, the network shuts down. Examples of centralized systems include traditional web sites.

A *hierarchical network* service is a service similar to centralized networks, except the centralization is hierarchical. Instead of a single server serving everyone, the workload is offloaded to sub-servers of any depth, which then serve clients. If any of them goes down, only a part of the network goes down, and thus the network is somewhat resilient. In the networking literature, these networks are described as "decentralized". However, in the context of *decentralized ownership* systems such as bitcoin and OpenBazaar, we will use the community definition of decentralization, which we define below, and we will reserve the term “hierarchical” to mean these systems. There are still single points of failure in these systems; if the root node of the hierarchical tree is brought down, the whole system can be shut down. Examples of such systems are web sites that operate load balancing systems to split up the workload. In these cases, the server nodes are still distinguished and different from the clients. Federated networks often have this form; for example, XMPP, diaspora, and IRC allow servers operated by different owners to communicate with each other. An important property of these protocols is that the servers trust the peers they are connected to.

In addition to network topology, we will now distinguish systems based on their ownership characteristics.

A network or service which is owned and operated by a well-defined party, which can be an individual, a company, or a group of companies or individuals working together is a system of *centralized ownership*. In a centralized market, the controlling owner is a third-party other than the buyers and sellers transacting within it. For example, in Google’s Play Store, the market is owned and operated by Google, even though buyers are end-users and sellers are developers of apps. These developers can be different from Google, but the market is still centralized. Centralized ownership systems can have any network form above; they can be centralized network, hierarchical network, or distributed network systems. For example, Skype is a centrally owned system which is structured in a distributed network. The fact that Microsoft can, in principle, control all the nodes in the system and, for example, block users by using a command signed with the operator’s private key if they are served a warrant, is what makes ownership centralized.

Decentralized systems are not owned by any party. For example, in offline markets, flea markets, or bazaars, often exhibit this property: Bazaars are locations where traders can simply meet up and sell or buy goods. They are not operated, owned, or controlled by any party.

A critical property of decentralization is the lack of centralized control. As there is no owner, there can be no decision making in who participates in the market. Therefore, the market is necessarily open to anyone willing to participate. This makes decentralized markets naturally resistant to censorship. As there is no owner, censorship, and in general any legal orders, have to be executed in a case-by-case basis by individual buyers and sellers. This is true for all decentralized systems. Examples of decentralized systems are bitcoin, OpenBazaar, Twister, Freenet, i2p, BitTorrent, PopCorn Time, Tor[1] and Gnutella. Decentralized ownership systems are necessarily structured in a distributed network manner. As they do not have a single point of failure and are not susceptible to control by an individual party, even if a legal order is served, we say that such systems enjoy *sovereignty*; they are the equivalent of nation-states, but in the Internet realm.

A contrast of centralized and decentralized systems is shown in Figure 1: Centralized and Decentralized . On the left-side, a centralized network centralized ownership system is shown. A designated node, highlighted in black, is superior to the other network nodes in that they have special ownership rights on the network. As the gatekeeper of the network, they must authorize all transactions between regular nodes which directly connect to them. The black node can be ordered to shut down, taking the system down with it. On the right, the gatekeeper node is gone, and the nodes connect directly to each other. In addition to the peer-to-peer network connetivity, the system has no central owner who can decide and censor participation in the system. It is a distributed network decentralized ownership system, simply referred to as *a decentralized system* in the cryptocurrency communities.

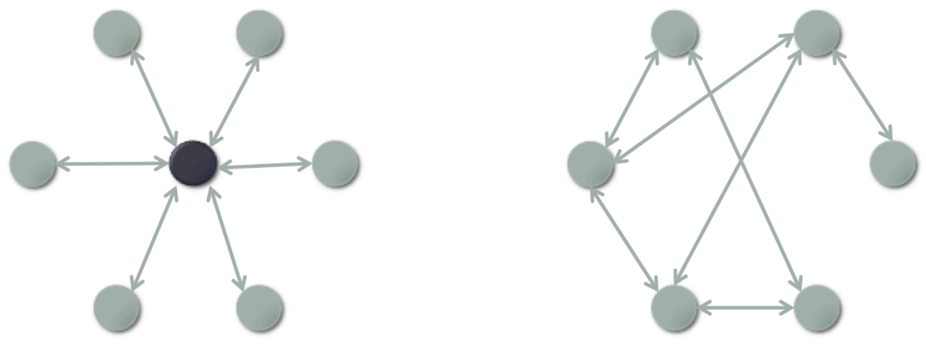

Figure 1: Centralized and Decentralized systems

This distinctive feature deserves an example. If Google's Play Store is found to contain illegal content, a subpoena or other form of court order can be used by a government to order the operator, Google, to remove certain content from their marketplace. In decentralized markets, as there is no owner, the court order has to be delivered directly to the merchant or buyer. Decentralization makes it technically impossible to impose central

[1] Tor, i2p, BitTorrent, and PopCorn Time are not always completely distributed due to indexing services such as torrent trackers or, worse, aggregators like ThePirateBay sometimes being centralized.

control. No matter what laws are put in place, the decentralized system will resist court orders. Decentralized systems are also resistant even to their own developers when it comes to censorship.

Conversely, if a censorship attempt is made, this means that a controlling party can get to decide which good can be sold and which cannot. Therefore, traditional censorship requires centralization.

Decentralization also implies that fees cannot be imposed on the system. If a fee structure is employed in the system, it cannot be enforced by anyone, as there is no owner. Even if the developers include code that allows them to collect fees, this code can be easily removed by participating parties.

Anonymity and pseudonymity

Anonymity in a marketplace ensures that the participants of the marketplace remain anonymous. We define anonymity similarly to how Tor defines anonymity: Anonymity is preserved when actions cannot be attributed to the physical person performing them. As the definition is broad and abstract, a specific threat model is required to indicate whether anonymity is preserved.

Pseudonymity is a concept similar to anonymity. In a pseudonymous setting, anonymity is maintained, but the anonymous actions are associated with a pseudonym, and various actions can be correlated: It is revealed that certain actions were taken by the same pseudonym, and this pseudonym can carry a reputation.

Pseudonyms are not necessarily associated one-to-one with a physical identity. A physical person may not have a pseudonym, or may have multiple ones. A pseudonym may be controlled by multiple physical entities, for example by a group of people. A pseudonym may also be abandoned by its physical owner to create a new one, which is not associated with the original, because the activity of the pseudonym remains anonymous (in addition to pseudonymous).

Pseudonyms may also be owned by software and do not necessarily have a human owner. In the particular case of decentralized systems, a pseudonym may be associated with a network where no central control of the pseudonym exists at all, which allows it to enjoy sovereign rights. An example of one system where such construction becomes possible is Ethereum.

A decentralized anonymous marketplace

We introduce a decentralized anonymous marketplace called OpenBazaar. This system is the collaborative work of the OpenBazaar team. Our contribution to OpenBazaar was to create the marketplace from scratch in collaboration with the rest of the OpenBazaar team. In the current work, we illustrate our contributions to the trust and security portions of OpenBazaar, which we consider a foundational portion of it.

Our contributions

At this point, we would like to state that some of the current work is descriptive, while some is normative.

The following properties are studied in this work, but are not our contributions. We simply describe them here, as we consider them an important portion of the trust and security mechanisms:

1) The 2-of-2 and 2-of-3 multisig use was invented in the bitcoin community and implemented for OpenBazaar by Brian Hoffman.
2) The MAD-style transaction was invented and used elsewhere. It was first proposed to be used in bitcoin in NASHX.
3) The ricardian contracts were invented in various settings and modified and extended for the OpenBazaar implementation by Washington Sanchez.

We have made several contributions in this work. Our contributions specifically include:

1) The suggestion to use 2-of-2 and 2-of-3 multisig for a decentralized marketplace was introduced in (Zindros, 2014). The game-theoretic study of the various transaction schemes, including the formulation for the 2-of-2 and 2-of-3 multisig is also our contribution.
2) The use of MAD using purely a bitcoin schema and its game-theoretic analysis in comparison to traditional multisig, including the ability for arbitration.
3) The transitive web-of-trust using multiplicative rules and maintaining anonymity.
4) The use of namecoin for naming stores, including an implementation.
5) The separator attack against a multiplicative web-of-trust.
6) The vendor-in-the-middle information-stealing and money-stealing attack.
7) The novel proof-of-burn using perturbations to build an identity, including an implementation.

History

Webs-of-trust

Webs-of-trust are traditionally a means to verify ownership of encryption and signing keys by a particular individual whose real-world identity is known. The first widely deployed web-of-trust is the GPG/PGP web-of-trust (Zimmerman, 1995). In these webs-of-trust, a digital signature on a public key is employed to indicate a binding between a digital cryptographic key and an identity. Such a digital signature does not designate trust, but only signifies that a particular real-world individual is the owner of a digital key.

Webs-of-trust have also been utilized for different purposes in various experimental settings. In Freenet (Clarke, Sandberg, Wiley, & Hong, 2001), it is used to guard against spam (Freenet Web Of Trust). In a commercial setting, the Bitcoin OTC web-of-trust successfully attempts a centralized approach to establish a true trust network (Grinberg, 2011), contrasting identity-only verification webs-of-trust of the past.

Economic origins of money and trade

In his nominal work preceding the invention of bitcoin, Szabo (Szabo, 2005) describes the origin of money and trade. The value of money is explored and understood as a means of exchange. As money is valued through its exchange value and not necessarily by some other backing, it is understood that bitcoin also gains its value from its ability to be used and exchanged. As such, we expect bitcoin's value to be closely associated with the exchange ability, usability, and security of online marketplaces such as OpenBazaar.

It is interesting to observe that traditional money shared many important properties with the modern form of bitcoin, especially as augmented by online marketplaces. A quote from the original work is revealing:

"To be useful as a general-purpose store of wealth and means of wealth transfer, a collectible had to be embedded in at least one institution with a closed-loop cycle, so that the cost of discovering and/or manufacturing the object was amortized over multiple transactions. It had to have certain functional properties, such as the security of being wearable on the person, compactness for hiding or burial, and unforgeable costliness. That costliness must have been verifiable by the recipient of the transfer -- using many of the same skills that collectors use to appraise collectibles today."

The closed-loop cycle properties of bitcoin, while already existent, will be substantially extended by a marketplace which shares the same principles as the coin. The properties of security, compactness, and unforgeability are also parts of modern cryptocurrencies and their historical study is unmistaken.

Bitcoin as an exchange medium is based on the idea that a currency can be purely virtual. There is no inherent value in bitcoin other than its exchange value. Its scarcity is backed by computing power. It is understood that any medium, physical or virtual, can be used as a medium of exchange, as long as there is an implicit agreement for this and provided it cannot be copied arbitrarily. Bitcoin suggests the use of cryptographically secured digital information as money.

The trading ability used in traditional collectibles historically has evolved through peer-to-peer exchanges. When markets became more popular, they evolved into bazaars, unofficial locations where merchants would gather to exchange goods or services without central control. These trade locations were lated sometimes centralized and controlled by third parties, including land owners in feudal ages, or countries through taxation in more recent years.

Money evolved from collectibles such as shells to scarce commodities such as gold, where it remained decentralized in the sense that no central authority generated the money, but anyone was, in principle, able to mine or find such carriers of value.

As the banking system evolved, the generation of money became centralized and, in recent ages, governments are controlling the supply of money (European Union, 2008).

History of online trade

Since the invention of the Internet, e-commerce has evolved to become a major way of trading and has dominated traditional trade. Money on the Internet is controlled by private corporations in the form of credit card companies and banks. Every payment and every transaction on the Internet is not only authorized by a credit card company, but also transparently visible and censorable as such.

This is starting to change with the introduction of decentralized currencies. While the market share of bitcoin is minuscule compared to other systems of online payment, it is in principle now possible to make transactions which are decentralized and uncensorable.

The two foundational requirements for making online trade possible are money, on one side, and marketplaces on the other. Where money allows the representation of value in a trade, marketplaces allow trade discovery.

Internet marketplaces started in a centralized fashion as well. eBay, Amazon, the Apple and Google app stores, and iTunes are all centralized marketplaces. While eBay allows users to sell and buy from each other, the platform itself is centrally controlled by this corporate third party. Furthermore, all of these stores do not enable anonymity, as they require credit-card-based payments.

The first step towards anonymous marketplaces was made by the infamous yet innovative dark market SilkRoad (Justin Norrie, 2011). SilkRoad operated via Tor (Lee N. , 2015) and allowed anonymous purchases via bitcoin. It focused mostly on illegal markets, in particular drugs. After it was shut down by law enforcement, a series of clones popped up, including SilkRoad 2 (Greenberg, 'Silk Road 2.0' Launches, Promising A Resurrected Black Market For The Dark Web, 2013) , TheMarketPlace (Markowitz, 2013), Agora (Evans, 2014), Evolution (Wired, 2015) and others.

TheMarketPlace in particular is of interest, because it allowed the use of 2-of-3 multisig bitcoin transactions, in which the buyer, the seller, and the market owner participated. Unfortunately, as key discovery happened through the marketplace itself, the central control and unverifiability of keys made this experiment a failure.

While these markets exhibit anonymity, they are not decentralized.

History of OpenBazaar

The idea of OpenBazaar is not new, but it was recently enabled by the invention of blockchain technologies such as bitcoin. The vision for a digital decentralized and anonymous marketplace has a long history in the cypherpunk and crypto communities. We see a clear mention of market contracts in Wei Dai's bmoney post to the cypherpunk mailing list (Dai, 1998). Wei Dai writes:

"Until now it's not clear, even theoretically, how such a community could operate. A community is defined by the cooperation of its participants, and efficient cooperation requires a medium of exchange (money) and a way to enforce contracts. Traditionally these services have been provided by the government or government-sponsored institutions and only to legal entities. In this article I describe a protocol by which these services can be provided to and by untraceable entities."

It is clear that the ideas of bitcoin and OpenBazaar are tightly intervened. Wei Dai's vision for bmoney, which later became bitcoin, includes the important ideas of contract enforcement. These mechanisms become possible through cryptography and through unbreachable contracts. Some of these are utilized by OpenBazaar itself.

The connection between bitcoin and OpenBazaar becomes even more apparent in code included in bitcoin by its original author, Satoshi Nakamoto. In the prototypal C++ client, the source code includes an attempt to implement an OpenBazaar-like market. However, presumably due to complexity associated with such an implementation, the code was removed. Indeed, commit 5253d1ab77fab1995ede03fb934edd67f1359ba8 in the original bitcoin source code is exactly this removal (Nakamoto, Strip out unfinished product, review and market stuff, 2010). It would not be surprising to think that Nakamoto was hoping the OpenBazaar project would become a stand-alone project, using bitcoin as a primitive.

This idea was picked up by Arinich (Arinich, 2013). Arinich explains the necessity of decentralizing online marketplaces and the motives behind Satoshi's redacted implementation. Motivated by the failure of centralized online marketplaces, he introduced many ideas that would shape the world of decentralized marketplaces, including the use of namecoin and arbitration. Several of these ideas remain high-level descriptions, however.

In the meantime, Washington Sanchez and I independently explored the technical feasibility of these concepts. In my post to the LiberationTech mailing list in March (Zindros, 2014) many technical details are explained and suggestions for particular technologies are made. The ideas of 2-of-2 and 2-of-3 multisigs, escrows, reputation systems, high-level descriptions of webs-of-trust, voluntary fee schedules for arbitration, proof-of-burn for identity verification, and timelock-based trust are introduced. Furthermore, Tor as an anonymity-supporting transport layer, bitmessage as a messaging mechanism, DHTs as a storage mechanism, and namecoin for identity management are also suggested.

After we presented these ideas in the LiberationTech mailing list in March 2014, the project DarkMarket was independently started by Amir Taaki and his unSYSTEM team in April 2014 (Greenberg, Inside the DarkMarket Prototype, a Silk Road the FBI Can Never Seize, 2014), although it is unknown whether they were aware of our previous work. After the first quick and dirty implementation of DarkMarket, it was picked up and forked by Brian Hoffman, who then renamed it to OpenBazaar (Hern, 2014). Brian Hoffman, Sam Patterson, Washington Sanchez and I joined forces to lead the effort in developing OpenBazaar as an open source project. We were joined by more than 80 volunteers. We share credit for the creation of OpenBazaar with them. The OpenBazaar was the first practical implementation that solidified these ideas.

Primitives

Blockchain primitives

While decentralized market services have been envisioned for decades, it has only recently become possible to truly realize them. The basic technology that has made this possible is the invention of the blockchain primitive in cryptography and its implementation in bitcoin (Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008).

Blockchain primitives are our primary mechanism for dispute resolution and unbreachable contracts. They allow arriving at consensus in a decentralized way, and enable unbreachable contract mechanisms such as 2-of-2 multisig, 2-of-3 multisig with arbitration, and MAD.

As its importance to this work is foundational and the exploration of these primitives has been limited as far as the academic community is concerned, we spend an extensive amount of time discussing bitcoin and blockchain technologies in this work. These explanations are bibliographic and are based on the original bitcoin paper, the bitcoin developer guide (Bitcoin Developers), and previous work of the author. The acquainted reader can skip the next sections.

Hash primitives

Cryptographically, we require certain modifications to the traditional models of hash function collision resistance. The new collision-resistance requirements are near-collision resistance and range-collision resistance. These are required for the hash functions SHA256 and RIPEMD160. The near-collision requirement is already known in the cryptographic community. For example, near-collision properties of SHA0 have been extensively explored (Biham & Chen, 2004).

Near-collision requires: It is not possible for a polynomial agent to find x and y under some key parameterization of a given hash-function H such that the Hamming distance between H(x) and H(y) is below a given threshold requirement ε. Usual collision resistance can then be expressed as near-collision resistance setting ε = 0. Therefore, near-collision resistance requirements are stronger than traditional collision resistance. More formally:

$$\forall PPT\ \mathcal{A}(1^{\kappa}): P\left[(x, y) \leftarrow \mathcal{A}, \left||H_{\kappa}(x) - H_{\kappa}(y)|\right| < \varepsilon\right] \in negl(\kappa)$$

Near-collision resistance of SHA512 is required for one of our proposed proof-of-burn mechanisms. Of course, formalizing the near-collision requirement for SHA512 is not possible, as it is not a keyed hash function.

Range-collision resistance is a different requirement imposed by blockchains. It extends traditional collision resistance as follows: We require that, given some threshold value ε and a given challenge y, it is not possible for a polynomial agent to find x such that ||H(x) – y|| < ε under some keyed parameterization of H and for some norm on the output space, for example arithmetic distance. Setting ε = 0 produces the traditional collision resistance definitions. More formally:

$$\forall y\colon \forall PPT\ \mathcal{A}(1^{\kappa})\colon P\big[x \leftarrow \mathcal{A}, \big||H_{\kappa}(x) - y|\big| < \varepsilon\big] \in negl(\kappa)$$

Range-collision resistance for an image of y = 0 is required for the security of bitcoin.

Bitcoin

Bitcoin and blockchain primitives are used in OpenBazaar to enable unbreachable contracts. We chose Bitcoin as a payment system because it is the most stable among cryptocurrencies.

For use in OpenBazaar, we considered several alternative cryptocurrencies. Some cryptocurrencies were considered for their more favourable properties, in particular as it pertains to unbreachable contracts. For example, Ethereum-like (Wood, 2014) cryptocurrencies could have been used to achieve better unbreachable contracts due to the expressiveness of their Turing complete properties. However, these altcoins are quite unstable and experimental at the moment, and did not allow for a concrete implementation. Turing completeness and storage facilitation possibly allow simpler implementations of decentralized marketplaces such as OpenBazaar. For example, trust and ratings can be stored in decentralized stores offered by these cryptocurrencies, and more complex trust mechanisms such as time locking or complicated financial reward and punishment systems could be implemented. More research and development is needed to explore these options.

Introduction

Bitcoin is an experimental decentralized cryptocurrency, a monetary system without central control whose integrity is based on the foundations of cryptography. In the current section, after we develop the problem and the reasons of existence of a decentralized cryptocurrency, we will present the basic operating principles of the system, as well as its foundation from an economic and mathematical point of view. Some technical details of the protocol and the intuition of correctness and completeness will be presented, but without formal mathematical proofs. We will illustrate the reasons the protocol is able to secure transactions, avoid unwanted transactions, and achieve anonymity. The goal of this section is the presentation of the cryptocurrency and of the basic operating principles, and we attempt a complete description of the technical principles, but without getting into unnecessary details. Readers already acquainted with bitcoin can skip this section.

The problem

In the crypto community, particularily in the cypherpunk community (Manne, 2011), there exist the desire to preventatively use cryptography to circumvent the ability of privacy intrusion from states or other actors who have the ability to eavesdrop on data exchange transfers, legally or illegally (Hughes, 1993). Traditional online payments today do not exhibit the advantages of offline payments. In particular, there is no ability to exchange money anonymously, as all online transactions take place through specialized, centralized services who have access to the identity of their participants. Services such as PayPal require to know the identity of the people who transact and so anonymity is lost. The same is true for all sorts of electronic payment systems, including of course those that take place using credit cards and which require services such as Visa and Mastercard. In any case, it is impossible to exchange money using a pseudonymous identity, as it is required to validate the true physical identity of someone through strict mechanisms such as opening a bank account which requires a national government-certified identity with a photograph, as well as the validation of one's place of residence, with the small exception of paysafe cards and similar services.

These services which offer the ability of online payments secure the transactions of their participants, but do not allow anonymous transactions such as those possible in cash, and in addition they charge a small fee for every transaction, making it impossible to trade very small amounts. More importantly, these centralized services excert control over which transactions are valid and which aren't, directly censoring transactions and accounts. There are several serious cases in which PayPal has arbitrarily censored important causes or withheld money from its clients. In fact, this practice is so habitual that there is a surprising number of publicly documented cases, often related to politics and activism (Orlowski, Paypal freezes Cryptome, 2010) (Orlowski, Cryptome: PayPal a 'liar, cheat and a thug', 2010) (Orlowski, PayPal says sorry to Cryptome, 2010) (Morran, 2011) (Smith, 2010) (Addley & Halliday, 2010) (Whitney, 2011) (Corbin, 2014) (Koetsier, 2013) (Waxman, 2011) (Novak, 2013) (Lokot, 2015) (Dolgov, 2015) (Freedom House, 2015) (The New York Times, 2003).

Finally, even the use of cash or traditional fiat currency such as the euro and the dollar has its problems, as in this case the currency is produced by a government, with the ability of centralized macroeconomic control which may or may not be desired by the currency users. Following the tradition of the cryptographic community in the preventative use of cryptographic technology without assuming any trust towards third parties such as the government, current systems are unsatisfactory.

Solutions that have emerged to solve this problem in various times are, as mentioned in the monetary history section, the use of gold or other scarce commodities which have some inherent value, with prices determined by a free market. However, this cannot be used for online payments, as it has limited usability, and is slow and insecure due to the possibility of theft or loss.

This problem was the motivation for the search of a better solution which is based on digital currencies.

Bitcoin history

Bitcoin started in 1998 when Wei Dai published in the famous practical cryptography and anonymity mailing list ”cypherpunks”. He posted a sketch where he explained the idea with which an economy not requiring central control would be possible. Cypherpunks are known for their desire to use cryptography preventatively to achieve anonymity, privacy, and political and economic change through software engineering (and not simply academic exploration) of systems whose goal is to be used widely and in production. Wei Dai's sketch entitled “bmoney” (Dai, 1998) set the foundation for the development of Bitcoin.

In 2009, Satoshi Nakamoto developed the first version of his software in C++ and at the same time published a nominal paper with the title “Bitcoin: A Peer-to-Peer Electronic Cash System” (Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008). That paper founded the theory which was necessary for the development of the system and included a short mathematical proof of the security of the system. The software was designed not only as a simple proof-of-concept, but as a complete production system aimed to scale, a move that follows the cypherpunks tradition. The software is open source and published under the MIT license, a tradition we also followed with OpenBazaar (see below). Satoshi Nakamoto suddenly disappeared after the completion of the first version of Bitcoin (Wallace, 2011). Much speculation surrounds his person, as it is said – in the way traditional among cryptographers – that due to paranoia he never revealed his real name, and the particular identity is nothing but a pseudonym. Noone has met him in person, there is no information about him or even a picture. He claimed to be of Japanese origin, but none of his writings or source code have a word of Japanese. There have been numerous failed attempts to identify him (Jeffries, 2011) (Davis, 2011) (Penenberg, 2011) (Ormsby, 2013) (Markoff, 2013) (Wile, 2013) (Biggs, 2013) (Greenberg, Bitcoin Community Responds To Satoshi Nakamoto's 'Uncovering' With Disbelief, Anger, Fascination, 2014) (Winton, 2014) (Clinch, 2014) (Frisby, 2014).

The Bitcoin system is maintained and developed today in open source form through the Bitcoin developers community.

The idea behind Bitcoin

Bitcoin's main goal is to solve the problem of central monetary policy. It allows fast payments which are securely confirmed within an expected 10 minutes, without requiring a trusted third party. Its value emerges from the free market as a commodity which exhibits the requiements of a good exchange medium (see “Origins of money” above) and is backed by computing power. It secures transactions using modern, provably strong cryptography, and provides anonymity to the transating parties.

Decentralization

The network decentralization in bitcoin is achieved using a peer-to-peer network to which all participants who wish to use bitcoin for their transactions connect. The participants run the bitcoin software on their computer.

This can be either the classic Satoshi bitcoin client which has since been extensively improved by other developers, or some other wallet which implements the bitcoin protocol. The latter has become a more popular option among users, as alternative wallets can have desirable properties such as Simple Payment Verification support and can work on smaller and more limited devices such as mobile phones where storage, bandwidth, memory, and battery are scarce. Both the protocol as well as the software are open, as is required in modern cryptography. The users can check if the source code follows the theoretical model and matches the security proofs explored in the theory, or at least assume that this can be done by any expert who desires to do so. Bitcoin software is available for all modern operating systems.

After the user has run the program on their computer, it then connects to other peers on the network using typical peer-to-peer discovery schemas. These include the use of predefined IP lists from recently known Bitcoin clients for the initial connection, the use of some public servers for the use of other nodes (e.g. a deprecated peer discovery mechanism was through IRC), and of course the ability to manually connect to a good known IP address. In this way, each node connects to a number of other peer nodes on the network (Antonopoulos, 2014). Furthermore, each node produces a public key which they publish on the network and keeps the respective private key secret on the local system. The secret key of the user is required to make the payments, and so must be properly protected, as an adversary could use it to perform transactions with the user's money in their stead.

Payment schema

A coin owner transfers their coin by publishing a digitally signed statement. In particular, the sender of money, Bob, digitally signs his desire to perform a transaction, including the recipient of the money, Alice, in the message. By verifying Bob's signature, Alice can ensure that Bob is truly the one who authorized the transaction.

The first problem in such a schema is that the money must somehow be produced. It would be undesirable if each user were able to produce money arbitrarily and then sign it to produce a valid transaction. In short, we want each node on the network to be able to confirm that each user in fact has the money they are transfering. The only way to achieve this, as there is no central trusted third party, is to publish on the network exactly who has how much coin. This is required so that we can confirm a coin is valid and that it can be spent by their owner only once.

The network maintains collectively a list of unspent coins, the unspent transaction output set; the technical definition of the term is introduced later. In this way, each recipient can confirm that the money they received from the sender were owned by the sender at the time of the transaction. When a new full node is connected to the network, the nodes with which it connects inform it about the current unspent transaction output, the list of unspent coin. More specifically, when a node is newly connected to the network they are informed about the full history of transactions which took place on the network since the beginning of time. When a node reconnects to a network, they use a synchronization mechanism to become informed about transactions that took place since the last time the node was connected to the network.

For a node to be able to maintain a valid list of unspent coin, it is also required for new transactions to be published on the network while they take place. This is achieved through a mechanism known as broadcasting. During broadcasting, when two nodes perform a transaction, they both broadcast to their neighbours the details of that transaction. These details include the sender, the recipient, and the amount of the transaction (exact details of what a transaction contains are presented in the graph-theoretical modeling below). The neighbours recursively keep publishing this transaction until the whole network becomes aware of it.

Anonymity

The publishing of transactions on the network destroys their anonymity, as each node has access to the full history of all the transactions ever made by everyone else. Nonetheless, it is easy to create a schema in which anonymity is preserved even though transactions are published. Bitcoin uses a system in which a different private key is used for each transaction. This means that we use a particular public key to receive a certain amount of money and then its respective private key to sign-off the spending of that same amount of money. However, we will use a different public key to receive money each new time – and the respective secret key to spend it.

The use of a different public key for every receiving transaction is easy and is already automatically performed by most bitcoin wallets. Ensuring that there is no correlation between IP addresses and keys and that each key remained unnamed, the network can achieve anonymity. This is possible because the correlation of use between two keys owned by the same physical person is not trivial. However, the correct avoidance of correlation attacks through forensic analysis of the blockchain is a difficult manner, and true anonymity is rarely achieved in practice (Reid & Harrigan, 2013).

In this schema, every user has full knowledge of the statistical elements of the global economy, in detail, but anonymously. They can know exactly the amount of bitcoin exchanged daily on the whole network, the amount in each transaction, the fequency of transacting, etc. but not the identity of those who execute these transactions. This type of anonymity is similar to the anonymity that exists in public trading of companies where the exchange of shares is published among the financial statistics of the company, but without publishing the identity of the buyer or seller.

Technical bitcoin background

Let us now proceed to a more technical analysis of the nature of the bitcoin currency, the system, network, and security in detail. This technical analysis will be necessary to understand the self-enforcing contracts employed in the security of the OpenBazaar protocol. This is required to explain how exactly it is possible to prove that someone owns a coin that indeed is his own and he did not produce arbitrarily. It also allows us to illustrate that a coin can only be spent from its rightful owner and prove that the probability of double spending of the same coin that we can achieve is negligible. To achieve these properties which are required for a system that secures transactions, it is useful for each coin to have an identity and to be defined as an entity. As this is about a decentralized network, it is impossible for each coin to have an incremental number or have its identity produced by a central authority, but it has to be created and verified in a decentralized manner.

A coin in Bitcoin's system is defined as a chain of digital signatures. The coin begins its life through mining (see next section) that results in the production of the coin which is a string of data that contains information about the coin such as, for example, its nominal value. At the transfer of the coin from the first owner, Bob, to the second, Alice, Bob concatenates the initial coin with the public key of Alice. Next, he applies a hash function to the result and signs the hashed value using his own private key. This is now a coin that belongs to Alice.

When Alice's turn comes to pay Charlie using the money she received from Bob, she will concatenate her coin with Charlie's public key, then hash the result, and sign the result with her own private key. As only Alice knows her private key, the coin can only be spent by Alice and no one else. The fact that the coin is indeed destined for Charlie can be verified by himself simply by examining the contents of the coin he recieved, as this must contain his public key. If it doesn't, then Charlie can refuse the transaction. The fact that the coin indeed came from Alice can be verified by Charlie easily by verifying Alice's digital signature. Of course, in addition to a simple signature check, it must also be verified that the coin was indeed in Alice's posession before it is transfered to him by verifying that the coin was part of the unspent transaction output set.

The sequence of these signatures is shown in the diagram Figure 2: Chain of coin transactions in bitcoin.

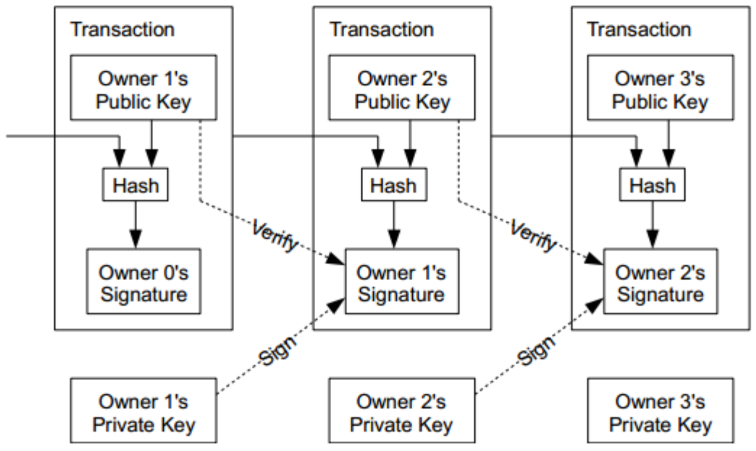

FIGURE 2: CHAIN OF COIN TRANSACTIONS IN BITCOIN

Double spending

While we have made sure that only the rightful owners of a coin can spend it and that the receiver of the coin can verify that the coin belonged to the sender as well as the fact that the receiver is actually himself, we have not yet achieved any insurance as far as coin double spending is concerned. The problem occurs in the fact that a sender can spend the same coin multiple times. As the network is decentralized, a double spending may not be immediately noticed, but may require several minutes until the network becomes aware of it.

A different way to look at the problem is that double spending could occur in a longer timespan such as a month or a year. In this case it is impossible to verify the true recipient of a coin, as a malicious adversary can claim that a transaction took place much earlier in the past, as no trusted third party that stores the order in which transactions took place exists. It is also a problem that is impossible to solve without modifying our initial scheme. The obvious solution that rejects a transaction if it occurs multiple times is not acceptable, as this would allow an adversary to cancel previous transactions by double spending in the future, something that would cause both himself and the recipient of the money to lose that money, but remains unwanted, as we would like the transactions to be confirmed and the reciepient to be certain that they have received the money and that the money can be indeed spent in the future. Furthermore, an adversary cannot be detected and isolated, as, due to anonymity, it is impossible to know which IP address they use, and can continue to act maliciously using a different public key each time.

This problem is solved by Bitcoin using an 'arrow of time' system which enables us to know with certainty the order in which transactions took place. Supposing that it is possible to know which transaction happened before which, our system is easily fixed by assuming that the only valid transaction is the first transaction in which a coin was spent. This way, when the coin is spent, the receiver can verify that the coin was not spent again in the past and accept the transaction, or reject the transaction if there has been a double spend. In addition, the whole network can reject double-spending transactions, i.e. transactions which spend money that is not within the unspent transaction output set.

Of course, it is impossible to rely on digital signatures as a certification of date of a transaction commencement, as a malicious adversary can easily sign false statements. In traditional academic cryptosystems, the arrow of time is implemented with a publication (for example in a newspaper or on Usenet), something someone can verify independently. This is something impossible to do in the case of Bitcoin, as we start relying on a trusted third party which is begging the question. Bitcoin uses a new system to achieve the arrow of time which is implemented by taking advantage of proof of work chains termed the blockchain.

Proof of work

The bitcoin network nodes share a common chain which contains the transactions in the order they took place. This chain is common between all of them (modulo forks) and consists of a linked list of blocks. Each block contains a list of transactions that took place near a specific moment in time. To build the chain, every block is hashed to produce a hash. Then, every next block contains inside its contents the hash of the previous block. This results in every next block in the chain requiring the previous block to have been computed before it can be computed. Due to the collision resistance of hashes, the previous block will be required to be hashed before the next block can be computed.

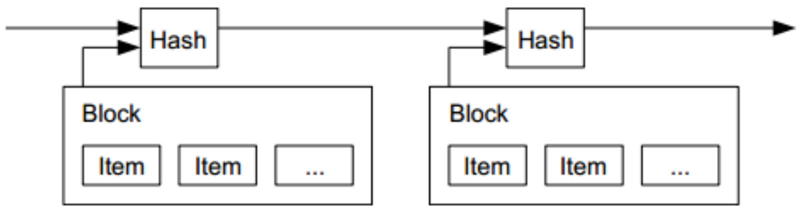

FIGURE 3: A PROOF-OF-WORK CHAIN OF BLOCKS IN BITCOIN

Even though blocks can be arranged chronologically, someone can easily change the order of two blocks within the chain by changing the respective hashes and producing new ones wherever required, an undesired intervention, as this would allow an adversary to fake the order of transactions in time. To avoid this problem, bitcoin introduces a technically difficult cryptographic problem which is now associated with the production of every block.

From the assumption that the hash inversion is hard, proof of work is established by seeking a range-collision of the hash function on the block. More specifically, it is asked that the hash of a block is smaller than a given number, the target, which is calculated collectively by the peer-to-peer network. The freedom given to each full node when producing a candidate block when it comes to changing the resulting hash is to modify a nonce, a value which is concatenated with the rest of the block data and serves simply to modify the resultant hash value. As we believe hash functions are one-way, the only way to achieve a hash value within the desired range is with a series of brute-force trials of different nonces, a computational process which is exponential in terms of the number of digits of the hash function output. By setting the correct target value, the peer-to-peer network can alter the difficulty of this procedure.

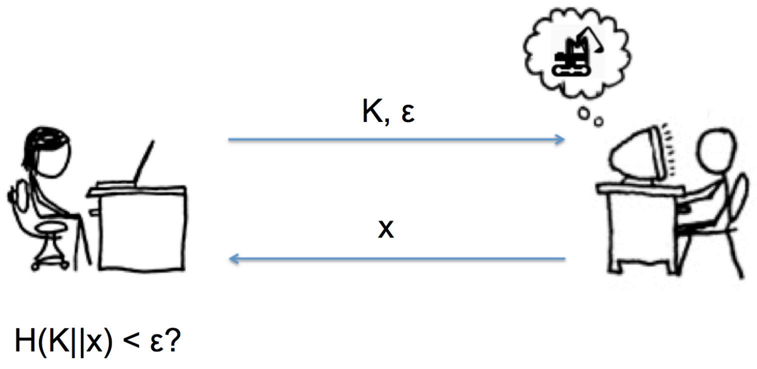

FIGURE 4: THE PROOF-OF-WORK PROTOCOL

Figure 4: The Proof-Of-Work protocol visually illustrates the proof-of-work protocol. Alice, on the right, challenges Bob, on the left, with a proof-of-work challenge. Given a known target ε, and a fixed pre-image K, she asks Bob for a nonce value x, such that the hash target ε is met under a hash function H.

A high-level implementation of a proof-of-work algorithm is shown in Listing 1: A Simple Proof-Of-Work Algorithm.

```
x = rand();

do {

    ++x;

} while (H(K||x) >= ε);

return x;
```

Listing 1: A Simple Proof-Of-Work Algorithm

The proof-of-work target is evaluated collectively by the network using a prespecified algorithm. Consequently, all the nodes of the network simltaneously attempt to produce a new block which contains all the transactions that had not been contained so far within the accepted blockchain in addition to meeting the target proof-of-work requirements. As the network can control the difficulty of proof-of-work, the expected frequency of block generation is controlled.

It is obvious that, while proof-of-work is required from each node which produces a block as the last item in the blockchain, for this block to be accepted as valid it must only contain valid transactions. This is ensured because all other nodes will never accept a block at the end of the blockchain which contains the double-spending of the same coin or the spending of a coin that is not part of the unspent transaction output set.

When a new block is computed by a node, this is then broadcasted to all their neighbours which, in turn, transmit it to the whole network. While the frequency of blocks generation is controlled so that it is small enough (for example in the bitcoin blockchain at an expected rate of 1 block per 10 minutes) so that two blocks are not produced simultaneously from two different nodes, this can on occasion take place, as the brute-force hash inversion cannot be predicted. In this case, when a node receives both these blocks that extend the blockchain, a blockchain from the two can be chosen arbitrarily among the blockchains that share the same number of blocks. A node accepts a block as valid by accepting the transactions contained therein as valid and starting proof-of-work on a new block on top of the existing blockchain ending on the accepted block. In case a block is not accepted and an alternative block is accepted, the transactions within the unaccepted block are not lost; they will instead be included in the alternative accepted block, or in a future block. The node which will continue extending the blockchain is the one who ultimately decides which fork of the blockchain will be treated as valid, and which will become an orphan.

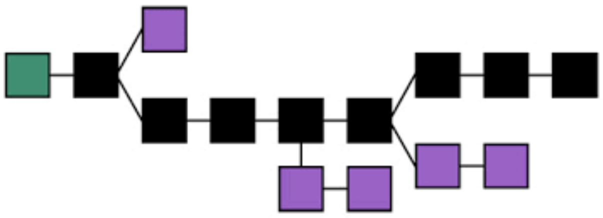

FIGURE 5: A BITCOIN BLOCKCHAIN WITH SOME ORPHAN FORKS

In Figure 5: A bitcoin blockchain with some orphan forks, each square constitutes a block which contains multiple transactions which took place in nearby times. The currently valid blockchain is shown in black. Orphan chains are shown in purple, chains that were created by extending the current chain at the same time as other extensions, which were eventually deemed invalid as the winning miner chose the current blockchain. The first block, or genesis block, is shown in green.

The existence of a transaction in a block confirms that the transaction has taken place. The deeper the block is within the blockchain, the more conformed a transaction becomes. This becomes clear if we consider the time required for an adversary to modify the blockchain in order to reorder transactions, to cancel a transaction, or to add a transaction before a given transaction. This time becomes exponential in the number of blocks that have followed the block in which a given transaction is confirmed. Therefore, a user wishing to verify that a transaction was confirmed can wait until 5 or 6 confirmation blocks appear after the block that first confirms their transaction so that they can be sure the transaction will not be altered. The exact calculation for the probability of cancelling a transaction for a determined adversary given a number of confirmation blocks after the block containing a transaction and with a given CPU power can be found in the original Nakamoto paper.

The modification of the blockchain is hard for an adversary as the blockchain is constantly extended. To obtain control of the blockchain, the adversary would need to control the majority of the CPU power of the network. Bitcoin's security is founded on the fact that this will not occur. Various fork-based attacks have been explored in the literature (Eyal & Sirer, 2014), but the security of bitcoin has also been formalized and rigorously explored (Garay, Kiayias, & Leonardos, 2015).

Mining

The creation of new blocks is termed mining. The first mining was done by Satoshi and the block that was produced is called the genesis block. Every valid chain starts from the genesis block. The coinbase transaction of the genesis block contains a timestamping method, which indicates that no bitcoin existed before the 3rd of January 2009. This is achieved by publishing a news story from that date within that block:

"The Times 03/Jan/2009 Chancellor on brink of second bailout for banks"

As this event did not take place before that date and would have been impossible to predict the Times publication exactly, this indicates that bitcoin was not a pre-mined currency.

Every block mined contains a reference to the miner that generated it. Mining a block, in addition to confirming the transactions it contains, rewards the miner with a number of bitcoins that are created for the specific block. This happens through a transaction known as the coinbase transaction, and this is the way new bitcoin is generated. The amount of coin mined in each block is preagreed on by the network and is steadily reduced so that the available amount of coin in the market converges to 21,000,000 bitcoins. Currently, each coinbase transaction generates 25 BTC. This amount is halved every (expected) four years. The convergence function is shown in Figure 6: Total bitcoins over time.

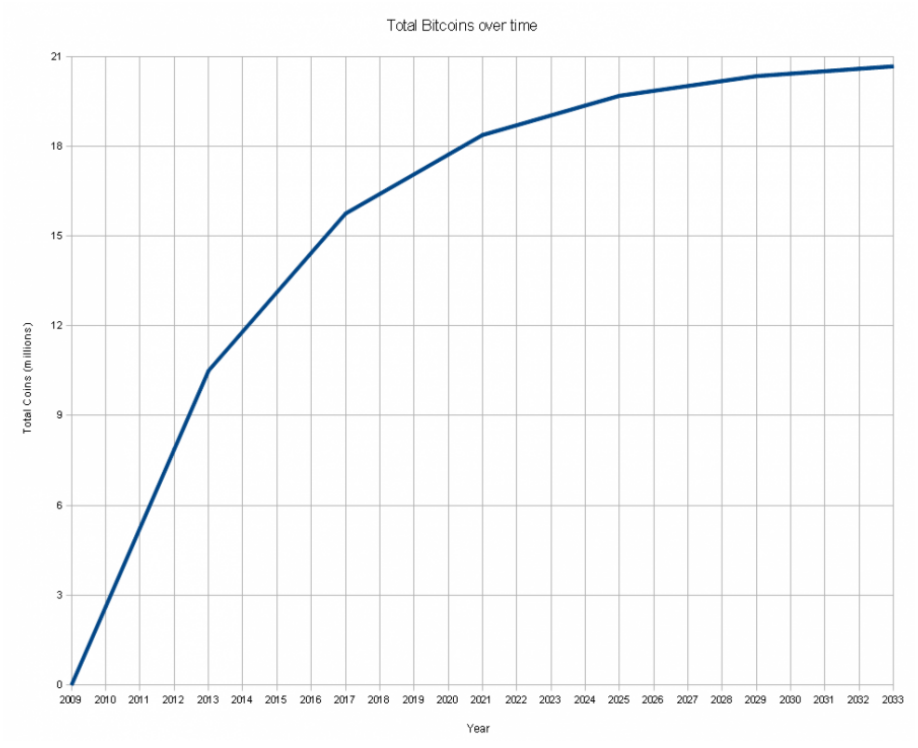

Figure 6: Total bitcoins over time

Technical details

A few technical details on the implementation of the bitcoin cryptography are appropriate. Bitcoin uses a signature scheme to validate the owner of coin during a transaction. The scheme uses elliptic curve DSA. The hash function used for proof of work is a double SHA256 function. SHA256 is also used to protect the public keys of coins before being spent; this is the pay-to-pubkey-hash functionality.

The Bitcoin DAG

The above explanation of bitcoin's functionality is quite simplified. To fully understand the abilities bitcoin enables, which we utilize to a great extend in this work, the set of transactions must be modeled like a directed acyclic graph (DAG) in which transactions are the nodes and edges connect transaction outputs with transaction inputs. Note that this is dissimilar to simplistic graph-theoretic modeling of economic systems where agents are nodes and transactions between them are edges.

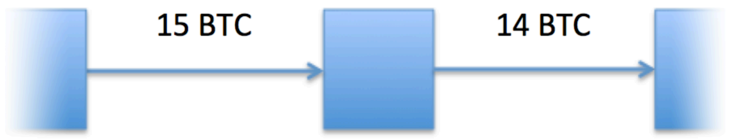

FIGURE 7: A BITCOIN TRANSACTION NODE

Figure 7: A bitcoin transaction node illustrates a bitcoin transaction node, indicated by a blue square. It has one incoming edge of value 15 BTC and an outgoing edge of value 14 BTC. Its input edge is connected as an output edge to a previous transaction (shown partially on the left), while its output edge is connected as an input edge to a next transaction (shown partially on the right).

In the bitcoin transaction graph, each transaction can have multiple inputs and multiple outputs. Edges are directional and indicate monetary value flow. The Kirchhoff property mandates that the outputs are at most equal to the inputs and can be summarized as follows for all valid transactions t:

$$\forall t \sum_{o \in out(t)} v(o) \leq \sum_{i \in in(t)} v(i)$$

Where in and out denote the set of all in-edges and out-edges of a transaction respectively and v is a value function that evaluates the amount of coin flow of an edge.

The Kirchhoff difference is then paid as a fee to the miner who first confirms the particular transaction, and the collection of all fees is the incentive to the miner, in addition to the coinbase amount:

$$fees(block) \stackrel{\text{def}}{=} \sum_{t \in tx(block)} \left[\sum_{i \in in(t)} v(i) - \sum_{o \in out(t)} v(o) \right]$$

In the figure above, the fees paid to the miner by the particular transaction amount to 1 BTC.

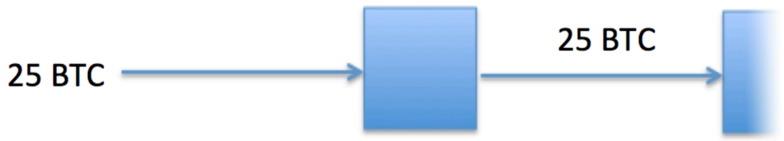

FIGURE 8: A COINBASE TRANSACTION

When a transaction is created, all inputs and outputs must be specified. However, some input edges can be dangling; this means that they are input edges to some nodes, but they are not connected to output edges. These are called *coinbase* transaction inputs and they denote coins generated as reward for mining. A coinbase transaction example is shown in Figure 8: A coinbase transaction.

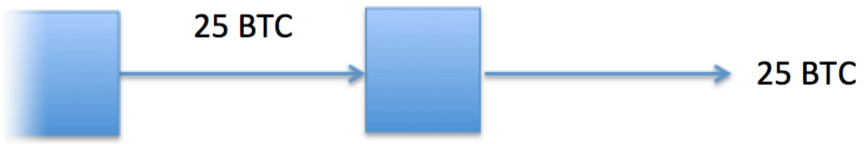

FIGURE 9: AN UNSPENT TRANSACTION OUTPUT

Output edges can also be dangling; such edges are output from a particular transaction node, but are not connected to some other node as inputs. These outgoing edges constitute the *UTXO*, the *unspent transaction output set*, which is available for spending (Buterin, 2014). They will be connected to input edges in the future when the money in the transactions is spent. An example unspent transaction output is shown in Figure 9: An unspent transaction output.

Edges are weighted with a weight-function v which associates a monetary value to each edge.

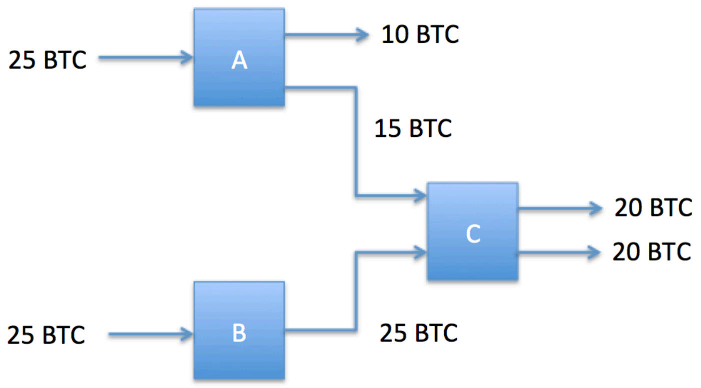

FIGURE 10: A COMPLETE BITCOIN TRANSACTION GRAPH

In Figure 10: A complete bitcoin transaction graph, a complete bitcoin transaction graph is shown. Of course, the real-world transaction graph is much larger. In this transaction graph, transactions A and B are coinbase transactions with dangling inputs of value 25 BTC each. Transaction A has two outputs; one with value 10 BTC and one with value 15 BTC. This could be due to a payment being made for a product of value 15 BTC. The 10 BTC output of transaction A is called a *change* output. Transaction C is funded with multiple inputs; 15 BTC from transaction A and 25 BTC from transaction B. Because C does not pay any fees to miners, it may take some time until it gets fully processed.

It is now easy to visually illustrate the double spending attack as seen in Figure 11: A double spend. One specific output edge from the UTXO is attached to multiple input edges in different transactions addressed to different people. If these transactions are not broadcasted simultaneously but there is delay in their propagation, they may be seen by the network at different times. If merchants do not wait for confirmation, an adversary could trick them into both receiving the same coin.

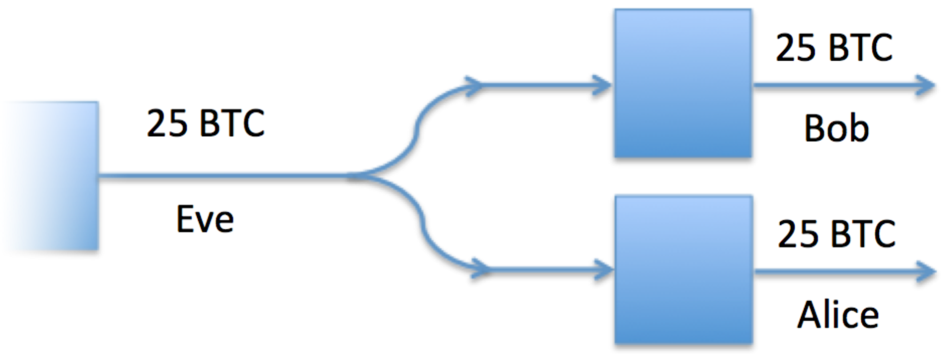

FIGURE 11: A DOUBLE SPEND

The Bitcoin script

In the bitcoin DAG, edges also have associated code with them, which specifies what constitutes a valid spending of a dangling edge. In bitcoin, this code is written in a language termed 'bitcoin script' which is a push-down automaton-style scripting language.

The scripting language works by executing one command at a time from a list of commands. The list of commands cannot have any branching or other flow operators such as conditionals; they are all executed in order. The only flow-interrupting commands available are exception-style commands which raise exceptions and abort the whole execution flow. The scripting language operates on a data stack which contains pieces of data to be processed.

The script is executed when an attempt is made to connect a dangling output to an input of a new transaction, thereby indicating a spending. The script is then executed by giving it a particular set of inputs associated with both the old and new transaction. The spender is responsible for providing the initial stack that the script will be executed upon. The script code is then executed in full. If an exception occurs, the spending is aborted, indicating that the spending was not rightful; if execution finishes without exception incidents, the data stack is examined and compared to the stack of a single element, TRUE, which is expected to be found on the stack. If it is found, then the transaction is considered rightful and the money is considered transferred, otherwise the transaction is considered invalid and aborted. It is worthwhile noting that there is no other operation occuring when money is spent or transferred; there is no ledger or decentralized database maintaining account balances other than the transaction DAG.

Bitcoin scripts can express a multitude of scenarios, the most common of which is called the payment to a 'public key hash'. In this scenario, the spender must provide proof of ownership of the private key associated with the public key to which the payment is being made. This is performed by giving a digital signature as part of the initial stack that the script runs on. The digital signature signs the transaction that contains the input the spender wishes to connect the dangling output to, thereby indicating the intention of the spender to send the money to a particular new transaction.

The standard bitcoin script is shown in Listing 2: The standard bitcoin script.

```
OP_DUP
OP_HASH160
<pubKeyHash>
OP_EQUALVERIFY
OP_CHECKSIG
```

LISTING 2: THE STANDARD BITCOIN SCRIPT

This script is executed with two parameters initially placed on the stack; the "sig" parameter and the "pubKey" parameter. Upon execution, the DUP operator duplicates the alleged "pubKey" on the top of the stack, so that the stack now contains two instances of the "pubKey" and one instance of the "sig". The HASH160 operator pops the top of the stack and hashes it, then pushes the result of the stack to the top. The "pubKeyHash" constant is a constant provided hard-coded in the output script by the payer which indicates which public key is authorized to spend the money; this is the bitcoin address of the receiver, for example mz6KvC4aoUeo6wSxtiVQTo7FDwPnkp6URG. This simply pushes a constant on top of the stack. The following EQUALVERIFY operator then compares the hard-coded pubKeyHash with the result of the HASH160 operator, the currently two top items in the computation stack. This is an example of an operator that can cause an exception; if the equality fails, execution stops through an exception, and so payment fails. This ensures that only the rightful owner can spend the money. Finally, the CHECKSIG operator uses the pubKeyHash and sig values that have been pushed on top of the stack to perform a signature verification on the transaction to which the dangling edge was connected as input. The CHECKSIG operator returns either TRUE or FALSE and leaves the result on the computation stack, thereby indicating a successful or failed spend respectively. The pay-to-pubkey-hash script is the standard method of making payments in bitcoin. While some of the original bitcoin versions supported direct payments to public keys, it is considered more secure to pay to hashes of public keys, as this provides an additional layer of protection in case elliptic curve cryptography is weakened by attacks one day.

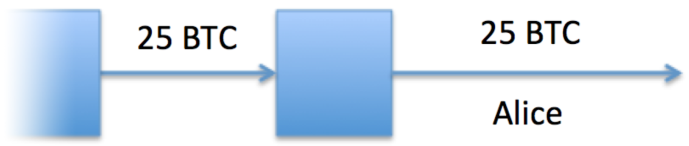

FIGURE 12: AN UNSPENT PAY-TO-PUBKEY-HASH TRANSACTION

Figure 12: An unspent pay-to-pubkey-hash transaction illustrates an unspent transaction output of value 25 BTC which is payed to Alice's public key. We will use this

notation to illustrate transactions paid to specific public keys using the pay-to-pubkey-hash standard output script. That is, we will note the transaction output amount on top of an edge and a description of the output script at the bottom if the output script is significant.

Multisig

Another usual output script is a multisig script. This allows any of k signatures out of n to spend the money reserved in the dangling output. The value k is called the *threshold* value. This allows many applications, some of which we propose in OpenBazaar for trust reasons. It is fundamental that the n keys are not necessarily controlled by the same party. Bitcoin provides specialized operators for multisig verification, in particular the CHECKMULTISIG operator. Multisig scripts are similar to the simple pay-to-public-key-hash script, except they allow for and may require multiple signatures.

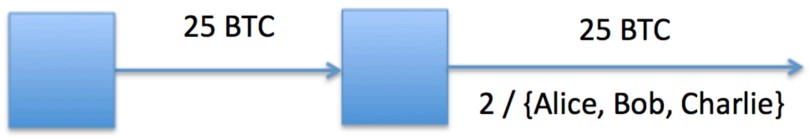

FIGURE 13: A 2-OF-3 MULTISIG TRANSACTION

Figure 13: A 2-of-3 multisig transaction illustrates an unspent 2-of-3 multisig transaction in which any 2 of Alice, Bob, and Charlie can sign together to spend the output. We will use this notation to denote multisig transaction output scripts when relevant.

For complicated output scripts such as multisig, bitcoin also employs an indirect script reference mechanism called p2sh / pay-to-script-hash (Andresen, 2012). These output scripts are similar to regular (bare) output scripts, except the script code itself is hashed. This way, the burden in fees of putting the output script in the blockchain lies on the spender, not the payer.

Multiple transaction inputs

A transaction can have multiple inputs as seen in Figure 10: A complete bitcoin transaction graph. These inputs must all be connected to certain outputs from previous unspent transactions. All these outputs contain scripts that must be given appropriate script parameters to confirm the fact that they are valid to spend; for standard output scripts, these parameters importantly contain digital signatures on the newly created transaction authorizing the connection of output and input edges. It is significant that some transactions can contain inputs that are signed with keys controlled by different parties. In that case, all parties must authorize a transaction for it to be valid. This disparity is something we will leverage to achieve trust in a game-theoretic setting for OpenBazaar, combining it with the multisig setting above.

It is important to notice that a transaction becomes valid atomically; a transaction is either valid or invalid. It is not possible for a transaction to be partially valid. This concept is important when multiple parties are putting inputs into a transaction. If only some of the parties sign the transaction outputs connected as inputs to the new transaction, the transaction will be invalid. The new transaction will become valid and will be a candidate for inclusion in the blockchain only when all parties have completely signed. This allows for a safe mechanism of payment where a party only pays if their counterparty also pays. This schema is further explored in the game theoretic sections of this work.

Threat model

Before we explain the trust issues of OpenBazaar, we would like to introduce our threat model and exactly what our trust goals are.

OpenBazaar makes important assumptions about the strength of its adversaries. To understand how we designed and developed OpenBazaar, it is crucial to understand who the adversaries of OpenBazaar can be, what resources they are able to employ, and what their goals are. Furthermore, it is important to understand what the adversaries are not capable of.

Assumed adversaries

Our adversaries can be broadly categorized in 4 different categories:

- Malicious users
- Malicious corporations
- Malicious governments
- Malicious developers

Each of these constitutes a separate entity with different resources and different goals. These are explained below.

Malicious users

A malicious user is a user who tries to break the security of OpenBazaar, usually for financial gain. We treat malicious users as game-theoretic agents who are able to invest approximately as much as they would win out of a security breach, as long as their winnings are significantly larger than their losses.

The goal of malicious users is to make money. The two primary ways of making money by breaking the security of OpenBazaar are these:

- Being able to receive a product without making a proper payment
- Being able to receive money without delivering a product

As these attacks are financially incentivized, there is practically no limit as to what capital can be invested in such attacks if it's possible to earn it back. However, we assume that the monetary sums that could be spent are unable to break the bitcoin network security or our cryptographic primitives, for example by brute-force attacking 1024-bit RSA keys (Bos, Kaihara, Kleinjung, Lenstra, & Montgomery, 2009). At this time, this assumption limits the monetary amounts that are safe to exchange over OpenBazaar to a few million US dollars.

These attackers are the easiest to model, as they play within the closed OpenBazaar system and can be treated game-theoretically. Generally, in our games, these agents can be assumed to be ε-good, meaning they will not attempt a malicious strategy if there is no financial gain in it (for a formal definition of ε-goodness, see page 77). We aim to fully protect users from such malicious actors.

Malicious corporations

Certain corporations may find the OpenBazaar network undesirable and may want to break its security in order to bring it down. Their financial incentive may not be part of the closed OpenBazaar system: They may be able to make profits outside of OpenBazaar by compromising the reliability and security of OpenBazaar.

The goals of such agents are the following:

- Bringing down the majority of OpenBazaar nodes
- Disrupting the majority of connectivity of the OpenBazaar network
- Breaking the trust people have on the OpenBazaar network

"Breaking the trust" here means creating arbitrary buyers and sellers that do not follow the expected strategy and default on their payments or shipping; or giving false trust to untrustworthy nodes; or giving false negative trust to trustworthy nodes.

The incentive for such corporations may be that they are losing money because of competitive sales on the OpenBazaar network (Allison, 2015).

We currently assume that such corporations are able to spend similar monetary amounts as malicious users to attack the network.

However, malicious corporations cannot be modeled as ε-good, as they wish to cause harm on the network through external incentives. We aim to partially protect our users from such malicious actors, through reputation systems (see page 54) and through self-enforcing contracts with positive margins in our nash equilibria (see page 75) that can decentivize such malicious actors. Our reputation systems which require proof-of-burn (see page 65) or similar sybil-resistant schemes help in making these attacks more costly. Webs-of-trust can fully protect careful users from such malicious actors, although great care is required from the user side.

Malicious governments

As the OpenBazaar software can be operated worldwide, malicious governments should be taken into account. Malicious governments may wish to bring the network down for censorship reasons or for legal reasons (Greenberg, Inside DarkMarket: a Silk Road the FBI can't touch, 2014).

The goals of such agents are similar to the goals of malicious corporations. In addition, a malicious government has the following goals:

- *Deanonymization*, unmasking the anonymity of an OpenBazaar user
- *Censorship*, blocking certain categories of products or individual products from being traded

Unmasking the anonymity of an OpenBazaar user is technically equivalent to making an association between the GUID of a user and any real-world identity-revealing information such as IP addresses (Greenberg, Creators of New Fed-Proof Bitcoin Marketplace Swear It's Not for Drugs, 2014).

A malicious government has similar resources as a malicious corporation.

Malicious governments can be categorized into *active* and *passive* based on their willingness to interfere with networks. A passive government is unwilling to manipulate data as a man-in-the-middle at the network level, and is only willing to be a passive eavesdropper. An active government is happy to interfere with network traffic by manipulating it.

A passive government has access to these additional resources:

- They can manipulate the legal framework of their country
- They can introduce new laws
- They can issue arbitrary subpoenas and warrants
- They can issue secret warrants and take decisions in secret courts (Zimmerman M. , 2013)

An active government has, in addition, access to the following resources:

- They can sybil-attack the issue of identification documents such as passports
- They can man-in-the-middle Internet connections within their country, up to but not defeating the Tor security assumptions (Johnson, Wacek, Jansen, Sherr, & Syverson, 2013)
- They can break Internet connectivity within their country (Chulov, 2012)
- They can issue arbitrary PKI certificates for HTTPS and TLS protocols (Stamm & Soghoian, 2012)

Malicious governments are the hardest attack to guard against. While our goal is to be able to defend against malicious governments, we are not able to do this currently. Our decentralization efforts are oriented around the idea that basic protection from malicious governments should be possible.

We aim to provide full protection against a passive malicious government, and partial protection against active malicious governments.

On the topic of denial-of-service attacks by breaking network connectivity completely, we do not have a mechanism of defence. We rely on the fact that countries will prefer to keep Internet connectivity for the most part of their users. Mesh networks (Johnson, Matthee, Sokoya, Mboweni, Makan, & Kotze, 2007) can be used to guard against such attacks.

Malicious developers

The last malicious actor is a malicious developer with commit access. These developers can be manipulated by one of the above actors through law (secret subpoenas) or bribery in order to achieve their end-goals. Therefore, the goals of a malicious developer are aligned with the goals of those above.

We prefer to list this malicious actor separately, as they have access to a different arsenal of attacks.

In particular, a malicious developer has access to the following resources:

- They can merge arbitrary pull requests, in effect writing arbitrary code

Our primary means of defence against malicious developers are secure development practices (see page 51).

Secure development model

To be able to meet our technical needs for the threat model in terms of malicious developers, we aimed to ensure that third-party users and potential auditors can easily verify the integrity of the software in multiple ways. This is part of our contribution through this work. To achieve this, we transitioned the DarkMarket project to a more secure development model in OpenBazaar through the various means described below.

Testing and automated build

We have introduced an automated build system to OpenBazaar. We used travis (Meyer, 2014) and introduced several unit tests on the foundations of the system to achieve some verifiability. This ensures that, during review, no basic properties of the system are violated. Such violations would be indicated by either a combined suspicious change in both the code and the tests, which we hope would be obvious, or an indication of a failing build, which we disallow by policy and is apparent to all developers and users in an obvious way.

The tests that run as part of our automated build include a coverage test with coveralls to ensure that we maintain good and improving statistics of code coverage (Poisot, 2015). We envision this will drive the developers of the project to continue developing in a testable way. Finally, we have included tests that verify the coding style of the code using pylint for static analysis of Python and jshint for static analysis of Javascript. Ensuring a healthy coding style makes the code more reviewable, auditable, and hence indirectly improves security. As part of this work, and with the important help of cryptography student Nikolaos Korasidis we contributed with several patches that visibly improve the coding style of the project.

Open source

We have successfully pursued a migration from the existing AGPL license of DarkMarket to the freer MIT license. Part of this legal work was to collect consent from the existing DarkWallet (unSYSTEM) team that built the first DarkMarket version before it was forked into OpenBazaar. While OpenBazaar has largerly rewritten most if not all of the old code, it was of philosophical and of tribute importance to us to consider the agreement of the original team. Furthermore, a lot of this work involved communicating to the community the need for this change, which we did through our blog and through the relevant subreddit, /r/openbazaar. The blogpost announcement is available in the appendix of this work. Based on the MIT-licensed bitcoin core example, we consider the transition to the MIT license an important step to support the security of OpenBazaar and to encourage the community to contribute with code and auditing.

Reviewing process

To ensure a more rigorous development and deployment process, we have, as part of our contributions in this work, pursued a transition from the cow-boy-style development of the original DarkMarket project into a more disciplined master/develop workflow.

Development now happens in the 'develop' branch and the 'master' branch is used only for releases. Security-wise, this plays an important role, as it allows for auditing the master branch more rigorously by third parties. Furthermore, this allows all commits on the master branch to be GPG-signed by the release maintainer.

We have chosen to adopt Vincent Driessen's master/develop model of development (Driessen, 2010). This requires all developers to maintain their own forks and every change in the develop branch passes through a code review of another developer. From a security point of view, this code review ensures that at least two developers see every line of code merged into the project, thus ensuring that a lone rogue developer cannot introduce malicious code to the marketplace.

Web-of-trust

One of our major contributions to the OpenBazaar ecosystem is its web-of-trust system.

The OpenBazaar web-of-trust maintains three important factors: First, it maintains strict pseudonymity through anonymizing mechanisms; second, it establishes true trust instead of identity verification; and third, it is completely decentralized.

The OpenBazaar web-of-trust is used in a commercial setting. Trust is used to mitigate commercial risk, which involves losing money. Traditional identity-verifying webs-of-trust such as GPG are in purpose agnostic about the trustworthiness of the web-of-trust participants. That type of web-of-trust verifies the identity of its members with varying certainty. However, it remains to the individual to associate trust with a particular person for a particular purpose: Whether they can be trusted with money, for example (Zimmerman, 1995).

In OpenBazaar, the participants are inherently pseudonymous. In this setting, we wish to maintain an identity for each node. This identity is strictly distinct from the operator's real-world identity. While the operator may choose to disclose the association of their real-world identity with their pseudonymous node identity, our network provides certain assurances that such pseudonymity will not be broken. Hence, pseudonymity is closely related to anonymity: Pseudonymity allows the creation and maintenance of an online "persona" identity with a certain history; anonymity ensures the real-world identity of the persona operator remains unassociated with the persona. Hence, our goal is to provide both pseudonymity and anonymity.

A true web-of-trust is a directed graph where nodes are individuals and edges signify trust relationships. In contrast to identity-verifying webs-of-trust, in a true web-of-trust, edges do not signify identity verification; they signify that the edge target can be trusted in a commercial setting transitively. For example, when trust is given to a vendor, it signifies that the vendor is trustworthy and will not scam buyers by not delivering goods (for stronger assurances against such scams through self-enforcing contracts see page 75). When given to a mediator, it signifies that the mediator is trustworthy and will resolve conflicts between transacting parties with a neutral point of view judgment. And when given to a buyer, it signifies that the buyer is trustworthy and will pay the money she owes. Clearly, trust is not symmetric.

While identity verification is a concept successfully leveraged by individuals for secure communications and other transactions, it is meaningless to try to verify the identity of a pseudonymous entity, because a pseudonymous entity is the cryptographic key in and of itself, and therefore identity verification would constitute a tautology. It is therefore required to adopt a different meaning of "trust" than in a traditional setting of real-world identities. Hence, trust in this case signifies the trustworthiness of an individual in a commercial setting, their financial dependability, their reliability as mediators, and their credibility as trust issuers.

A modified web-of-trust

Webs-of-trust historically have provided a setting for ensuring correct identity association with asymmetric cryptographic keys. Traditional webs-of-trust are not applicable to networks where anonymity is a desired benefit. We propose a pseudonymous web-of-trust where agents vote for the trust of others. By disclosing only partial topological information, anonymity is maintained. An inductive multiplicative property allows propagation of trust through the network without full disclosure. We introduce additional global trust measures (see page 62) to thwart Sybil attacks through an artificial cost of identity creation and maintenance and to bootstrap the network. The network is applied to a commercial setting to manage risk. Finally, we highlight certain attacks that can compromise the trust of the network under certain assumptions.

The aim of our web-of-trust is to construct a means to measure trustworthiness between individuals in a commercial setting where goods can be exchanged between agents. The goal of such a measure is to limit, as much as possible, the risk of traders in an online decentralized anonymous marketplace.

Risk management in OpenBazaar is based on two pillars: On one hand, Ricardian contracts are used to limit risk through mediators, surety bonds, and other means which can be encoded in contract format. On the other hand, the identity system is used to establish trust towards individuals.

Trust management in OpenBazaar is established through two different types of mutually supporting systems; projected trust and global trust. Projected trust is trust towards a particular individual which may be different for each user of the network; hence, trust is "projected" from a viewer onto a target. Global trust is trust towards a particular individual which is seen as the same from all members of the network. Projected trust is established through a pseudonymous partial knowledge web-of-trust, while global trust is established through proof-of-burn and proof-of-timelock mechanisms.

In this section, we describe projected trust, which is our basic trust system. In the next sections, we augment it with global trust and unbreachable contracts to construct the full system.

A naive solution

The most obvious and naive choice for a trust system in a commercial setting is a global voting system. While this solution is obviously flawed, it constitutes an important foundation upon which the next ideas are built. In this section we describe this idea in order to determine its flaws, which are solved in our next proposal.

In typical centralized systems, including eBay and Bitcoin OTC, there is a global rating which is determined as the sum of the individual ratings of others towards an individual. Attacks of fake ratings are taken care in the system in an ad hoc fashion which is enabled by the centralized nature of the system.

In decentralized systems, Sybil attacks (Douceur, 2002) make this situation particularly more challenging and without an easy way to resolve. Hence, this naive solution is undesirable. The solution to this problem is to first introduce a cost of identity creation and maintenance for global trust, and allow for the trust through edges to be projected. We explore the projection solution below, and we study the global trust later.

Partial topological knowledge

It is a conclusive result in the literature that, by analyzing graph relationships between several nodes, given certain associations between nodes and real identities, it becomes possible to deduce the real identities behind other nodes. In particular, if an attacker is given only global topological information about a web-of-trust as well as some pseudonymous identity associations with real-world identities, they can deduce the real-world identities of other nodes (Narayanan & Shmatikov, 2009). Hence, by revealing the complete topology of the web-of-trust graph between pseudonymous identities, an important loss of anonymity arises. For this reason, we propose a web-of-trust with partial topological knowledge for each node. Under this notion, every node only has knowledge of its direct graph neighbourhood – they are aware of the direct trust edges that begin from them and end in any target. This does not disclose any information about the total graph, as they are arbitrarily selected by each node. These ideas have been explored previously in the literature as friend-to-friend networks (Popescu, 2004).

Trust association

In the pseudonymous web-of-trust, each node indicates their trust towards other nodes that they understand are trustworthy. This understanding can come from real-life associations among friends who know the real identity of the pseudonymous entity. An indication of trust does not harm anonymity in this context. This understanding can also come from external recommendations about vendors using friendly names. For instance, under a threat model that trusts some centralized third parties, it is possible to establish trust in this manner. As an example, if a vendor is popular on eBay, and a buyer trusts eBay, the vendor can disclose their OpenBazaar user-friendly name on their eBay profile, and the buyer may opt to trust that identity directly.

Trust is indicated as edges with a source, a target, and a weight. The weight is a floating-point number from -1 to 1, inclusive, with 0 indicating a neutral opinion, -1 indicating complete distrust, and 1 indicating complete trust. These can be displayed in a graphical user interface in a more simplistic manner; for example, the canonical client may opt to present a star-based interface for positive trust, and flagging for negative trust; or, alternatively, it may choose to present a simple thumbs-up and thumbs-down option, indicating discrete trust of 1 or -1 if employed, or 0 if not used. These trust edges remain local and are not disclosed to third parties. We call the direct edges "direct trust" and the weights will be denoted as w(A, B) to indicate the direct trust from node A to node B.

Trust transitivity

As topological knowledge is partial, we resolve the projected trust between nodes through induction. Let t(A, B) denote the projected trust as seen by node A towards node B. Projected trust is then defined as follows:

$$t(A,B) = \begin{cases} w(A,B), \text{if } w(A,B) \text{is defined and } w(A,C) > 0 \\ \alpha \sum_{C \in N(A)} \frac{w(A,C)t(C,B)}{|N(A)|}, \text{otherwise} \end{cases}$$

Where:

- t(A, B) denotes the projected trust from A to B.
- w(A, B) denotes the direct trust from A to B.
- N(A) denotes the neighbourhood of A; the set of nodes to which A has direct trust edges.
- |N(A)| denotes the size of the neighbourhood of A.
- α is an attenuation factor which is constant throughout the network.

The meaning of the above equation is very simple: If Alice trusts Bob *directly*, then Alice's trust towards Bob is clear. If Alice does not trust Bob directly, then, since we wish to retain only partial topological knowledge, Alice can deduce how much to *indirectly* trust Bob from her friends. She asks her friends how much they trust Bob, directly or indirectly; the trust from each friend is then added together to produce the projected trust. However, the trust contributed by each friend is weighted based on the local direct trust towards them; if Alice trusts Charlie directly and Charlie trusts Bob (directly or indirectly) then the projected trust that Alice sees towards Bob is weighted based on her trust towards Charlie;

for example, if Alice trusts Charlie a little, he's not allowed to vouch for Bob a lot. Finally, the projected trust is normalized based on the number of friends one has, so that it remains within the range from -1 to 1.

The condition w(A, C) > 0 is imposed so that negatively trusted neighbours are unable to vouch for their trust towards others – otherwise they would be able to lie about their trust towards others if they knew they were negatively trusted and impose onto them the opposite trust from what they claim.

The attenuation factor α is used to attenuate trust as it propagates through the network. Thereby, nodes further away from the source gain less trust if more hops are traversed. We recommend a network-wide parameter of α = 0.4 as is used by Freenet (Freenet Web Of Trust), but this can be tweaked based on the network's needs.

This simple algorithm assumes that trust is transitive; if Alice trusts Charlie and Charlie trusts Bob, then Alice trusts Bob. This is a strong assumption and may not always hold in the real world. Nevertheless, we believe it constitutes a strong heuristic that allows a network to deduce trust with partial topological knowledge.

A comparison to other webs-of-trust

It seems worthy to compare this approach to existing webs-of-trust to point out their differences. In contrast to GPG (Zimmerman, 1995), our proposal maintains both anonymity and true trust. In contrast to Freenet (Freenet Web Of Trust), trust is used for commercial purposes and not just to fight spam. In contrast to Bitcoin OTC (Folkinshteyn) (Lee A. , 2012), the network is decentralized, the topology is only partially known, and the trust is only projected and not global.

Certain services provide webs-of-trust in a centralized way, but they do not advertise them as such. eBay ratings (Mui, Mohtashemi, Ang, & Szolovits, 2001) (Standifird, 2001) provide webs-of-trust, but the topology is globally known and the platform is centralized. Facebook's social graph (Ugander, Karrer, Backstrom, & Marlow, 2011) provides trust through friendships. Interestingly, Facebook's social graph allows partial topological knowledge depending on the privacy settings of the participants. However, centralization is still a problem. In centralized solutions, pseudonymous identities through user-friendly names can also be maintained without a blockchain. The simplicity of implementation benefit is also a strong one.

Centralized solutions have the drawback of a single-point-of-failure. As one of the goals of OpenBazaar is to avoid Achilles' heels (single points of failure), centralized solutions become unacceptable. As mentioned in the sections on decentralizatoin, the purpose is, for instance, to eliminate the ability for government intervention in the web-of-trust through secret warrants served to the administrators of such a centralized system that may require the handover of private encryption and signing keys.

Bootstrapping the web-of-trust

It is hard to establish trust targeting a new node on the network with no web-of-trust connections to it. This issue is addressed in the Global Trust section. However, it is easier to allow new users entering the network to trust individuals who have established some trust through the web-of-trust. Bootstrapping trust is widespread practice in the literature (Clarke, Sandberg, Wiley, & Hong, 2001).

This bootstrap can be achieved by including a hard-coded set of OpenBazaar node key fingerprints that are known to be good in the distribution. At the beginning, these can be the OpenBazaar developers. It is crucial that the number of nodes included is wide so that no individual can influence the bootstrapped trust. Advanced users are advised to further diminish the weights of the direct edges to the bootstrap-trusted nodes and to include higher-weighted edges to people they physically trust.

The special bootstrapping nodes must be configured to always respond to trust queries, regardless of whether they trust the query initiator.

This practice in particularly useful to guard against illicit uses of the network, which have been experienced in similar, centralized but anonymous solutions (Barratt, 2012). By ensuring the bootstrap-trusted nodes highly rate only non-illicit traders, it is expected that the network will promote legitimate uses of trade. While black market goods can still be traded, the user will be required to opt-in for such behavior by manually introducing trust to nodes who highly rate trusted black market vendors.

Graph separator attack

The web-of-trust security strongly depends on the size of graph separators. Let us examine the trust as seen from a node A to a node B. Let G' be the graph induced from the web-of-trust graph G as follows: Take all the acyclic paths from A to B in G whose every non-final edge is positive. These contain nodes and edges. Remove all nodes and edges that are not included in any such path to construct graph G'. Then G' is the induced trust graph from A to B. An (A, B) separator is a set of nodes of G' which, if removed from the graph, make the nodes A and B disconnected.

We define an entity "controlling" a set of graph nodes as being able to arbitrarily manipulate the reported t and w functions when the controlled nodes are inquired about their trust beliefs.

We will prove the following theorem for any separator. The smallest separator, mentioned below, is simply easier to manipulate by a malicious party, as they are required to control a smaller set of nodes.

Theorem: A malicious entity controlling the smallest (A, B) separator on the induced graph G' will be able to change a negative projected trust to a positive projected trust towards the target as seen on the original graph G.

Proof: Let S be any (A, B) separator on G'.

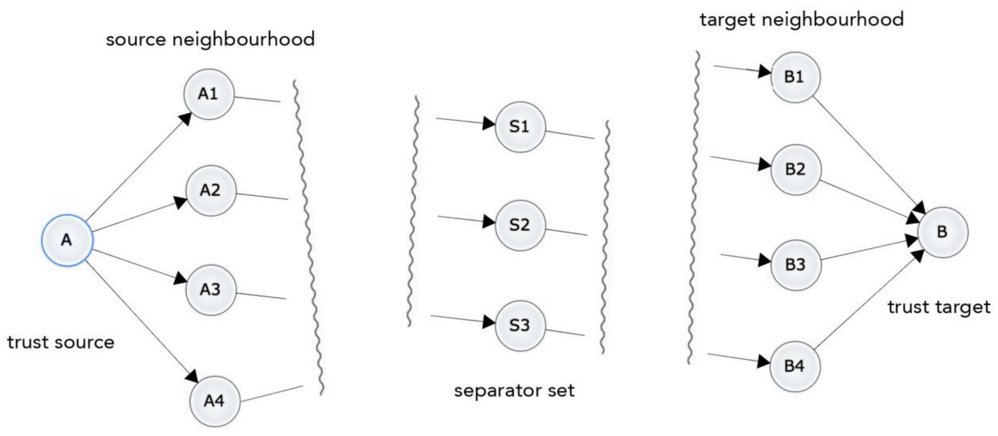

FIGURE 14: A GRAPH SEPARATOR ON THE WEB-OF-TRUST

Since the trust from A to B was originally negative, this means that A and B are connected on the induced graph G'. As the projected trust is negative, this means that there are direct negative edges in G' from some intermediate parties B1, B2, ..., Bn to B; these edges cannot be indirect, as indirect edges are not traversed for trust discovery.

We will show that the trust through any of Bi can be manipulated to be positive. Indeed, take some arbitrary Bi. Then there will exist some paths from A to B with Ci as their penultimate node. Take some arbitrary path P_i,j from A to B through Bi. We will show that this path can be manipulated to produce positive trust. Since S separates A and B, then there must exist some element s in S that is also in P_i,j. Since s is controlled by the malicious entity, they can modify their claimed direct trust towards B and set it to 1. They can also completely ignore any indirect trust, including Bi's opinion:

$$w(s, B) = 1$$

Through this mechanism, every path can be manipulated through the control of S. Therefore, all paths ending in a negative edge can be eliminated. Since A and B were originally connected, they will remain connected through this trust manipulation. Finally, at least one path ending in a positive edge will exist. Therefore, since t(A, B) will be a sum of positive terms, A's projected trust towards B will be bounded from below as follows:

$$t(A,B) \geq \alpha \, \frac{t(A,s)}{|N(s)|}$$

But t(A, s) must be necessarily positive. Therefore, t(A, B) must also be positive on the induced graph. However, projected trust on G is only based on results on the (A, B) induced graph G', and the value of t(A, B) will necessarily be the same as seen on G. ■

We have shown that a determined attacker can manipulate trust if they control some separator on the network. Therefore, it is crucial to avoid hub nodes in the network, and especially the existence of trust through only non-disjoint paths. As the web-of-trust develops, it is important to form trust relationships that connect nodes from regions with long distance.

This result is expected; a separator of size 1 is essentially an Achilles' heel for the system. Such single-points-of-failure necessarily centralize the web-of-trust architecture and completely undermine the decentralized design.

Separator attacks and more rigorous proofs on various models of similar broadcasting problem have been studied extensively in the literature (Pagourtzis, Panagiotakos, & Sakavalas, 2014).

Topology detection through queries

As one of the web-of-trust goals is to disallow malicious agents to learn the global network topology, it is crucial that the default node configuration is to not respond to trust queries initiated by nodes they do not trust. If this mechanism is not employed, a malicious entity can query all known network nodes and eventually deduce the global network topology easily. Using the disclosed topological information, the adversary can subsequently deanonymize the network (Narayanan & Shmatikov, 2009).

The exception to this is bootstrapping nodes, which, due to necessity, must be configured to answer all trust queries. Some ε-positive trust can be used as the minimum threshold of trust below which the canonical client does not respond to queries. To ensure queries are authenticated, imagine a query from A to C about whether B is trustworthy. The query must be signed with A's OpenBazaar private key to ensure topological information is not revealed to unauthorized parties asking for it. The query must be encrypted with C's OpenBazaar public key to ensure that nobody else can read it, even if the inquirer is authorised. Finally, the response must be signed with C's private key, to ensure that trust is not manipulated as it travels the network, and encrypted with A's public key, again to ensure topological confidentiality.

In this sense, most trust must necessarily be mutual. However, the topology of the graph still remains directed, as the trust weights can be different in either direction.

Global trust

In addition to the projected trust system provided by the web-of-trust, it is desired to provide some global trust in the network. This serves to allow the network to function with nodes that need to receive trust, but are not associates with other users of the network, or wish to remain pseudonymous even to their friends. In addition, this allows to bootstrap the network for someone who wishes to use it without explicitly trusting any entities.

Global trust mimics the trust given in real-world transactions towards vendors that have invested money in their business, something that can be physically verified. When a buyer visits a physical shop to purchase some item, they trust that the shop will still be there the next day in case their item is faulty; they do not expect the shop to disappear overnight[1]. The reason for this expectation is that it would clearly be unprofitable for the vendor to open and close shops every day, as it costs money to establish a store, and the money needed to establish a store is more than the money a vendor could gain by selling a faulty product.

Similarly, hotels ask for the passport of their residents (US Department of Homeland Security, 2009) in order to be able to legally hold them accountable to limit financial risk. While it is possible to create counterfeit passports every other day, it is economically irrational to do so, as building a new identity is more costly than a couple of hotel nights.

In these situations, rational agents are economically incentivised not to cheat on transactions through physical verification of proof that it would be costly to forfeit their identity – either to create a counterfeit passport or to shut down a physical store overnight. However, in a pseudonymous digital network, such mechanisms need to be established artificially. We start by exploring some approaches to the problem that do not meet our goal: Proof-of-donation and proof-to-miner. Subsequently, we introduce a mechanism that solves the problem, proof-of-burn. Alternative proposals that could solve the problem such as proof-of-timelock are also explored. These global trust mechanisms are then combined with projected trust to build a total trust score.

In all schemes, the person wishing to create trust for a pseudonymous node pays a particular amount of money, which can be provably associated with the node in question. The differentiation comes from whom the payment is addressed to.

[1] Interestingly, real-world shops *do* sometimes disappear overnight (Brabec, 1998)

Proof-of-donation

In a proof-of-donation (also called "proof-of-charity" in the community) scheme, the pseudonym owner pays any desired amount to some organization, which is hopefully used for philanthropic or other non-profit purposes. The addresses of organizations that are allowed to receive donations would be hard-coded in the canonical OpenBazaar client and payments towards them would have been verified by direct bitcoin blockchain inspection by each client. A proof-of-donation first seems desirable, as money is transferred to organizations for good. One possible scenario could include funding the OpenBazaar project itself through this scheme.

The technical way to achieve proof-of-donation is to simply make a regular bitcoin transaction with a donation target as its output. The transaction must also include the GUID of the target OpenBazaar identity that the donation is used for; the GUID for example could be included in an ε-valued proof-of-burn output (see next sections).

Nevertheless, a proof-of-donation scheme is inadequate for our purposes, as it introduces an Achilles' heel. In particular, if a malicious agent is able to access the private cryptographic keys of one of the donation targets, they are able to game the system and manipulate trust arbitrarily (Hearn, 2013). Compromising the private cryptographic keys can be achieved through various ways by powerful agents; for example, a secret warrant can be issued by a government ordering that the private keys are handed to the court.

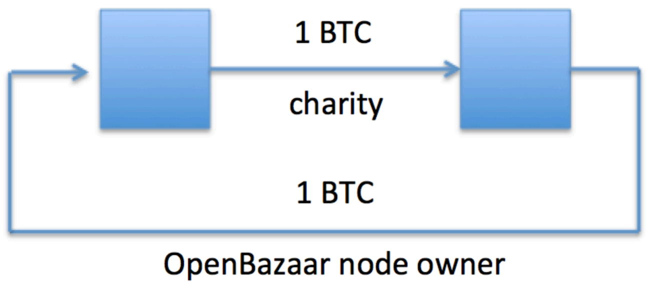

FIGURE 15: THE PROOF-OF-DONATION ATTACK FEEDBACK LOOP

This attack would work as follows: The malicious agent first generates a new OpenBazaar identity. Subsequently, they donate a small amount of bitcoin to the donation target organization they have compromised, including their OpenBazaar node as their target identity in the proof-of-donation, thereby gaining a certain amount of trust. They then use the private keys they control to give the money back to themselves, potentially through a certain number of intermediaries to avoid trackability. Finally, they repeat the process an arbitrary amount of times to gain any amount of trust desired, thereby gaming the system.

The attack is illustrated in Figure 15: The Proof-Of-Donation attack feedback loop. The figure is a slight abuse of notation; there are no feedback loops in the bitcoin graph, and in fact loops are impossible because of the hashing mechanism used to address transactions. We use the feedback notation to illustrate that the coins return to their original owner. In reality, the attack is a long chain of transactions where the owner remains the same entity controlling different cryptographic keys.

Proof-to-miner

In a proof-to-miner scheme, the payment to create trust for an identity is paid to the miner that first confirms the bitcoin transaction which is the proof. This scheme initially seems desirable, as there is no single entity which can be compromised, and it incentivizes the network to mine more, thereby making bitcoin more secure and, in turn, OpenBazaar more secure.

Technically, proof-to-miner can be achieved by including an OP-TRUE in the output script of the transaction as shown in Figure 16: A proof-to-miner transaction, where we use a "*" edge subscript to denote that anyone can spend it. This OP-TRUE can then be followed by code containing information about the node; for example, the code could contain a push of a constant giving the GUID of the OpenBazaar node followed by a pop. While anyone is, in principle, able to spend the output of the given transaction, a miner is incentivized to only include their own spending in their confirmation (Bitcoin Developers). That is, when a miner sees an anyone-can-spend transaction, they will include an immediate spending transaction in their attempted block that they are mining. Again it is important to include the GUID of the OpenBazaar identity in the transaction, otherwise the proof cannot be tied to a specific OpenBazaar node.

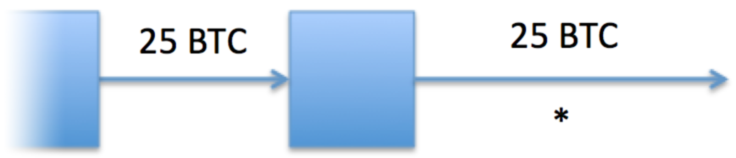

FIGURE 16: A PROOF-TO-MINER TRANSACTION

Unfortunately, it is again possible to game this system. The system is susceptible to a rogue miner attack. The rogue miner attack works as follows: The rogue miner first generates a new OpenBazaar identity. Subsequently, they make a proof-to-miner bitcoin transaction with any amount they desire, but they keep the transaction secret, without broadcasting it to the network. They then perform regular bitcoin mining as usual, but include their secret proof-to-miner transaction in their block confirmation attempt, without revealing the transaction to the network. Including an additional transaction does not increase the cost of mining; therefore this approach can be employed by existing rational miners. If they succeed in generating a block that contains their proof-to-miner transaction, they broadcast the secret transaction on the network together with their mined block and they gain identity trust in the OpenBazaar network, and can use the money again in the same scheme to increase their trust arbitrarily. If they do not succeed in generating a block, they keep the transaction secret and double-spend the money in a future transaction in the same scheme, until they are able to generate a block.

Using this method, a miner can accumulate arbitrarily large amounts of trust (Todd, Trusted identities through provable coin expenditures, 2012) for their identity, thereby breaking OpenBazaar security. However, a separation in the commit/donate steps of the proof-to-miner is in fact sufficient to overcome this problem (Todd, Purchasing fidelity bonds by provably throwing away bitcoins, 2013).

Proof-of-burn

Proof-of-burn schemes have been in use by the cryptocurrency community in various settings (CounterParty, 2014). In proof-of-burn, the payment to create trust for an identity is paid in a way that remains unspendable. Because it is unspendable, the system cannot be easily gamed as in the previous approaches.

Technically, proof-of-burn (P4Titan, 2014) makes a regular bitcoin transaction including an OP-RETURN in the output script of the transaction. Again, the GUID of the OpenBazaar identity is included in the transaction to enable blockchain validation.

Proof-of-burn makes Sybil attacks infeasible, as it requires the attacker to create multiple high trust entities in the network, which is costly. In essence, this is equivalent to bitcoin's proof-of-work scheme and leverages the existing blockchain for the proof.

Global trust based on proof-of-burn is based on how much money was burned to establish a particular identity. We use $g(x)$ to denote the global trust derived from the fact that an amount x has been spent to establish the trust of the identity. We notice that x is the sum of all the amounts that has been provably burned for this particular identity. In addition, we notice that when a particular transaction output is used to establish trust towards some identity, this output is necessarily only associated with one identity. Verification of this proof can be done by the canonical OpenBazaar client through direct bitcoin blockchain inspection. $x(B)$ is a function of the person whose proof-of-burn is to be determined, B. For simplicity, we will for now denote it as x.

To determine the numerical trust for the global trust associated with a particular identity, we work as follows. First, we calculate x as the sum of verified proof-of-burn amounts, in bitcoin, associated with the target identity.

Next, we use the following function to evaluate the trust towards the identity:

$$g(x) = 1 - \left(\frac{1}{2}\right)^{\frac{x}{c}}$$

Where x denotes the amount spent for the proof-of-burn g denotes the global trust associated with the identity, and c is the *base trust cost* of the system.

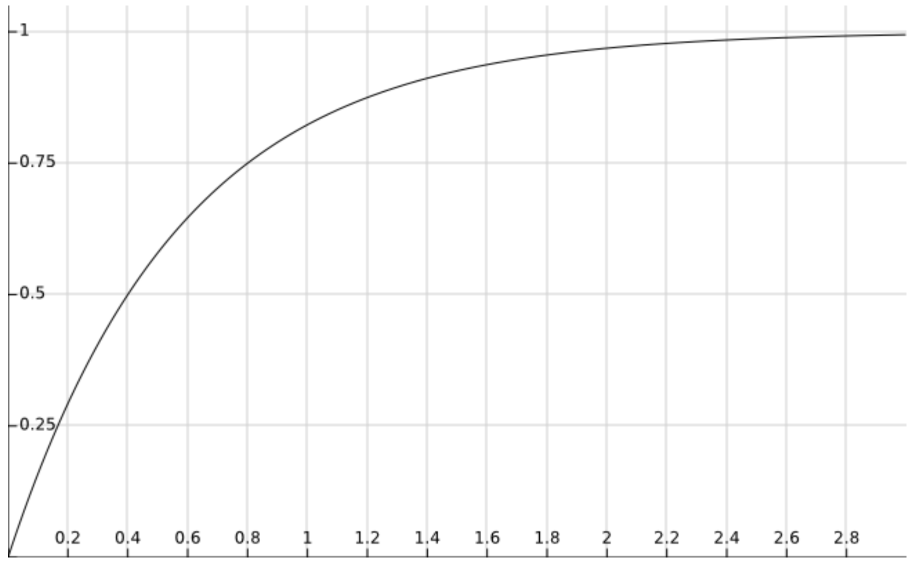

FIGURE 17: THE GLOBAL TRUST FUNCTION FOR C = 0.4

The *base trust cost* is a hard-coded value in the canonical client which is the amount of money required to establish basic trust in the system. The value can be determined based on the current exchange rate of bitcoin, and can be updated in the future depending on the network's needs. As the value of bitcoin is expected to rise, it is expected that the value for c will drop. This has the additional side benefit that historically older accounts accumulate more trust as time goes by, as long as the price of bitcoin rises.

To clarify the rationale of the above equation, note its following values, visualized in Figure 17: The global trust function for c = 0.4:

- g(0) = 0. A pseudonymous identity that has not provably burned coins has no global trust.
- g(c) = 1/2. A pseudonymous identity that has spent the base trust cost easily establishes a 50% global trust.
- $\lim_{x \to \infty} g(x) = 1$. Notice that it takes exponentially more money to approach 100% global trust.

We recommend that the base trust cost is a very small affordable amount for any human user. This will make the cost to enter the network small, but still avoid Sybil attacks. Such schemes have long been used in the literature as proof-of-work to avoid denial of service attacks (Back, 2002) (Naor & Dwork, 1992) (Juels & Brainard, 1999). Proof-of-burning has the same benefits as proof-of-working, as it delegates the proof-of-work to the bitcoin blockchain.

Almost-collision coin burning

The de facto standard for burning coin in bitcoin is through an OP-RETURN script (Bitcoin Foundation, 2013). This script has the important advantage that it contributes to bitcoin's network scalability, as it allows full nodes to prune their UTXO when proof of burn transactions are detected. The mechanism employed to achieve that is simple: While a UTXO is maintained for all unspent regular transactions, when an OP-RETURN transaction is received by a full node, the full node can avoid adding that transaction to the UTXO completely, as the OP-RETURN script constitutes a proof that the amount remains unspendable and hence no future transaction can attach this dangling output to its input; it is hence a permanent dangling output edge.

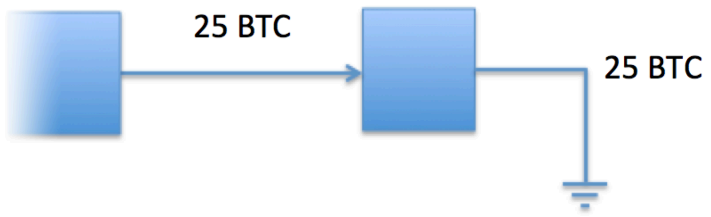

FIGURE 18: A PROOF-OF-BURN TRANSACTION

OP-RETURN scripts work by having the first operator of the bitcoin script be an OP-RETURN, indicating an immediate exception in the execution of the script, hence making spending impossible. After the initial OP-RETURN operator, the rest of the script data can contain information about why the coin was burned, so that different applications can demand different burning, and so that the association with an account is possible. For example, in OpenBazaar's case, it is important to associate the burned amount with an OpenBazaar GUID, which can be included as non-executable code after the OP-RETURN. The fact that this code is non-executable follows from the fact that it will never be executed due to the earlier exception. Figure 18: A Proof-Of-Burn transaction shows a visualization of a proof-of-burn transaction; we use the ground symbol to indicate an unspendable output edge from a transaction. In this example, 25 BTC are burned to establish an identity.

However, the OP-RETURN approach lacks certain usability properties that we wished to preserve in our OpenBazaar implementation. In particular, for simplicity of implementation and usage, as well as for separation of concern reasons, we decided that OpenBazaar does not need to include a bitcoin wallet implementation. Instead, the user can use any existing wallet software they wish. Hence, to make payments required by OpenBazaar, either for product purchases or for burn transactions, the user would have to utilize their wallet directly.

Today, wallets do not have the ability to create OP-RETURN scripts in any usable way. The only way to create burn transactions are through manual issuing of script commands by the user, which can be confusing or impossible to execute for an average user without a programming background. Furthermore, the OP-RETURN script must be associated with an OpenBazaar GUID, something that makes the inclusion of this ability in existing wallets harder.

For these reasons, we designed an alternative mechanism for coin burning which uses simple standard pay-to-pubkey-hash transactions. Furthermore, it is easy for regular wallets to create such transactions, and users can easily understand the process and make the payment without worrying that an unnecessary amount of money will be transferred and without requiring special programming knowledge.

Our schema for burning is based on the following cryptographic assumption, a resistance to an almost-collision: It is computationally infeasible to calculate two hash pre-image values x1, x2 such that:

$$\left\| H(x1) - H(x1) \right\| < \varepsilon$$

Where the norm denotes the Hamming distance of two strings and ε is a small constant, in our case 1. This assumption is strongly supported by the fact that a hash function is cryptographically secure; if this equation did not hold, a collision would have been found, modulo one bit, which indicates the hash is broken up to almost all of its bits.

Under this assumption for H = RIPEMD, our schema asks for the burner to take the ECDSA public key associated with their OpenBazaar identity and turn it into a bitcoin address by following the regular schema for 1-prefixed bitcoin addresses. Regular bitcoin addresses are generated from regular bitcoin ECDSA keys as shown in Figure 19: The standard bitcoin address generation algorithm. In comparison, the very similar provably unspendable address generation process is shown in Figure 20: OpenBazaar provably unspendable address generation.

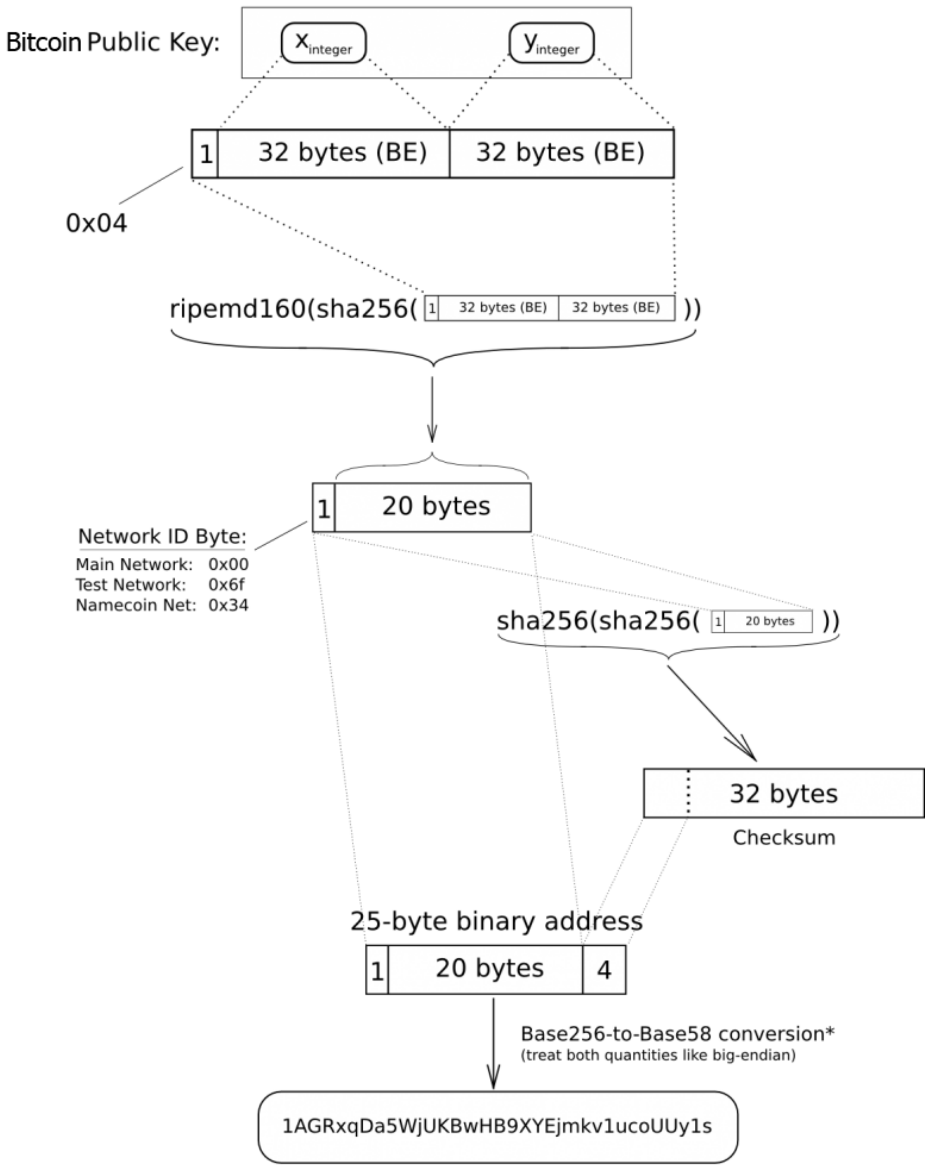

FIGURE 19: THE STANDARD BITCOIN ADDRESS GENERATION ALGORITHM (COURTESY ETOTHEIPI)

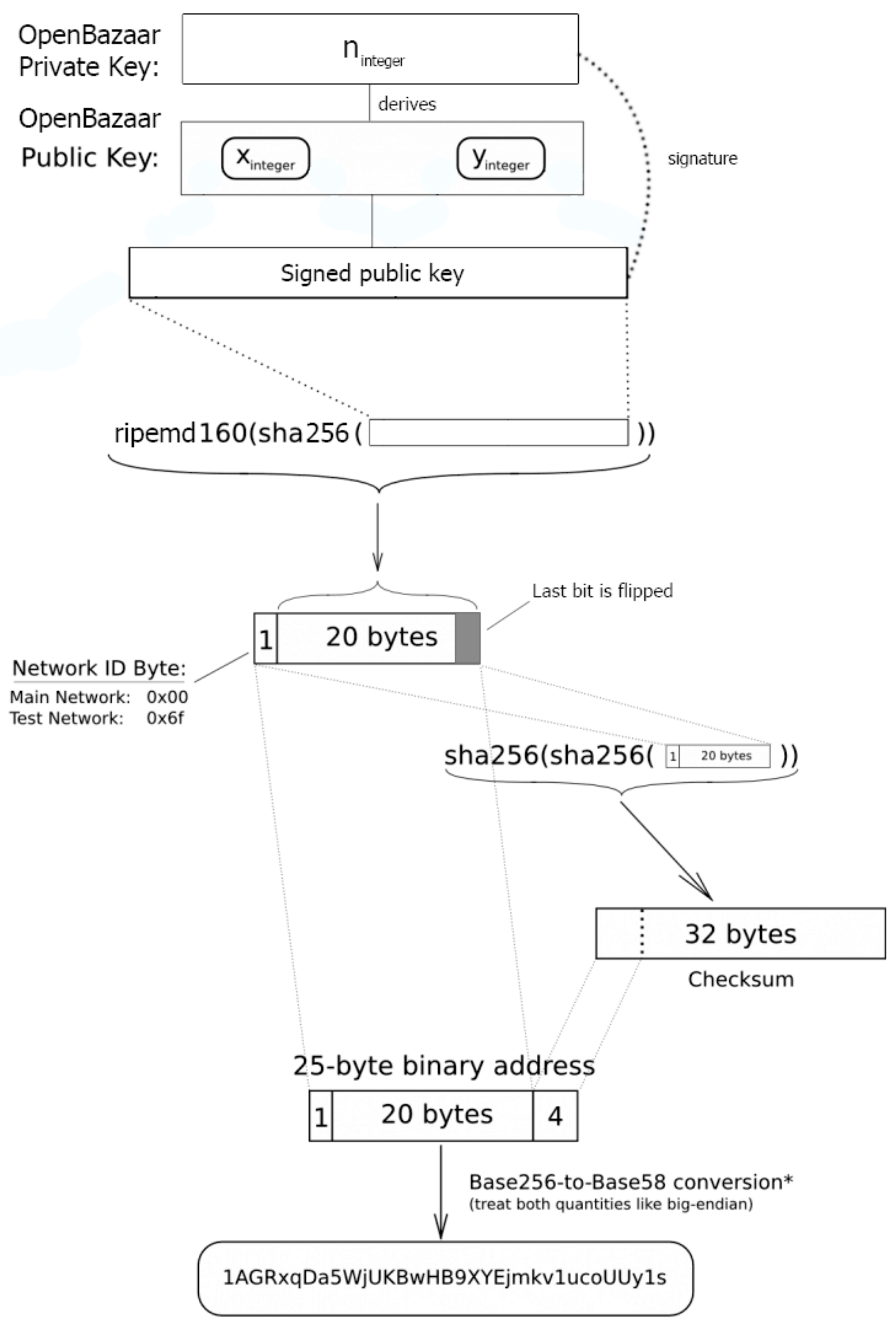

FIGURE 20: OPENBAZAAR PROVABLY UNSPENDABLE ADDRESS GENERATION

To generate an address that is provably unspendable, the burner starts with their ECDSA OpenBazaar public key and applies a similar process to generate a RIPEMD hash from the signed public key. However, the burner perturbates the RIPEMD hash result before base58-encoding it. Specifically, they flip the last bit of the hash output. The rest of the process follows identically. Finally, the burner transfers the amount of coin they wish to burn to this generated address. Our actual implementation of this process in OpenBazaar is shown in Listing 3: The OpenBazaar Proof-of-Burn source code, where we use the obelisk bitcoin library for base58 encoding and bitcoin address checksum generation.

```
def burnaddr_from_guid(guid_hex):
    _LOG.debug("burnaddr_from_guid: %s", guid_hex)

    prefix = '6f' if TESTNET else '00'
    guid_full_hex = prefix + guid_hex
    _LOG.debug("GUID address on bitcoin net: %s", guid_full_hex)

    guid_full = guid_full_hex.decode('hex')
    guid_prt = guid_full[:-1] + chr(ord(guid_full[-1]) ^ 1)
    addr_prt = obelisk.bitcoin.EncodeBase58Check(guid_prt)
    _LOG.debug("Proof-of-burn address: %s", addr_prt)

    return addr_prt
```

LISTING 3: THE OPENBAZAAR PROOF-OF-BURN SOURCE CODE

We will now illustrate the properties of correctness, uniqueness, and security for this scheme.

Correctness. To verify the correctness of the burn, a third party performs the same transformation as the burner. They begin from the public ECDSA key of the OpenBazaar node whose trust they wish to verify and follow the bitcoin address generation process, applying the same perturbation as the burner after the RIPEMD stage. Arriving at the final bitcoin address, the verifier then checks the blockchain for money that was sent to this address. This concludes that the burn an honest burner performs will be correctly verified by an honest verifier.

Uniqueness. Under the assumption that RIPEMD160 is a cryptographically secure hash function, assumptions already made by bitcoin, the uniqueness of burn address for each OpenBazaar key follows directly.

Security. For this scheme to be secure, we must prove that the burned money cannot actually be spent by anyone. Indeed, if the money were spendable, the spender would have to know the private key associated with a public key which hashes to the perturbated

RIPEMD160 value. However, this would allow the generation of an almost-collision in RIPEMD160, as the public key that can be used for spending the burned money and the public key of the OpenBazaar identity would constitute pre-images of hashes that only differ by one bit. From the almost-collision resistance assumption, we conclude that this is computationally infeasible.

The almost-collision method of coin burning introduces scalability challenges for the bitcoin software. We wish to make two remarks in regards to these scalability issues. First, we feel a failure for bitcoin to scale given a potential massive motivated community (or others) use of our primitive constitutes a security problem for bitcoin itself, which must be addressed without requiring players to behave fairly to the system. This could be a problem for bitcoin. If bitcoin is susceptible to such denial-of-service attacks, the use of bitcoin as a payment system must be reconsidered.

Second, because we support the bitcoin ecosystem and wish to provide suggestions for solving its scalability issues, these transactions can in fact be eliminated if proof-of-burn transactions are accompanied by the pre-image before perturbation. The accompanying pre-image constitutes proof that the money is unspendable, similar to the way OP-RETURN scripts constitute proof of unspendability. As these pre-images will be publicly available on the OpenBazaar network, in case OpenBazaar becomes largerly adopted, full bitcoin nodes can utilize the OpenBazaar network to detect prunable UTXO outputs which perform proof-of-burn through almost-collision pay-to-pubkey-hash scripts.

Regardless, the optimizability of the payment network is of little concern to its financially motivated users and its technical implementation details remain an open research problem.

However, since OpenBazaar has since switched to using SHA512[1] for hashing GUIDs and because we wish to support the scalability of bitcoin, we recommend that a migration to OP-RETURN is done when usability issues permit.

Proof-of-timelock

While proof-of-burn is equivalent to proof-of-work, we propose an additional mechanism that can be used separately or in combination with proof-of-burn. In proof-of-timelock, the proof-of-stake ability of a blockchain is leveraged to produce a system that eliminates Sybil attacks without having to resort to the destruction of money or, equivalently, CPU power.

[1] Note that the switch to SHA512 invalidates the security proof, since there is no cryptographic statement regarding the cross-collisions between SHA512 and RIPEMD160.

In proof-of-timelock, the individual interested in establishing trust towards a pseudonymous identity provably locks a specific amount of money in a transaction that gives the money back to them. This transaction has the property that it remains unexecuted for a specific predefined amount of time. However, the fact that the transaction is going to take place in the future, the exact amount, and the amount of time of the lock are publicly verifiable.

While proof-of-burn allows identities to be created in a way that is costly to recreate, proof-of-timelock ensures it is impossible that an enormous amount of identities associated with one real-world individual can co-exist at a specific moment in time. Proof-of-timelock is a weaker insurance than proof-of-burn; proof-of-burn can be thought as proof-of-timelock, but for an infinite amount of time.

We predict that people will feel considerably more comfortable ensuring their identities through proof-of-timelock rather than proof of burn. The psychological burden associated with money destruction may not be an easy one to overcome.

A Turing-complete blockchain such as Ethereum (Wood, 2014) allows an implementation of this mechanism. Nevertheless, we are reluctant in using this scheme, as Turing-complete blockchains are yet to be proven feasible in practice and may pose problems in terms of scalability, performance, and fees. Another mechanism that could be used for this scheme is Bitcoin's OP_CHECKLOCKTIMEVERIFY operator (Todd, OP_CHECKLOCKTIMEVERIFY, 2014).

As such, we recommend proof-of-timelock as an alternative mechanism, but further research is needed to conclude whether it is feasible as an underlying mechanism for a decentralized anonymous market. It is worthy noting that timelock-based mechanisms have been studied in theoretical cryptography in several settings (Rivest, Shamir, & Wagner, 1996), but the practical applications, while intriguing, remain limited in practice as far as implementation is concerned.

Total trust

Based on the projected and global trust metrics presented above, we propose the following measure as the total trust towards a network node:

$$s(A,B) = \frac{1}{2}\left(t(A,B) + g\big(x(B)\big)\right)$$

The projected and global trusts are added together to produce the total trust as seen from A to B. The weights used here are 50% for each, but it is advised to tweak these weights based on empirical evidence during development. For advanced users, the weights can be customizable.

The total trust can then be displayed in the user interface of the node of user A when she is viewing the profile of user B. Additional interface elements that are possible to include can be the exact amount of money spent in the proof-of-burn scheme, as well as the direct links yielding the cumulative projected trust for the particular target through induction.

Unbreachable contracts

Ricardian contracts

OpenBazaar uses the notion of Ricardian contracts. Ricardian contracts are contracts that are both human-readable and machine-readable. They describe a trade in detail, including the product being sold, its description, the hash of images of the product, the price and dates, and other information. Ricardian contracts were invented by Ian Grigg (Grigg, 2004) and their application to OpenBazaar was invented by Washington Sanchez (Sanchez, 2014).

Ricardian contracts are interesting because they can stand in traditional court systems if disputes are filed. This is important as we transition from the traditional legal system to decentralized legal systems. When this transition is fully completed, Ricardian contracts can be modified to be only machine-readable. In fact, as explained below, Ricardian contracts will become unnecessary as soon as everything in a contract is unbreachable. Unbreachability can be achieved for several ideas such as the promise of payment or the promise of product delivery as we illustrate below. However, some disputes still require human arbitration. Examples include “product not as advertised” or “product arrived damaged” issues. This is the reason why Ricardian contracts contain all the details of a trade so that disputes can later be resolved.

In the next sections, we explore 2-of-2 and 2-of-3 trades. In the 2-of-2 trade, the inclusion of a Ricardian contract is only useful if the real identity of participants can become known. In this case, traditional court systems can be used to resolve disputes. In 2-of-3 trades, an arbiter is introduced, and hence the necessity for a traditional court is not necessary, nor is it necessary to reveal the true identity of the participants. In cipherpunk settings such as the bitcoin community, the property of maintaining pseudonymity even in the case of dispute is valuable.

In the sections below, we will not describe the technical details of the file format of Ricardian contracts. However, a few words about the structure of Ricardian contracts are necessary to understand how trades can work. The Ricardian contract in OpenBazaar is a human-readable JSON file initially created by a merchant. The merchant includes the details of the product in the original JSON file, termed the “offer”, such as the product title, description, hash of images, price for the product, availability, and shipping options. An example is shown in Table 1: An example ricardian contract. This offer contract is then signed by the merchant’s ECDSA key to verify its authenticity.

TABLE 1: AN EXAMPLE RICARDIAN CONTRACT

```
{"order": {
  "id": {
    "contact": {
      "email": "dionyziz@gmail.com"
    },
    "guid": "6c80b332c81a880c1c0e06982d4ae94ac00e0bd5",
    "handle": "dionyziz",
    "onename": "dionyziz",
    "pubkeys": {
      "bitcoin": "1vXhpZpeDWLmp7vN7k52x3WwRSZK3DT6X",
      "pgp": "45DC00AEFDDF5D5CB988EC862DA450F3AFB046C7"
    },
    "role": "customer"
  },
  "metadata": {
    "category": "service",
    "category_sub": "invitation to tender",
    "expiry": "3 hours"
  },
  "order": {
    "comments": "Deliver to my co-worker before she gets to work.",
    "date": "2015-07-04 09:00:00",
    "delivery_address": "900 W Eddy St, Chicago, IL 60613, US",
    "item": "Chocolate cupcake box",
    "pickup_address": "1060 W Addison St, Chicago, IL 60613, US",
    "quantity": 1,
    "type": "delivery"
  }
 },
 "signatures": {
   "bitcoin": "HEfKMskTI7tfKEMRF36AxYaQ1OtUsd...",
   "pgp": "iQIcBAEBCAAGBQJWWlggAAoJELNinf0..."
 }
}
```

When a user browses the products of a merchant, they receive these signed offers and the OpenBazaar client parses them to display the products in question. When the user wishes to purchase a product, the amend the contract with their own details, such as their shipping address and amount, sign it, and send it to either the merchant directly (in case of 2-of-2 trades) or the arbiter (in case of 2-of-3 trades). In case of an arbiter, the arbiter signs the contract including their own details, such as what they are exactly willing to arbitrate, and sends it to the merchant to fulfill the order.

Contracts are very general. They can be about physical goods, digital goods, services (as seen in the example above), lending, gambling, financial agreements, or decentralized replacement markets for AirBNB and Uber (La'Zooz Developers, 2015).

Game theoretic primitives

In the next sections, a "fair Nash equilibrium" is a Nash equilibrium in which the purpose of the game, as designed by the game designer, is achieved; in our case, a trade is completed successfully and both parties are satisfied (i.e. the merchant receives payment and the buyer receives the product).

For the following theorems, we are dealing with rational players that will always play to maximize their utility; in cases of markets, their financial profit. We introduce the notion of *ε-good rational players*. These players prefer a strategy which is arbitrarily designated as the *status quo* strategy at the definition of the game over other alternative strategies, provided the alternative strategies do not have better utility. In particular, when multiple strategies maximize the utility function for the player in question, the player will select the status quo strategy.

This preference can be modelled by introducing some arbitrarily small positive constant ε and modifying the utility function to be more favourable by ε for the status quo strategy. Then, we will say that ε-good rational players will prefer a strategy S if there exists some $\delta > 0$ such that for all ε for which $0 < \varepsilon < \delta$, if the status quo strategy is modified to include a favourable adjustment ε to the utility function, then a classically rational player will choose strategy S.

These strategies are selected to correspond to behavior which is considered socially acceptable; in particular, we choose the status quo strategies to be those strategies in which a player lets the other player win if they do not have anything to lose. These are a good model for real human behavior, as people tend to play fair when there is no reason to play unfair. Specifically, for markets we will designate the status quo strategy to be the strategy in which the buyer receives their purchased product and the seller receives their payment.

This assumption allows us to argue that the status quo games constitute *unique* Nash equilibria under the ε-goodness assumption. Without the ε-goodness assumption, while status quo strategies are Nash equilibrium strategies, they are not the only ones, and hence the assumption is needed to explain the behavior of players who choose a "fair" strategy when there's nothing to lose. While the ε-good notion models our real-world intuition for player behavior, this formalization does not say anything more than this: When we require ε-goodness, our Nash equilibrium is not unique; if we do not require it and ε-bad players can be tolerated, our Nash equilibrium is unique.

The idea of ε-goodness differs from the traditional game theoretic notion of an ε-Equilibrium. In our considerations, the status quo solutions are true equilibria; ε-goodness is only required for proof of uniqueness.

In the following theorems, we will show that closed systems utilizing 2-of-2 and 2-of-3 multisig transactions between 2 and 3 ε-good rational agents respectively are unique Nash equilibria in which nobody is incentivized to steal from the other. These situations are desired for a decentralized trading system, as they indicate stability and ensure trust towards the system is provided to users.

In our game-theoretic modeling of the marketplace, we treat a trade as a game between rational agents who both gain some utility value from successfully completing a trade. This utility can be arbitrarily small, but is a non-negative value, and so players are incentivized to trade; we will denote this by λ. In the edge-case of the utility function being valued at zero, players are not incentivized to trade, and so will prefer not to enter in a game. This λ parameter is the value gained from trading: The merchant gains because they sell the product for more than what they spend to build it; and the buyer gains because they see utility in the product they purchase.

2-of-2 trades

The simplest form of self-enforcing contracts in trades can be achieved through 2-of-2 multisig trades (Spilman, 2013). In this work, we provide a game-theoretic analysis of the scheme and compare it with alternative suggested schemes.

The 2-of-2 trading scheme is the simplest self-enforcing contract for trades. It works as follows. A buyer, Alice, wishes to buy a product P from a vendor, Bob, for a price v. To do this, Alice creates a transaction with a single input of value v.

The transaction has a single output of value v. The output is a 2-of-2 multisig script, i.e. it requires a threshold value of 2 keys to sign the output for it to be released. These two keys are Alice's key and Bob's key.

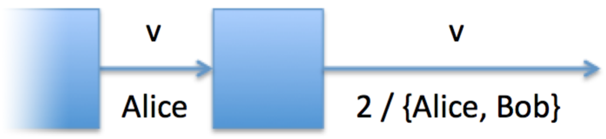

FIGURE 21: THE 2-OF-2 TRADE IN THE COMMIT STEP

The trade then works as follows: In the commit step, Alice creates the 2-of-2 transaction and pays a value v into it. Alice then publishes this transaction to the blockchain and provides it as a proof-of-payment to Bob as seen in Figure 21: The 2-of-2 trade in the commit step; at this stage, the 2-of-2 multisig output is unspent. In the verification step, Bob verifies that the proof-of-payment is valid by examining the transaction on the blockchain. He ensures that the output script is a 2-of-2 multisig script, that he hasn't seen this proof-of-payment before, and that one of his own public keys is included in the multisig. Bob then ships the product P to Alice. In the finalization step, once Alice receives the product, she signs off the 2-of-2 multisig script to an output address owned by Bob. Bob then adds his own signature to the 2-of-2 multisig script that sends the money to him, and thereby receives the money. This concludes the correct execution of the protocol as seen in Figure 22: The 2-of-2 Trade after finalization.

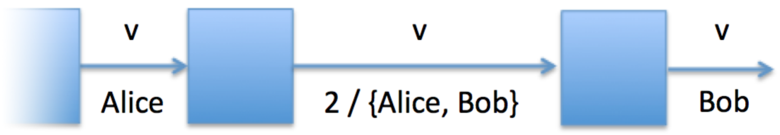

FIGURE 22: THE 2-OF-2 TRADE AFTER FINALIZATION

If Alice wishes to play unfairly, she would want to receive the product without paying for it. In this rogue scenario, if Alice tries to convince Bob to send the product without properly creating a 2-of-2 multisig transaction, Bob will be able to detect this and will not send out the product. On the other hand, if Alice creates a valid 2-of-2 transaction, her money is committed to buying the product and she's unable to double-spend it once it enters the public blockchain. Therefore, the buyer cannot receive the product without paying.

If Bob wishes to play unfairly, he would want to receive the money without sending out the product. In this rogue scenario, Bob would wait for the 2-of-2 transaction. Bob would then desire to receive the money without sending the product. However, the 2-of-2 transaction binds the money until Alice signs-off the finalization. Unless Alice finalizes early, Bob will not get paid until the product is delivered. Therefore, the seller cannot receive payment without delivery.

However, this does not ensure that an honest agent will receive compensation or the product. Indeed, on one hand, if Alice is an honest buyer, she may lose money in this game. This would work as follows. If Bob the vendor puts up a fake product for sale, Alice can be tricked in locking up her money in a 2-of-2 multisig transaction which remains stale forever, thereby burning her coins without delivery. While this scenario is possible in the 2-of-2 scheme, there is no incentive for Bob to play the rogue strategy. As such, the fair strategy constitutes a Nash equilibrium, but not the only Nash equilibrium: An alternative Nash equilibrium allows Bob to advertise non-existent products without delivering them. However, this does not provide any financial gain for Bob.

On the other hand, an honest vendor is not ensured to receive payment for their services either. This would work as follows. If Alice locks her funds in a 2-of-2 multisig and provides it as proof-of-payment to Bob, Bob can deliver the product. However, nothing ensures Bob that Alice will finalize the transaction to release the funds to Bob. Alice can simply walk away with her product and leave the money hanging. While this scenario is possible in the 2-of-2 scheme, there is no incentive for Alice to play the rogue strategy. As such, the fair strategy again constitutes a Nash equilibrium, but not the only Nash equilibrium: The alternative Nash equilibrium allows Alice to keep the money locked up. However, this does not provide any financial gain for Alice.

In our assumption of ε-good agents who receive some ε positive utility from playing the status quo strategy, this game is an acceptable way to trade goods. In practice, trust in the system can be improved by making the default strategy the status quo strategy in the implementation, making it easier for the user to play the status quo strategy and difficult for them to play the unfair strategies. For example, finalization can occur automatically after two weeks, unless the buyer indicates that they have not received the product.

TABLE 2: THE 2-OF-2 TRADE GAME

Buyer \ Seller	No game	**Doesn't ship**	**Ships**
No game	v, v		
Doesn't finalize		0, v	v + λ, 0
Finalizes		0, 2v + λ	v + λ, v + λ

The game is summarized in Table 2: The 2-of-2 trade game. The utility values represent the total amount of assets owned by the players after the trade (money + product value); initially, Alice starts with v assets, the money and Bob starts with v assets, the product. The green strategy is the status quo strategy; after trading, the value held by both parties is v each. The red strategy represents a rogue seller who doesn't have a product for sale; for this player, it is equally preferable not to play. The blue strategy represents a rogue buyer who doesn't finalize; for this player equally preferable to finalize. The orange strategy is cannot occur under the rules of the game.

However, this game is clearly unsatisfactory and unsettling for large financial transactions, especially when real-world players who want to break trust to the system are introduced. In later scenarios, we propose games in which the only Nash equilibrium is the fair strategy, even without requiring ε-goodness from agents. Ideally, we would wish that dishonest players would be punished financially, which we are able to achieve with MAD trades. Nevertheless, we first explore a simpler scheme, that of 2-of-3 trades.

Another issue with the 2-of-2 trade is that, from a game theoretic point of view, the default in the payment finalization could be manipulated by the buyer in what we call the "terrorist negotiation attack". Indeed, assume a buyer and seller in a one-shot game of trade. The buyer transfers the money to a multisig account which they provide as proof-of-payment. Subsequently, the seller sends out the product. Upon receiving the product, the buyer reveals to the seller that they do not wish to finalize the transaction. A rational seller and buyer could then come to a mutual agreement for any finalization. In particular, a rational seller would agree to a finalization where any small positive ε value is transferred to them with the rest of the money going to the buyer. On the other hand, a rational buyer would also agree to a finalization where they receive any small positive ε value. Under such circumstances, it is clearly not a rational strategy for the buyer to finalize the transaction, but they would prefer to work out a deal. This is clear from the orange square in the game theory strategy matrix above.

However, human nature allows us to assume that such "terrorist negotiation" deals will not be accepted. As long as the party at stake does not agree to buy into such an unfair deal, there is no incentive for a rational agent to play unfairly.

2-of-3 trades

The 2-of-2 trade game described above is satisfactory under the ε-goodness assumption. However, it undermines trust in the system and is worthy of improvement.

A first improvement in the 2-of-2 trade is the introduction of an arbiter. While the arbiter introduction constitutes a trusted third party, it is significant that we allow the trading parties to choose any arbiter they wish. Hence, there is no central authority that controls the arbitration in all transactions. This differentiates the OpenBazaar arbitration model from arbitration models developed in centralized bitcoin marketplaces such as Evolution where the central market took the role of the arbiter (OpenBazaar Team, 2015).

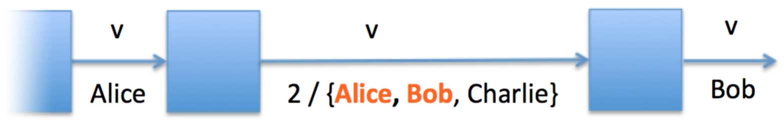

FIGURE 23: A CORRECTLY COMPLETED 2-OF-3 TRADE AFTER FINALIZATION

In the 2-of-3 trade, we work similarly to the 2-of-2 trade model, except that the multisig is a 2-of-3 multisig where a third party, the arbiter, is introduced. When Alice and Bob play as expected, the stages of the game are the same as in the 2-of-2 trade protocol and the transaction graph after finalization is shown in Figure 23: A correctly completed 2-of-3 Trade after finalization. The signing parties in the spent multisig output are highlighted in orange.

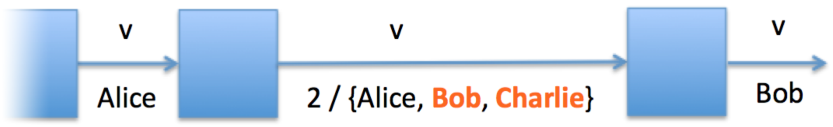

FIGURE 24: DISPUTE RESOLUTION THROUGH 2-OF-3 TRADE ARBITRATION WITH FORCED FINALIZATION

In case of dispute, the arbiter is allowed to make a decision. In particular, the arbiter is allowed to finalize the transaction by signing the transaction together with the vendor as seen in Figure 24: Dispute resolution through 2-of-3 Trade arbitration with forced finalization. In this case we say that the finalization step is *forced*. Dispute resolution can also result in a reversal of the trade where the arbiter is given the ability to reverse the transaction by signing the transaction together with the buyer as seen in Figure 25: Dispute resolution through 2-of-3 Trade arbitration with reversal. In both cases, the arbiter's decision can be made through proceedings similar to traditional court rulings, depending on the particular arbiter policies that both parties pre-agree on. Such procedures could involve the presentation of evidence by the transacting parties. For example, the buyer could present evidence that the product arrived damaged.

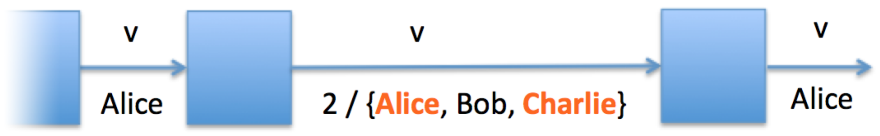

FIGURE 25: DISPUTE RESOLUTION THROUGH 2-OF-3 TRADE ARBITRATION WITH REVERSAL

Under normal circumstances, however, the arbiter does not involve themselves in the trade.

The 2-of-3 trade is unsatisfactory because of the third-party trust requirement. The arbiter is not enforced to act as they advertise and could in fact collude with the buyer or the seller to provide an unfair outcome. In particular, the danger is for vendors who own both a vendor and an arbiter account in a way that appears independent. These vendors can then, for example, easily choose to default on their delivery and still get paid. In a similar manner, large-scale buyers who control both a buyer account and in addition an arbiter account can easily get away by receiving products without proper payment. While it could be argued that the reputation of the arbiter is at stake, the problem of decentralized anonymous reputation has not yet been solved, as is clear by the current work. Some of these problems may be partially mitigated using timelocks (Todd, OP_CHECKLOCKTIMEVERIFY, 2014).

Because of their simplicity and usability, 2-of-3 multisig trades have been implemented in OpenBazaar.

MAD trades

The 2-of-2 and 2-of-3 trades are both unsatisfactory for different reasons each. The 2-of-2 trade is vulnerable to ε-bad agents who wish to harm the trust in the system. The 2-of-3 trade is vulnerable because trust in third party arbiters is arbitrary and collusion can occur.

To resolve this, the MAD (mutually assured destruction) model was proposed (Yoo). While MAD has been presented previously in the literature through third-parties or altcoins, we introduce a simple schema to achieve MAD on the bitcoin blockchain.

In the MAD model, Alice wishes to purchase a product P of value v from Bob. Alice and Bob then agree on a security parameter for each of the agents, α for Alice and β for Bob, two values in bitcoin. While for the game theoretic proof it suffices to set $\alpha = \beta = \varepsilon$ for any positive constant ε, for simplicity we can set $\alpha = \beta = v$.

The trade transaction has two inputs: One from Alice and one from Bob. Alice's input contains v + α bitcoins, while Bob's input contains β bitcoins, for an input sum of v + α + β total. As both inputs have been signed to make the transaction valid and allow it to be published on the blockchain, this allows the parties to atomically commit their money to the transaction. This is an important property for the safety of the two parties: If one of the two parties bails out, the other can leave and double-spend their money freely, as the transaction cannot be included in the blockchain. This atomicity is a significant trust property which is only possible now that blockchain exists as a primitive and was not possible in alternative earlier cryptographic systems. Note that the seller, Bob, also has to put money in the transaction for this payment. At the stage before shipping, Alice has paid an extra α amount for security, while Bob has paid an extra β for security.

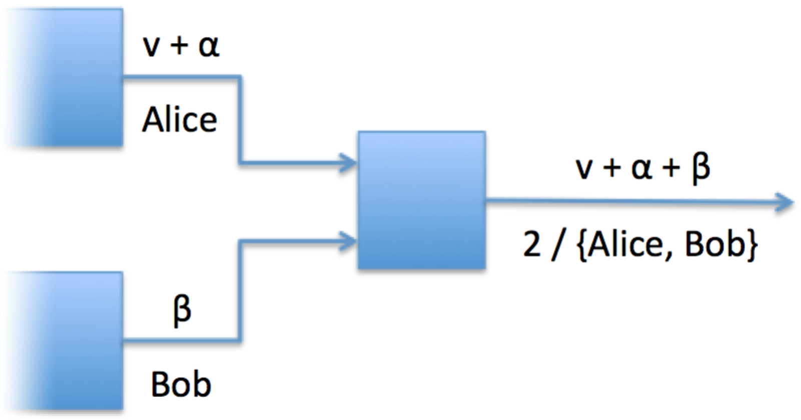

FIGURE 26: A MAD TRADE AFTER THE COMMIT STAGE

The transaction contains one 2-of-2 multisig script output keyed with Alice's and Bob's public keys of total value v + α + β, as seen in Figure 26: A MAD trade after the commit stage. As before, under normal circumstances, upon receiving the proof-of-payment Bob ships the product to Alice. Once Alice receives the product, the two parties finalize the trade in the following way: They create a new transaction, the finalization transaction, with a single input of value v + α + β claimed from the 2-of-2 multisig script output of the previous transaction, which they sign. The new transaction contains two outputs, one pay-to-pubkey-hash of value v + β payable to Bob's public key and one pay-to-pubkey-hash of value α payable to Alice's public key. Upon finalization, the outputs of this new transaction can now be independently claimed by Alice and Bob, as they are now pay-to-pubkey-hash transaction outputs, as seen in Figure 27: A MAD trade after finalization. In the end, Bob has received back his security β and Alice has received her security α; and Alice has also transefered the value v to Bob.

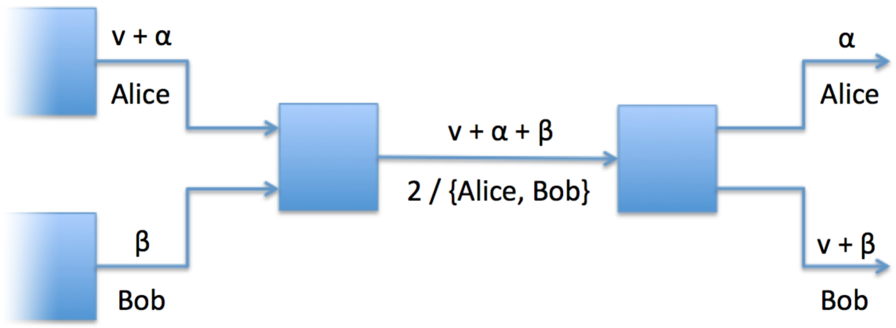

FIGURE 27: A MAD TRADE AFTER FINALIZATION

The MAD scenario has the desirable property that ε-goodness is not required by the players, and so the only Nash equilibrium strategy is the fair strategy. In particular, the rogue strategies described above in the 2-of-2 trade scenario are not applicable here. Specifically, if Alice fails to finalize the transaction, it will cost her a value α extra, which no rational player would be willing to pay (i.e. she would be paying v + α bitcoins for P, which is only worth v bitcoins). On the other hand, Bob is not incentivized to advertise any product P which he doesn't intend to sell. Indeed, if Bob does this, they will be penalized by having to pay out β bitcoins.

A game theoretic strategy summary for the MAD game is presented in Table 3: The MAD trade. The utility for each player represents the assets held, in money and product value. The players start with assets α + v for Alice (holding the money) and β + v for Bob (holding the product). The orange strategy is impossible under the rules of the game (but possible under terrorist negotiation attacks). The red rogue strategy of avoiding finalization costs Alice a value of α. The blue rogue strategoy of avoiding shipping costs Bob a value of β. The status quo strategy is the green strategy.

TABLE 3: THE MAD TRADE

Buyer \ Seller	No game	**Doesn't ship**	**Ships**
No game	v + α, v + β		
Doesn't finalize		0, v	v + λ, 0
Finalizes		α, 2v + β + λ	v + α + λ, v + β + λ

However, “terrorist negotiation” attacks remain possible in this scenario. In addition, the MAD schema is counter-intuitive for most users, as it “freezes” funds on the buyer’s side that are additional to the value of the product, and it requires the seller to also have a non-zero balance in their account, which can become a usability problem, as it remains difficult to educate users on its usage.

It is worthy pointing out that an arbiter can easily be introduced in MAD transactions. The scheme is modified to change the output address from a 2-of-2 multisig to a 2-of-3 multisig, with the inputs and outputs remaining exactly the same, and the arbiter not having to pay for any inputs. The normal trade then commences without arbiter intervention as above. In case of dispute, however, the arbiter is given the option to resolve it at will be reverting the whole transaction, forcing a finalization, or using α and β as collateral for damage. The introduction of an arbiter in both the simple 2-of-2 and the MAD scenario defends against human error of lost keys, but the possibilty of collusion and the question of third-party trust arises again.

For this reason, we consider the most desirable trade type to be the simple MAD trade without arbiters. However, various trade policies can be implemented by any decentralized marketplace, allowing the user to choose which trade type they prefer.

Man-in-the-middle loss of anonymity

This framework is susceptible to a man-in-the-middle attack which is unavoidable in pseudonymous settings. The attack works as follows: A malicious agent wishes to gain trust as a vendor without really being a trustworthy vendor. They first create an OpenBazaar vendor identity. Next, they choose one other vendor that they want to impersonate. They subsequently replicate their product listing as their own. They also monitor the actual vendor's catalog for product changes, and they relay messages between buyers and the actual vendor when buyers message them. When a buyer purchases a product for them, the rogue vendor also forwards the purchase to the real vendor. Notice that this problem does not directly apply to mediators.

If, after the purchase, the buyer and seller rate each other positively, this rating will not impact the actual parties, but will only be reflected on the rogue vendor. Hence, the rogue vendor will gain trust as both a seller and a buyer without actually being either. This process can be automated. At a later time, the rogue vendor can use the man-in-the-middle position to read encrypted messages between buyers and sellers and may sacrifice their maliciously gained reputation to cheat on a desired buyer or seller. Continuous operation of such rogue nodes can undermine the network.

It is difficult to guard against such attacks. The question of whether someone "really" knows a pseudonymous vendor becomes philosophical; what does it mean to know someone who is pseudonymous to you? And if a man-in-the-middle vendor is always delivering goods, are they not also a trustworthy agent? It is recommended that users establish direct trust only with pseudonymous vendors whose real identity they already know, or have signals that the pseudonymous vendor is not being man-in-the-middled. The latter is difficult to establish, but may be possible through independent verification on different networks and a continued trustworthy history. If we assume that the delivery of products will not be intercepted (Zetter, 2013), the product delivery itself may be used as a mechanism to include a physical copy of key fingerprints in order to establish that no man-in-the-middle is being present. The idea of including the vendor's identity reference in the product delivery has been used in the past by vendors in rogue marketplaces such as the various versions of Silk Road and its later descendants such as The Marketplace, Evolution, and so on.

Nevertheless, negative trust is semantically a different notion from positive trust. Negative trust can be attached to vendors whose identity is unknown; if a man-in-the-middle behaves in an untrustworthy manner, it is imperative to rate them negatively. This distinction between positive and negative trust is made clear in the condition for $w(A, C) > 0$ to be positive in the equations above. It is a challenging problem to communicate this difference to the user via a clear user interface.

Vendor-in-the-middle attack

Traditional multisig marketplaces involve a 2-of-2 or 2-of-3 multisig system to secure transactions with or without an arbiter. As we have illustrated, 2-of-2 and 2-of-3 multisig systems in a closed game ensure that ε-good rational agents do not have incentive to play strategies that are unfair to other players.

Trading games between OpenBazaar users are not zero-sum games. As multisig addresses can be used to hold funds for an arbitrarily long time if not enough parties sign-off each transaction, the money in these inputs is essentially burned. As such, participating players can both lose money in these games (observe for example the strategy matrix in **Error! Reference source not found.**).

In the games described above, the system is assumed to be closed. This assumption is necessary to ensure the topology of the game remains unaltered by a potentially malicious, yet rational, party. Lacking the closed-system assumption not only makes these proofs difficult, but in fact makes them impossible, as the theorems no longer hold. This is illustrated in the final claim below, in which a feasible attack against an open system is demonstrated.

In closed systems of trade, we assume that there are two fundamental players, Alice and Bob, who wish to trade some product. If we allow the system to be open, however, malicious parties can introduce additional players as illustrated in the attack below.

Removing the assumption of a closed system invalidates our proofs above, and, in fact, allowing a malicious agent to modify the topology of the trade by introducing additional agents breaks the system's assurances completely. We will illustrate this attack in the following theorem.

Theorem: An open system of a 2-of-3 multisig trade between two ε-good players Alice and Bob can be gamed; that is, there exist strategies in which players playing the status quo strategy have a negative utility.

Proof: We will prove this theorem by constructing an unfair game in which a rational strategy for a malicious agent is demonstrated.

Consider six ε-good players: Two fair buyers:

- Bob with bitcoin address 1B and physical address 2B
- Charlie with bitcoin address 1C and physical address 2B

One fair seller:

- Alice with bitcoin address 1A

Fair arbiters:

- Arbiter1 with bitcoin address 1E1
- Arbiter2 with bitcoin address 1E2

Then, consider a malicious rational agent Mallory with bitcoin address 1M controlling three nodes in the game:

- EvilSeller1
- EvilSeller2
- EvilBuyer

The attack proceeds as follows:

Initially, Mallory creates the EvilBuyer node. Using the EvilBuyer node, Mallory discovers Alice's product D and bitcoin address 1A. Mallory subsequently creates the EvilSeller1 and EvilSeller2 nodes. He attaches bitcoin address 1M to EvilSeller1 and bitcoin address 1A to EvilSeller2. He then creates a duplication of the product listing of D in each of EvilSeller1 and EvilSeller2. This replication can involve minor modifications to the product so that it remains undetectable by automated or manual means; for example, minor changes in the title and description, and different pictures of the same product.

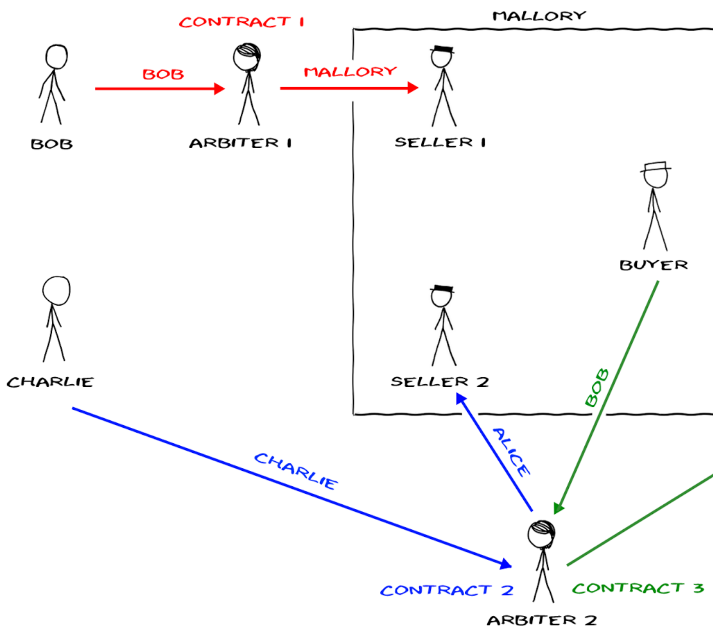

FIGURE 28: THE VENDOR-IN-THE-MIDDLE ATTACK

As Bob and Charlie are interested in the D product, they discover D as sold by EvilSeller1 and EvilSeller2 respectively and they place an order with an evil seller each: Bob places an order of D with EvilSeller1, and Charlie places an order of D with EvilSeller2. In each of these orders, Arbiter1 and Arbiter2 are used, and the following two 2-of-3 multisig addresses are generated:

- Multisig address I: (1B, 1E1, 1M)
- Multisig address II: (1C, 1E2, 1A)

These multisig addresses correspond to two Ricardian contracts:

- Ricardian contract I: (Bob, Arbiter1, EvilSeller1) with a shipping address 2B.
- Ricardian contract II: (Charlie, Arbiter2, EvilSeller2) with a shipping address 2C.

These Ricardian contracts can be signed without a problem by Mallory, as it is not necessary to be in posession of the private keys corresponding to 1A to create a valid signature.

Subsequently, Bob and Charlie fund these addresses with the price of D, as they expect the trade to complete normally. At this point, Mallory is aware of 1B and 2C, so he can use these in subsequent information exchanges.

As soon as these multisig addresses are both funded, Mallory sets the bitcoin address of EvilBuyer to 1C and uses EvilBuyer to place an order with Alice using Arbiter2 as an arbiter for the contract, but does not create a new multisig address. A third Ricardian contract is signed, in which Mallory decides the shipping address. In this contract, Mallory critically choses 2B as the shipping address:

- Ricardian contract III: (EvilBuyer, Arbiter2, Alice) with a shipping address 2B.

Again this Ricardian contract can be signed by Mallory without the need to be in control of the shipping address 2B or the bitcoin address 1C.

Once the third Ricardian contract is in place, EvilBuyer demonstrates that a payment has been executed by illustrating to Alice that Multisig address II has been funded. This concludes a valid proof-of-payment which is accepted by Alice, as it is funded with the correct amount of money, and includes the three expected bitcoin addresses corresponding to Ricardian contract III. Upon inspection of the valid proof-of-payment, Alice ships D to 2B.

At some point, Bob receives his product as promised in Ricardian contract I and finalizes multisig transaction I by signing it off.

Upon finalization, Mallory receives the value of D without having shipped any products, and departs from the game. ■

At this point, Alice will not have received any payment for the product she shipped, and Charlie will have paid without having received a product. Both of them will file a dispute with Arbiter1 and Arbiter2, but the dispute remains unresolvable.

We notice that trust between players either through sybil protection mechanisms or webs-of-trust, as described below, are insufficient to protect against this attack. In particular, Mallory is able to execute a trusted attack by gaining arbitrary trust for her buyer and seller nodes and subsequently executing a large-scale attack. While proof-of-burn mechanisms offer some assurance up to a certain amount, the whole purpose of multisig is completely defied by the attack.

As the arbiters only played a passive role in the attack, clearly this attack is also possible in a 2-of-2 trade game.

Such attacks are exploitable in any marketplace where direct purchases between buyers and sellers is allowed.

The defence against such attacks is pretty straightforward: The shipping address must be authorized by the party making the payment. Hence, a signature with the private ECDSA key that corresponds to the bitcoin address making the payment must be placed on the Ricardian contract also. This resolves the problem.

Friendly names

While we provide a trust mechanism through OpenBazaar which enables the estimation of risk through a web-of-trust and through reputation pledges, it is critical that users are able to use third-party channels for the discovery of stores. These third-party channels can be online forums and websites, or direct person-to-person communication through e-mail or instant messaging, in which they share a store with one another. In addition, a store may make a name of itself and become memorable, and a user may wish to revisit it in the future.

In such situations, we wish to provide the user with a memorable name, in the form of a URL, which can be used to recall the store later and share it easily with friends. In addition, store owners can make use of this name to build word-of-mouth reputation. Importantly, users that remember a user-friendly URL can then come back to that same store later, without needing to store long keys. For these reasons, it is imperative that stores can, with some limitations, choose the name they want to use, and that the names can remain memorable and short. For example, names such as “etsy” and “amazon” should be usable. This property is called name human-meaningfulness.

As such names will be used to identify stores, it is of essence that the names are impossible to replicate by an attacker. Once a store name is used by its rightful owner, an attacker should not be able to make use of the exact same name. “Rightful” is defined in a first-come-first-serve basis; the enforcement of trademarks through centralized legal frameworks is impossible and undesirable in a decentralized setting. Furthermore, the owner of a name for a store should be able to prove the ownership of that store and be the only one that can act as the administrator of the store. These abilities involve modifying products, entering into agreements, deleting the store, accepting payments, and so forth. Furthermore, the owner of a name should be able to transfer it to a new owner at will. This property of being able to manage a name on your own is called name security (Loibl, 2014).

Finally, names must not be attackable under the OpenBazaar threat model. There should not be any Achilles’ heel, and governments or corporations should not be able to seize names the find undesirable. Copyrights, trademarks, and traditional laws should not play a role in who owns a name, as they can be manipulated by law-making governments or lobbying corporations. These actors should in addition be unable to perform denial-of-service attacks on the name system. As such, names should be appointed in a first-come first-serve basis, with the first actor to claim a name becoming the owner of that name. This property is called decentralization .

GUIDs

As mentioned previously, OpenBazaar protocol exchanges are digitally signed by ECDSA keys. These ECDSA keys are different from bitcoin address keys. They are used to collect the purchases and sales of an OpenBazaar node and to build their decentralized pseudonymous reputation. The public portion of these ECDSA keys hashed under an appropriate algorithm constitutes the GUID of a store. This GUID can be used as an identifier for the store in a secure and decentralized manner. However, the GUID does not have the human-meaningfulness property.

We will now explore ways in which human-meaningfulness can be added to OpenBazaar store names.

Zooko's Triangle

Zooko conjectured that "you cannot have a namespace which has all three of: distributed, secure, and having human-meaningful keys." Introducing Zooko's triangle, he claimed that any two of these desirable properties can be combined, but not all three (Wilcox-O'Hearn, 2001). It is worthy at this point to repeat some definitions; that of a namespace system, of its security, decentralization and human-meaningfulness. A namespace system is a system that assings names to entities in a distributed system. Such a system can have some properties. The property of security in a namespace system indicates that a given node in the network has to authenticate to claim a name; that is, a name cannot be claimed arbitrarily once it has been assigned to a node in the network. The decentralized property requires the system to have no central points of failure or ownership, i.e. this system much enjoy soverignty rights. This incidentally means that the system is not prone to legislative control and cannot be shut down by traditional means; instead, to shut down the network, all nodes must be eliminated. Finally, the property of human-meaningfulness means that the names in the system are readable and memorable by humans and, more importantly, useres can freely choose their names from a set of names (provided that, if the security property is also desired, a name once reserved cannot be reused by a different party without authorization).

In the next paragraphs, several examples of systems which successfully make use of some of these properties are presented. Finally, Namecoin, a system which successfully "squares Zookoo's triangle" in a form that solves the conjecture in the negative, with a complete implementation, is presented. We make use of this system for OpenBazaar's identity system and explain its necessity.

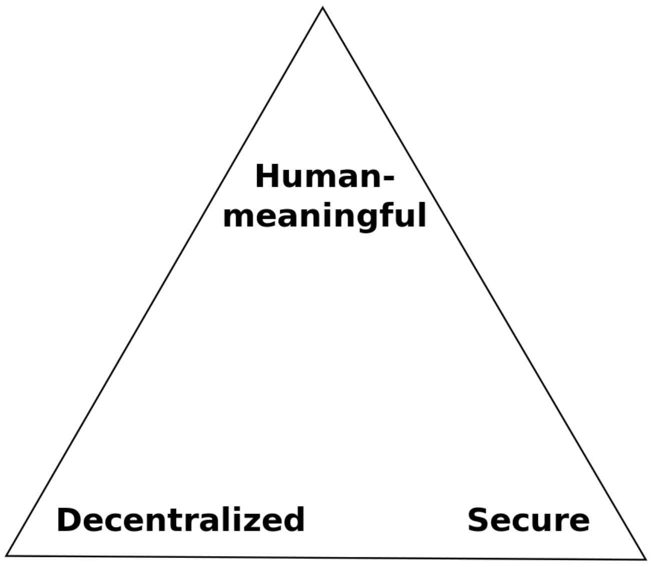

FIGURE 29: ZOOKO'S TRIANGLE

Figure 29: Zooko's Triangle shows Zooko's Triangle conjecture: The vertices indicate the three desired properties of human-meaningfulness, decentralization, and security. Any name system can lie on any edge of the triangle and have two vertices adjacent to it, but not all three. We will now proceed to illustrate by way of example various systems that go from lacking all three properties, to having all three properties, in order to illustrate the meaning of the conjecture and to show that it is, in fact, false (Swartz, 2011).

Clearly, a name system can lack all three properties. An example name system lacking all of these three properties is the IP-naming system in a local network using DHCP (Droms, 1997). It is not decentralized, as a centralized router is used to manage the names and can, for example, deny service or block specific IP addresses at will. It is also not secure, as IP addresses can be spoofed in the Internet Protocol by simply constructing a custom packet that contains as source the IP address to be spoofed (Tanase, 2003). Finally, it is not human-meaningful, as IP addresses are a simple series of numbers and not memorable.

Several real-world systems exhibit one of the three desired name properties.

The plain old DNS system (Mockapetris, 1987) has one of these properties, namely human-meaningfulness. DNS is not decentralized, as the root nameservers essentially control the whole hierarchy of the system and can be commanded to dissolve names. Furthermore, nodes within the hierarchy are in central command of the names that belong to them, and there have been numerous instances of domain name seizures. For example, a famous case with political underpinnings was the theft of Kim Dotcom's Megaupload domain name (Sisario, 2012) (Graeber, 2012). In addition, DNS is not secure, as the protocol is not authenticated. Therefore, attackers who are in control of the network layer or the data link layer are able to easily modify data by performing a man-in-the-middle attack (DNSCurve Team, 2009).

The PKI system used in TLS for HTTPS is an example of a system which exhibits only the second such property, security, where certificate key fingerprints are treated as the names. Clearly, key fingerprints are not human-meaningful. Key fingerprints are also centralized, with a set of root certificates being the central points of failure. In this case, lack of decentralization is evident in the fact that PKI is hierarchical. While this centralization is generally undesirable, it was successfully used for good purpose in the comodohacker incident (Bright, 2011) by browsers, in which a certificate was invalidated successfully through legal orders and agreement. This centralization has allowed vendors to remove certain certificates from the hierarchy root at will (Nightingale, 2011).

The property of decentralization alone can be found in naming systems such as mesh networks. In these networks MAC addresses are names which are truly decentralized, due to the fact that each device can decide about its own address (IEEE Standards Association). However, the system is not secure because MAC addresses can be spoofed (Pahwa, Tiwari, & Chhabra, 2010), and it is not human-meaningful, as MAC addresses are simply series of incomprehensible numbers.

Next, let's explore a few systems that expose any pair of Zooko's properties.

Tor (Galperin, 2014) contains a name system which is both decentralized and secure, yet it is not human-meaningful. Indeed, a tor hidden service (Dingledine, Mathewson, & Syverson, Tor: The second-generation onion router, 2004) name is secure in the sense that noone can immitate a name. In particular, because names are derived by hashing the public part of an asymmetric cryptographic key (Dingledine & Mathewson, Tor Protocol Specification, 2015), and controlling a name requires being in control of the respective private key. As such, stealing a name would require brute-forcing an asymmetric private key. Furthermore, the system is decentralized because no central authority is in control of the names. Each hidden service is truly owned by its creator and no legal power can take it away from them. Finally, names are not human-meaningful, as they are the output of a cryptographic hash function.

mDNS, a simplified DNS system to be used by small devices (Cheshire & Krochmal, 2013) is both decentralized and human-meaningful, but not secure. In mDNS, no central authority decides on the names, but each device can claim its own name, hence it is decentralized. As the names can be choosen by the devices themselves, they are human-readable. Finally, noone prevents a device from choosing the same name as another device, and hence the system is not secure.

DNSSEC, a secured version of DNS (Arends, Austein, Larson, Massey, & Rose, 2005) hopelessly under deployment for decades is a naming system which is both secure and human-meaningful, but not decentralized. DNSSEC inherits the centralization properties of DNS, yet it also introduces security by digitally signing DNS records when they are exchanged.

The following table summarizes the ways in which the Zookoo properties can be satisfied, with a relevant example for each combination:

TABLE 4: ALL COMBINATIONS OF ZOOKO'S PROPERTIES IN NAME SYSTEMS

Human	**Secure**	**Decentralized**	**Example**
No	No	No	DHCP IP
No	No	Yes	Mesh MAC
No	Yes	No	PKI
Yes	No	No	DNS
No	Yes	Yes	Tor
Yes	No	Yes	mDNS
Yes	Yes	No	DNSSEC
Yes	Yes	Yes	Namecoin

Namecoin

Namecoin makes it possible to have names that exhibit all three of the desired properties (Slepak).

Namecoin is a bitcoin fork (Gilson, 2013). In fact, it is the first fork of bitcoin. By using a blockchain, it arrives at decentralized consensus and hence, similarly to bitcoin, cannot be brought down by central authorities or the law. Namecoin is secure. This security is achieved in a similar way to Tor: The owner of each name is authorized by proving the ownership of a private asymmetric cryptographic key. The public key that corresponds to this private key is first published to the blockchain when a name is first registered ("name_new" operation). Further updates to the name require proof of ownership of the private key whose respective public key was published during registration. Updates to names can include changing the value the name corresponds to (similar to how DNS associates names with IP addresses), or transfering ownership of the name to a new key. Finally, human-meaningfulness is achieved by allowing each user to choose their desired name freely. The names are registered in a first-come-first-serve basis by enforcing a blockchain-based policy similar to the prevention of double spending in bitcoin.

OpenBazaar and Namecoin Integration

In OpenBazaar, each GUID is associated with a user-friendly name. These user-friendly names can be used as mnemonic names: If someone loses their trust network by reinstalling the node without first exporting, they know that certain agents remain trustworthy. Furthermore, user-friendly names are used in the trust bootstrapping procedure in which it becomes easier to peer-review that the bootstrapped nodes are the correct ones. Finally, user-friendly names can be exchanged between users out of the software usage scope; for example, a user can directly recommend a vendor by their user-friendly name to one of their friends via e-mail.

To maintain a cryptographically secure association between node GUIDs and user-friendly names, we utilize the Namecoin blockchain. A node can opt-in for a user-friendly name if they so choose. To create a user-friendly name for their GUID, they must register in the "id/" namecoin namespace (Namecoin Developers, 2014) with their user-friendly name. For example, if one wishes to use the name "dionyziz", they must register the "id/dionyziz" name on Namecoin. The value of this registration is a JSON dictionary containing the key "OpenBazaar" which has the GUID as its value. As Namecoin ids are used for multiple purposes, this JSON may contain additional keys for other services. The namecoin blockchain ensures unforgeable cryptographic ownership of the identity. When a node broadcasts its information over the OpenBazaar network, they include their user-friendly name if it exists. If a node claims a user-friendly name, each client verifies its ownership by performing a lookup on the namecoin blockchain. If the lookup succeeds, the name is displayed on the OpenBazaar GUI and the information is relayed; otherwise the information is discarded.

As namecoin names can be transferred, this allows for participants to transfer their identity to other parties if they so desire; for example, a vendor can transfer the ownership of their store by selling it.

As part of our work, and with the assistance of cryptography students Chara Podimata and Kostis Lolos, we have proposed and implemented a full integration between OpenBazaar and namecoin as part of this work.

In our implementation, we allow each store to be associated with a name in the namecoin “id/” namespace. These ids are mutually authenticated. On one hand, the owner of the id/ namespace name must indicate that they authorize the use of their name for an OpenBazaar store. This is done by including the store’s GUID in the JSON value associated with the namecoin key. On the other hand, stores can indicate their namecoin names and claim them when they exchange messages with other nodes on the network to indicate their store metadata. As this data is signed with the store’s ECDSA private key associated with the store’s GUID, this indicates an authorization by the store for the name use. The augmented JSON key is called ‘openbazaar’. An example namecoin entry for the author is shown in Listing 4: An example namecoin "id/" entry containing an OpenBazaar key.

```
{
    "name": "Dionysis Zindros",
    "xmpp": "dionyziz@gmail.com",
    "namecoin": "NBbz5d5KH8XBYbrJ7gGFmPWHzigkvRQJwR",
    "gpg": "45DC 00AE FDDF 5D5C B988  EC86 2DA4 50F3 AFB0 46C7",
    "bitcoin": "1vXhpZpeDWLmp7vN7k52x3WwRSZK3DT6X",
    "openbazaar": "6c80b332c81a880c1c0e06982d4ae94ac00e0bd5",
    "otr": [
        "513BCE4D E9E11585 F475FDFD 52462B7F 160A5753",
        "6E25C452 B624D392 56A74B45 85D04502 EE73F5F4"
    ],
    "bitmessage": "BM-2cVGoqb4wxJNMjCp1ftp6hVVvx7uitaYMu",
    "email": "dionyziz@gmail.com"
}
```

LISTING 4: AN EXAMPLE NAMECOIN "ID/" ENTRY CONTAINING AN OPENBAZAAR KEY

Namecoin integration allows users to remember store names and increases the trust in the system. Users can exchange names with their friends easily and payments can be made to names instead of unreadable identifiers. While we have not implemented this, a URL schema can easily be implemented so that a store can be visited by entering its URL in a user-friendly format, for example openbazaar://dionyziz. Clicking the URL can open up the store, and visiting a store by any means, even if the URL was not clicked, can securely dislpay the user-friendly name to the user for validation purposes, similar to the way users are expected to validate domain-name trust in the HTTPS security model.

```
def is_valid_namecoin(namecoin, guid):
    if not namecoin or not guid:
        return False
    server = DNSChainServer.Server(constants.DNSCHAIN_SERVER_IP, "")
    _LOG.info("Looking up namecoin id: %s", namecoin)
    try:
        data = server.lookup("id/" + namecoin)
    except (DNSChainServer.DataNotFound, \
DNSChainServer.MalformedJSON):
        _LOG.info('Remote namecoin id not found: %s', namecoin)
        return False
    return data.get('openbazaar') == guid
```

LISTING 5: A NAMECOIN VALIDITY CHECK IMPLEMENTATION

```
class Server():
    """A connection to a DNSChain server."""

    def __init__(self, addr,
                 fingerprint, http_host_header='namecoin.dns'):
        """
        Store configuration for requests to a DNSChain server.

        @param addr: The address of the trusted DNSChain server
        @param fingerprint: The key fingerprint of the DNSChain
server, for connection authorization
        """
        self._logger_helper(__name__)
        self.addr = addr
        self.fingerprint = fingerprint
        # Per http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html
        self.headers = {'Host': http_host_header}

    def lookup(self, name):
        """
        Looks up a name from the DNSChain server.
        @param name: The name to lookup, e.g. 'id/dionyziz'
        """
        full_url = "http://%s/%s" % (self.addr, name)
        request = urllib2.Request(full_url, None, self.headers)
        try:
            response = urllib2.urlopen(request)
        except urllib2.HTTPError, e:
            if e.code == 404:
                raise DataNotFound(e, name, self.headers['Host'])
        namecoin_string = response.read()
        try:
            data = json.loads(namecoin_string)
        except ValueError:
            raise MalformedJSON("%s\n%s" \
            % (ValueError, namecoin_string))
        return data
```

Listing 6: Parts of the PYDNSChain Implementation

A basic implementation of a namecoin validity check is shown in Listing 5: A Namecoin Validity Check Implementation. As part of this work, we have also developed a Python library for looking up namecoin names from centralized servers. The idea is that the OpenBazaar user can choose to use a namecoin server, or operate their own, while the OpenBazaar client simply issues HTTP-based queries to it. Parts of our pydnschain implementation are shown in Listing 6: Parts of the PYDNSChain Implementation.

Further research

In the following paragraphs, we briefly touch on topics that can be of interest to other researchers in the field. We explore some initial ideas in these areas, but it remains clear that the details are left incomplete and additional work is required to explore these possibilities.

Trusted roles

It may be meaningful to distinguish trustworthiness of a pseudonymous individual in different roles; for example, a person who is a trustworthy merchant may not be a trustworthy judge. Hence, trust to buy from a person may be different from trust to mediate. In our web of trust, we have assumed that this trust is the same under our projection mechanism. A fruitful research direction seems to be the ability to distinguish between roles and provide a generic framework for establishing trust in a decentralized setting. This could be the cornerstone of a decentralized electronic state, in which individuals provide attestments that their peers hold degrees or are trustworthy drivers.

GPG also distinguishes between trusting the identity binding between a real-world individual and a key and the trust directly given to an individual as far as they are concerned about *signing* other keys. In a similar setting as blurring the trust between different roles, the OpenBazaar for better or for worse considers these to be the same. In the OpenBazaar web-of-trust, trust is an intuitive concept and there need to be no formal rules followed when trust is given to others. The every-day statement "I trust this person" corresponds to actually giving trust to an individual. This is in contrast to the GPG web-of-trust in which signing keys requires a certain procedure of identity verification with which individuals may not be familiar with, and hence this differentiation is in order. However, this interpretation may be unsubstantiated and more research is needed to study the usability and intuitiveness of this scheme from a human-machine interaction point of view. Distinguishing the role of an assurer may also become a necessity in a universal trust network; as well, the roles for which an assurer can assure may be limited. This may require a set of tags for which people are assured, one of which may be the ability to assure.

Furthermore, we do not distinguish between uncertainty of trust and certainty in neutral trust as other authors (Jøsang, 1999). It may be helpful to employ such distinction in future versions of our web-of-trust.

Feedback and reviews

Feedback can be given by buyers to vendors, by vendors to buyers, and by buyers and vendors to mediators in text form. Feedback is a piece of text from a particular source pertaining to a particular target. Keeping the above man-in-the-middle vendor attack in mind, it may in cases not make sense to rate vendors or buyers directly if their real identity is unknown by the rater, even if they trade fairly, unless they can establish an existing trust relationship towards them in order to at least determine their legitimacy.

To avoid fake feedback, feedback must only be relayed by the OpenBazaar client if it is from a node that has transacted with the target. Therefore, feedback must be digitally signed and include a reference to the transaction that took place – the hash of the final Ricardian contract in question, as well as the bitcoin transaction where it was realized.

However, given that transactions are free to execute, to avoid Sybil attacks from vendors or buyers who transact with themselves, feedback must only be trusted when it is given from parties that are already trusted using the total trust metric defined below. Otherwise, it must not be displayed or relayed.

Association with other identity systems

It is worthy to attempt an association of the web-of-trust network with other identity management systems. However, given the highlighted differences in the previous section, such an association would compromise some of the security assumptions of the model. It is therefore mandatory that interconnection with other networks is an opt-in option for users who wish to forfeit some of our security goals.

An interconnection with the GPG web-of-trust may be achieved by allowing OpenBazaar nodes to be associated with GPG keys. A particular OpenBazaar node can have a one-to-one association with a GPG identity through the following technical mechanism: The GPG identity can provide additive trust to the existing trust of the system. To indicate that a GPG key is associated with an OpenBazaar identity and that the GPG key owner wishes to transfer the GPG trust to an OpenBazaar node, the GPG key owner cryptographically signs a binding contract which contains the OpenBazaar GUID of the target node, and potentially a time frame for which the signature is valid. The GPG-signed contract can then be signed with the OpenBazaar cryptographic key to indicate that the OpenBazaar node operator authorizes GPG trust to be used for their node. The double signed contract can then be included in the metadata associated with the OpenBazaar node and distributed through the OpenBazaar distributed hash table. Each client can inter-process communicate with the GPG software instance installed on the same platform to obtain access to existing keys and signatures.

Nevertheless, it is advised not to include such an implementation in the canonical client, as traditional GPG webs-of-trust are identity-verifying, not trust-verifying, as explored in the section above. Furthermore, the GPG web-of-trust does not offer any assurances on pseudonymity. While it is possible to exchange GPG signatures anonymously, the GPG web-of-trust is typically based on global topological knowledge of the GPG graph and is distributed through public keyservers. If a user opts-in to interconnect with the GPG web-of-trust, they are forfeiting these benefits of the OpenBazaar network.

An interconnection with the Bitcoin OTC web-of-trust is also possible. Existing trust relationships can be imported to the OpenBazaar client manually through a file, or automatically downloaded from the Bitcoin OTC IRC bot dynamically upon request, as the Bitcoin OTC website is an insecure distribution channel and does not offer HTTPS. In the dynamic downloading case, the threat model is reduced to trusting the TLS IRC PKI, which is known to be attackable by powerful third parties (Adkins, 2011).

This web-of-trust has the benefit of being a true-trust web-of-trust, and has a history of support by the Bitcoin community. A double signature is again required to interconnect two identities. The GPG key associated with the Bitcoin OTC network is used to cryptographically sign a binding contract similar to the one described above, and the rest of the procedure is identical to GPG identity binding.

In this case, an Achilles' heel is introduced to the software, as the user is required to trust the Bitcoin OTC web-of-trust operator and the distribution operator, both of which identities can possibly be compromised by a malicious third party; the system is centralized. The situation can be improved if the Bitcoin OTC operator begins GPG signing the trust network, or by requiring the Bitcoin OTC trust edges to include GPG signatures by the users involved. However, the topology of the network is again public, forfeiting pseudonymity requirements. Therefore, an implementation is again not advised for the canonical OpenBazaar client at this time.

If the user is not concerned with single-points-of-failure and centralization, the web-of-trust can be temporarily bootstrapped by binding identities to existing social networking services which include edges between identities as "friendships" or "follows". For example, the Twitter and Facebook networks can be used. Such bindings can be weighted with a low score in addition to existing scores as described in the Total Trust section below. The author strongly advises against such interconnections. The threat model of the OpenBazaar network is completely forfeited if such trust relationships are used. Centralized, non-anonymous services for trade such as eBay are widespread and can be used in OpenBazaar's stead if these assurances are of no concern to the user.

Further research is required to determine how such interconnections will impact the security of the network.

Turing complete blockchains

Turing-complete blockchains allow particularily interesting explorations. Further research is required to determine whether they are feasible. One could use a turing-complete blockchain to enable more advanced rating systems and trust. These could be crafted so that they are game-theoretically sound and provide better assurances than our trust system. Timelock mechanisms can also be explored in these schemes.

Trust-as-risk

Trust in this work was not quantified using a formal mathematical definition. We believe this is possible by assigning monetary values to the web-of-trust edges. This idea was first put forth by Washington Sanchez of the OpenBazaar team. These monetary values can indicate quantitatively how much money exactly each party is willing to risk to support their trust towards another party. While we don't fully understand it, we hope this model will allow more precise game-theoretic and probabilistic proofs to take place, giving much needed confidence to these financial cryptographic webs-of-trust. We are excited about this research direction and encourage researchers to explore this alterantive option of modeling trust.

Trust propagation and storage

The exact technical details on trust and reputation storage have not been explored in this work. One option would be to use a DHT with infohashes and a torrent-like mechanism for distribution. Blockstack's blockstore (Ali, 2015) provides interesting primitives that could be leveraged.

Circular trust propagation and convergence

The simple transitivity scheme which we use on the web-of-trust is flawed and it must be reworked before implementation. Currently, trust is impossible to calculate if cycles exist in the network graph. As the topology is only partially known, it is not trivial to detect cycles. Furthermore, basic cycle detection mechanisms used in routing protocols may easily compromise the anonymity of the network by revealing non-local topological information. The formulae to deduce trust must be reworked to take this important issue into consideration. It may be possible to avoid cycle detection and employ a mechanism which converges to the actual trust with good probability if some randomness factor is used to decide responding to queries whose answer is not known.

Cycles could be detected, for example, by carrying unique salted node hashes as a path is constructed, which allow nodes to check for themselves within a cycle path. More work is needed to establish such schemes.

Ethical considerations

The ethical considerations behind building OpenBazaar are important, even more significant than usual decentralized systems.

A marketplace like OpenBazaar replaces traditional online marketplaces in a very powerful and useful way that enables vendors and buyers to truly own their puchases, without censorship. It creates an environment where trade is truly free; this is a broader freedom than freedom of speech which is usually employed by decentralized systems. While OpenBazaar is simply a carrier of information, it is also an enabler of real-world activity. As such, it can be used for good and for evil.

OpenBazaar allows for many illegal activities to take place, and, if used correctly, makes it impossible for law enforcement to intervene. As law enforcement often interrupts illegal activity at points-of-sale, OpenBazaar makes it harder for them to operate, because points-of-sale are neither centrally controlled nor physical. Physical sting operations that police have traditionally used at point-of-sale locations to catch criminals now become impossible, especially because of the anonymity properties of OpenBazaar. Furthermore, the decentralized nature of the system makes censoring illegal products impossible. Each OpenBazaar store must be dealt with and closed down independently, through a physical interception of goods delivered. At the same time, as physical product delivery becomes more anonymous, for example through drone network or by hiding between the enormous volume of business letters, these operations will become harder and harder. When it comes to goods digitally delivered, interception during product dispatching becomes impossible.

The potential malicious uses of OpenBazaar are staggering. The simplest illegal activities that can be condoned are marginally illegal. Embargoes can be circumvented, porn production can be enabled in countries where it is illegal, tax can be evaded (Torpey, 2014). But the illegal uses of a truly decentralized anonymous marketplace range beyond these simple activities. Even the most harmful drugs can be freely sold (Kumar, 2015) without any accountability. Extreme and harmful pornography, such as child pornography (Bravura, 2014), can be freely traded for money. Gun parts and weapons can be traded (Shubber, 2014), including materials for mass-destruction weapons, among others nuclear, chemical and biological weapons. Finally, people can be traded as slaves.

As researchers and engineers building these new technologies, we must be constantly aware of the ethical implications of our work. We bear the responsibility to create technologies for good. While we understand that the decentralized marketplace can be used for evil, we treat it as an underlying transport technology. Similar to the Internet, which can host good and bad content, a marketplace can also be used for any content the users deem appropriate. We employ optional mechanisms to filter inappropriate content from the eyes of users who wish to stay legal; this is the best we can do in an open source decentralized setting.

We strongly believe such liberated marketplaces will be overwhelmingly used for good purpose and only a minority will use it for bad purposes. While the bad actors can never be completely avoided, we feel we are building an important infrastructure for future generations. Roads and bridges can be used by both criminals and honest workers as means of transportation; this argument has never hindered the progress of science and engineering. As such, we consciously decided to move forward with this implementation, albeit aware of its potential bad uses. We urge future researchers in the area to be mindful of the ethical implications of their work.

Conclusion

We have built an anonymous decentralized marketplace based on bitcoin, OpenBazaar, which we developed using secure practices and a clear threat model.

We presented the security of OpenBazaar by first introducing the required cryptographic primitives and by putting it within the historical economic context. We presented the essential theory behind bitcoin, which is a cornerstone of the network.

We introduced a new mechanism to allow for fully pseudonymous webs-of-trust. The web-of-trust is developed in a way that allows conveying true trust and not simple identity verification. Furthermore, anonymity is preserved through only partial topological disclosure. The notion of graph distance is maintained through an attenuation factor. The web-of-trust is designed to be used in a commercial setting, where trust is an indispensable tool for trade. Trust propagates using multiplicative properties.

The web-of-trust is bootstrapped through trust to a seed set of nodes, which requires the user to explicitly opt-in if they wish to trade illicitly.

Introduction to the trust system is obtained through global trust, which is weighted along with projected trust. Global trust is achieved through a proof-of-burn mechanism, and an alternative proof-of-timelock mechanism is explored. Together, these mechanisms allow a calculation of total trust between nodes. We presented specific formulae which can be used to achieve numeric results. Global trust is accompanied by friendly names which are achieved through an integration with namecoin, which plays an important role in users' trust.

In response to our proposals, we illustrated security concerns on mechanisms that are not appropriate for such an implementation, such as proof-of-donation and proof-to-miner, and we developed an attack on the web-of-trust through graph separator control.

In addition to the trust notions we developed, we presented self-enforcing contracts to avoid the need for trust altogether. We explored ricardian contracts and we used game theory to analyze 2-of-2, 2-of-3, and MAD trades. Finally, we explored attacks against such contracts such as the man-in-the-middle anonymity attack, the terrorist negotiation attack, and the vendor-in-the-middle attack.

Overall, this anonymous marketplace system can be used in a wide commercial setting securely and offers strong assurances of trust in which large financial trades can be performed safely.

Bibliography

Addley, E., & Halliday, J. (2010, 12 08). Operation Payback cripples MasterCard site in revenge for WikiLeaks ban. *The Guardian* .

Adkins, H. (2011, 08 29). *An update on attempted man-in-the-middle attacks.* Retrieved 11 30, 2015, from Google Online Security Blog: https://googleonlinesecurity.blogspot.gr/2011/08/update-on-attempted-man-in-middle.html

Ali, M. (2015, 04 21). *Data Storage.* Retrieved 11 30, 2015, from Blockstore Documentation: https://github.com/blockstack/blockstore/wiki/Data-Storage

Allison, I. (2015, 11 20). Move over eBay: Countdown to OpenBazaar and the decentralised marketplace revolution. *International Business Times* .

Andresen, G. (2012, 01 03). *Pay to Script Hash.* Retrieved from BIP.

Antonopoulos, A. M. (2014). *Mastering Bitcoin.* O'Reilly Media.

Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005, 03). *DNS Security Introduction and Requirements.* Retrieved 11 30, 2015, from IETF: https://www.ietf.org/rfc/rfc4033.txt

Arinich, G. (2013, 10 11). *Decentralized Anonymous Marketplace: The Concept*. Retrieved 11 14, 2015, from GitHub: https://github.com/goshakkk/decentralized-anonymous-marketplace-concept

Back, A. (2002, 08 01). Hashcash - A Denial of Service Counter-Measure.

Barratt. (2012). Silk Road: eBay for drugs. *Addiction , 107* (3), 683.

Biggs, J. (2013, 12 05). Who Is The Real Satoshi Nakamoto? One Researcher May Have Found The Answer . *TechCrunch* .

Biham, E., & Chen, R. (2004). Near-Collisions of SHA-0. *CRYPTO* .

Bitcoin Developers. (n.d.). *Bitcoin Developer Guide.* Retrieved 11 16, 2015, from bitcoin.org: https://bitcoin.org/en/developer-guide

Bitcoin Developers. (n.d.). *Protocol documentation*. Retrieved from Bitcoin wiki: https://en.bitcoin.it/wiki/Protocol_documentation#tx

Bitcoin Developers. (n.d.). *Script.* Retrieved 11 28, 2015, from Bitcoin wiki: https://en.bitcoin.it/wiki/Script#Anyone-Can-Spend_Outputs

Bitcoin Foundation. (2013, 10 24). *Core Development Update #5.* Retrieved 11 28, 2015, from Bitcoin Foundation: https://bitcoinfoundation.org/core-development-update-5/

Bos, J., Kaihara, M., Kleinjung, T., Lenstra, A., & Montgomery, P. (2009). On the Security of 1024-bit RSA and 160-bit Elliptic Curve Cryptography. *IACR Cryptology* .

Brabec, B. (1998). *Handmade For Profit.* USA: M. Evans.

Bravura, M. (2014, 07 22). OpenBazaar: Blazing the trail for bitcoin commerce without barriers. *Cryptocoin News* .

Bright, P. (2011, 09 07). Comodo hacker: I hacked DigiNotar too; other CAs breached. *arstechnica* .

Buterin, V. (2014). A next-generation smart contract and decentralized application platform.

Cheshire, S., & Krochmal, M. (2013, 02). *Multicast DNS.* Retrieved 11 30, 2015, from IETF: https://tools.ietf.org/html/rfc6762

Chulov, M. (2012, 11 29). Syria shuts off internet access across the country. *The Guardian* .

Clarke, I., Sandberg, O., Wiley, B., & Hong, T. W. (2001). Freenet: A Distributed Anonymous Information Storage and Retrieval System. In H. Federrath, *Designing Privacy Enhancing Technologies* (pp. 46-66). Berkeley, USA: Springer-Verlag Berlin Heidelberg.

Clinch, M. (2014, 03 07). 'Real' bitcoin creator: 'I am not Dorian Nakamoto' . *CNBC* .

Corbin, K. (2014, 01 21). Litigation Over PayPal Holds Languishes in Court . *eCOMMERCEBYTES* .

CounterParty. (2014, 03 23). *Why Proof-of-Burn.* Retrieved 11 25, 2015, from counterparty.io: http://counterparty.io/why-proof-of-burn/

Dai, W. (1998, 11). *b-money*. Retrieved 11 14, 2015, from Wei Dai: http://www.weidai.com/bmoney.txt

Davis, J. (2011, 10 10). The Crypto-Currency. *The New Yorker* .

Dingledine, R., & Mathewson, N. (2015, 08 12). *Tor Protocol Specification*. Retrieved 11 30, 2015, from https://github.com/torproject/torspec/blob/master/tor-spec.txt

Dingledine, R., Mathewson, N., & Syverson, P. (2004). Tor: The second-generation onion router. *Naval Research Lab* .

DNSCurve Team. (2009, 06 24). *Usable security for DNS*. Retrieved 11 30, 2015, from DNSCurve: http://dnscurve.org/forgery.html

Dolgov, A. (2015, 05 15). Paypal Blocks Russian Account Linked to Nemtsov Report on Ukraine. *The Moscow Times* .

Douceur. (2002, 03). The Sybil Attack. *Proceedings of the IPTPS '02 Workshop* .

Driessen, V. (2010, 01 05). *A successful Git branching model.* Retrieved 11 19, 2015, from Thoughts and writings by Vincent Driessen: http://nvie.com/posts/a-successful-git-branching-model/

Droms, R. (1997, 03). *Dynamic Host Configuration Protocol.* Retrieved 11 30, 2015, from IETF: https://tools.ietf.org/html/rfc2131

European Union. (2008). On the statute of the European system of central banks and of the European Central Bank. *Official Journal of the European Union*, (p. 8).

Evans, Z. (2014, 10 02). Agora Is the Web's Top Black Marketplace. *reason* .

Eyal, I., & Sirer, E. G. (2014). Majority Is Not Enough: Bitcoin Mining Is Vulnerable. *Financial Crypto* , 436-454.

Folkinshteyn, D. (n.d.). *OTC web-of-trust.* Retrieved 11 25, 2015, from #bitcoin-otc: https://bitcoin-otc.com/trust.php

Freedom House. (2015, 05 14). PayPal Blocks Donations for Report on Russian Actions in Ukraine . *Freedom House* .

Freenet Web Of Trust. (n.d.). Retrieved 11 2015, 14, from Freenet Project: https://wiki.freenetproject.org/WoT

Frisby, D. (2014). *Bitcoin: The Future of Money?* Unbound.

Galperin, E. (2014, 07 03). *Dear NSA, Privacy is a Fundamental Right, Not Reasonable Suspicion.* Retrieved 11 30, 2015, from Electronic Frontier Foundation: https://www.eff.org/deeplinks/2014/07/dear-nsa-privacy-fundamental-right-not-reasonable-suspicion

Garay, J., Kiayias, A., & Leonardos, N. (2015). The Bitcoin Backbone Protocol: Analysis and Applications. *EUROCRYPT* , 281-310.

Gilson, D. (2013, 01 06). What are Namecoins and .bit domains? *CoinDesk* .

Graeber, C. (2012, 10 18). Megaupload Is Dead. Long Live Mega! *WIRED* .

Greenberg, A. (2014, 03 06). Bitcoin Community Responds To Satoshi Nakamoto's 'Uncovering' With Disbelief, Anger, Fascination. *Forbes* .

Greenberg, A. (2014, 08 28). Creators of New Fed-Proof Bitcoin Marketplace Swear It's Not for Drugs. *WIRED* .

Greenberg, A. (2014, 04 25). Inside DarkMarket: a Silk Road the FBI can't touch. *WIRED* .

Greenberg, A. (2014, 04 24). Inside the DarkMarket Prototype, a Silk Road the FBI Can Never Seize. *WIRED* .

Greenberg, A. (2013, 11 06). 'Silk Road 2.0' Launches, Promising A Resurrected Black Market For The Dark Web. *Forbes* .

Grigg, I. (2004). The Ricardian Contract. *IEEE International Workshop on Electronic Contracting* , 25-31.

Grinberg, R. (2011). Bitcoin: an innovative alternative digital currency. *Hastings Science & Technology Law Journal , 4*, 160.

Hearn, M. (2013, 02 02). *Creating Bitcoin passports using sacrifices.* Retrieved 11 28, 2015, from Bitcoin talk: https://bitcointalk.org/index.php?topic=140711.msg1498806#msg1498806

Hern, A. (2014, 04 30). Silk Road successor DarkMarket rebrands as OpenBazaar. *The Guardian* .

Hughes, E. (1993, 03 09). *A Cypherpunk's Manifesto.* Retrieved 11 16, 2015, from activism.net: http://www.activism.net/cypherpunk/manifesto.html

IEEE Standards Association. (n.d.). *Guidelines for 48-Bit Global Identifier.* Retrieved 11 30, 2015, from IEEE: https://standards.ieee.org/develop/regauth/tut/eui48.pdf

Jøsang. (1999). An Algebra for Assessing Trust in Certification Chains. *Proceedings of the Network and Distributed Systems Security Symposium* .

Jeffries, A. (2011, 10 04). The New Yorker's Joshua Davis Attempts to Identify Bitcoin Creator Satoshi Nakamoto. *Observer* .

Johnson, A., Wacek, C., Jansen, R., Sherr, M., & Syverson, P. (2013). Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* .

Johnson, D., Matthee, K., Sokoya, D., Mboweni, L., Makan, A., & Kotze, H. (2007). *Building a Rural Wireless Mesh Network.* South Africa.

Juels, A., & Brainard, J. (1999). Client puzzles: A cryptographic countermeasure against connection depletion attacks. *Network and Distributed System Security Symposium,* .

Justin Norrie, A. M. (2011, 06 12). Drugs bought with virtual cash. *The Sydney Morning Herald* .

Koetsier, J. (2013, 08 14). GlassUp raised $100K on Indiegogo — but PayPal is refusing to pay up. *VentureBeat* .

Kumar, S. (2015, 06 16). OpenBazaar could be America's most dangerous tech startup. *Fortune* .

La'Zooz Developers. (2015, 06 01). *La'Zooz White Paper.* Retrieved 11 29, 2015, from La'Zooz: http://lazooz.org/whitepaper.html

Lee, A. (2012, 03 09). Interview with nanotube, founder of the Bitcoin-OTC, IRC's Marketplace on Bitcoin, Multisigs and Security . *Privacy Online News* .

Lee, N. (2015, 02 08). Anonymity is dead and other lessons from the Silk Road trial. *engadget* .

Loibl, A. (2014). Namecoin. *Seminar Innovative Internettechnologien und Mobilkommunikation SS2014.* Munich: Technische Universität München.

Lokot, T. (2015, 05 15). PayPal Blocks Donations for Printing Boris Nemtsov's Ukraine War Report. *GlobalVoices* .

Manne, R. (2011, 03). The Cypherpunk Revolutionary. *The Monthly* .

Markoff, J. (2013, 11 23). Study Suggests Link Between Dread Pirate Roberts and Satoshi Nakamoto. *New York Times Bits* .

Markowitz, E. (2013, 12 27). Meet the Black Market That's So Underground You May Never Find It. *vocativ* .

Meyer, M. (2014). Continuous Integration and Its Tools. *Software, IEEE , 31* (3), 14-16.

Mockapetris, P. (1987, 11). *IETF.* Retrieved 11 30, 2015, from Domain Names - Implementation and Specification: https://www.ietf.org/rfc/rfc1035.txt

Morran, C. (2011, 12 05). PayPal Rains On Regretsy's Secret Santa Campaign Over Use Of Wrong Button. *Consumerist* .

Mui, L., Mohtashemi, M., Ang, C., & Szolovits, P. (2001). Ratings in Distributed Systems: A Bayesian Approach. *Workshop on Information Technologies and Systems* , 1-7.

Nakamoto, S. (2008, 10 31). Bitcoin: A Peer-to-Peer Electronic Cash System.

Nakamoto, S. (2010, 02 15). *Strip out unfinished product, review and market stuff.* Retrieved 11 14, 2015, from GitHub: https://github.com/bitcoin/bitcoin/commit/5253d1ab77fab1995ede03fb934edd67f1359ba8

Namecoin Developers. (2014, 04). *Namecoin Specification.* Retrieved 11 30, 2015, from Namecoin: https://wiki.namecoin.info/index.php?title=Namecoin_Specification

Naor, M., & Dwork, C. (1992). Pricing via processing or combatting junk mail. *CRYPTO* .

Narayanan, & Shmatikov. (2009). De-anonymizing Social Networks. *IEEE Symposium on Security and Privacy* , 173-187.

Nightingale, J. (2011, 08 29). *Fraudulent *.google.com Certificate*. (Mozilla) Retrieved 08 30, 2011, from Mozilla Security Blog: https://blog.mozilla.com/security/2011/08/29/fraudulent-google-com-certificate/

Novak, B. (2013, 09 05). *PayPal Freezes Campaign Funds.* Retrieved 11 16, 2015, from Mailpile Blog: https://www.mailpile.is/blog/2013-09-05_PayPal_Freezes_Campaign_Funds.html

OpenBazaar Team. (2015, 03 19). *Evolution Exit Scam Shows Multisig Isn't Enough: We need Decentralization.* Retrieved 11 29, 2015, from OpenBazaar Blog: https://blog.openbazaar.org/evolution-exit-scam-shows-multisig-isnt-enough-we-need-decentralization/

Orlowski, A. (2010, 03 10). Cryptome: PayPal a 'liar, cheat and a thug'. *The Register* .

Orlowski, A. (2010, 03 08). Paypal freezes Cryptome. *The Register* .

Orlowski, A. (2010, 03 16). PayPal says sorry to Cryptome. *The Register* .

Ormsby, E. (2013, 07 10). The outlaw cult. *The Age* .

P4Titan. (2014, 05 17). *A Peer-to-Peer Crypto-Currency with Proof-of-Burn.* Retrieved 11 25, 2015, from Slimcoin: http://www.slimcoin.club/whitepaper.pdf

Pagourtzis, A., Panagiotakos, G., & Sakavalas, D. (2014). Reliable broadcast with respect to topology knowledge. *Distributed Computing* , 107-121.

Pahwa, P., Tiwari, G., & Chhabra, R. (2010, 01). Spoofing Media Access Control (MAC) and its Counter Measures. *International Journal of Advanced Engineering & Application* , 186-192.

Penenberg, A. L. (2011, 10 11). The Bitcoin Crypto-currency Mystery Reopened. *Fast Company* .

Poisot, T. (2015). Best publishing practices to improve user confidence in scientific software. *Ideas in Ecology and Evolution* .

Popescu, C. A. (2004, 04). Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System. *12th International Workshop on Security Protocols* .

Reid, F., & Harrigan, M. (2013). An Analysis of Anonymity in the Bitcoin System. In Y. Altshuler, Y. Elovici, A. B. Cremers, N. Aharony, & A. Pentland, *Security and Privacy in Social Networks* (pp. 197-223). New York: Springer New York.

Rivest, R. L., Shamir, A., & Wagner, D. A. (1996). Time-lock puzzles and timed-release crypto. *Technical Report MIT/LCS/TR-684* .

Sanchez, W. (2014, 05 25). *Ricardian Contracts in OpenBazaar.* Retrieved 11 29, 2015, from GitHub gist: https://gist.github.com/drwasho/a5380544c170bdbbbad8

Shubber, K. (2014, 04 30). DarkMarket Alternative Launches With Friendlier Title 'OpenBazaar'. *CoinDesk* .

Sisario, B. (2012, 01 20). 7 Charged as F.B.I. Closes a Top File-Sharing Site. *New York Times* .

Slepak, G. (n.d.). *okTurtles.* Retrieved 11 30, 2015, from DNSChain + okTurtles: https://okturtles.com/other/dnschain_okturtles_overview.pdf

Smith, Q. (2010, 09 10). PayPal Freezes MineCraft Dev's 600k Euros. *Rock Paper Shotgun* .

Spilman, J. (2013, 12 23). *The Future of Bitcoin Escrow.* Retrieved 11 29, 2015, from opine.me: http://opine.me/future-of-bitcoin-escrow/

Stamm, S., & Soghoian, C. (2012). Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL. *Financial Cryptography and Data Security* , 250-259.

Standifird, S. (2001). Reputation and e-commerce: eBay auctions and the asymmetrical impact of positive and negative ratings. *Journal of Management* , 279-295.

Swartz, A. (2011, 01 06). *Squaring the Triangle: Secure, Decentralized, Human-Readable Names.* Retrieved 11 30, 2015, from Aaron Swartz: http://www.aaronsw.com/weblog/squarezooko

Szabo, N. (2005). *Shelling out – The origins of money.* Retrieved 11 14, 2015, from http://szabo.best.vwh.net/shell.html

Tanase, M. (2003, 03 11). *IP Spoofing: An Introduction*. Retrieved 02 10, 2012, from The Security Blog: http://66.14.166.45/sf_whitepapers/tcpip/IP Spoofing - An Introduction.pdf

The New York Times. (2003, 06 26). Stamps.com Files Breach Of Contract Suit Against eBay. *The New York Times* .

Todd, P. (2014, 10 01). *OP_CHECKLOCKTIMEVERIFY.* Retrieved from BIP: https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki

Todd, P. (2013, 01 05). *Purchasing fidelity bonds by provably throwing away bitcoins.* Retrieved 11 29, 2015, from Bitcoin talk: https://bitcointalk.org/index.php?topic=134827.0

Todd, P. (2012, 07 04). *Trusted identities through provable coin expenditures.* Retrieved 11 28, 2015, from Bitcoin talk: https://bitcointalk.org/index.php?topic=91487.msg1007449

Torpey, K. (2014, 10 14). Bitcoin and Tax Evasion: Bringing 'Under the Table' Income Online. *Blockchain Agenda* .

Ugander, J., Karrer, B., Backstrom, L., & Marlow, C. (2011). The Anatomy of the Facebook Social Graph. *arXiv preprint* .

US Department of Homeland Security. (2009). *Interim Report on the EU Approach to the Commercial Collection of Personal Data for Security Purposes: The Special Case of Hotel Guest Registration Data.*

Wallace, B. (2011, 11 23). The Rise and Fall of Bitcoin. *WIRED* .

Waxman, S. (2011, 12 18). How Pay-Pal Squeezes Merchants with Unfair and Likely Illegal Business Practices . *Alternet* .

Whitney, L. (2011, 02 25). PayPal reinstates Bradley Manning support group account. *CNET* .

Wilcox-O'Hearn, Z. (2001, 10 12). *Names: Decentralized, Secure, Human-Meaningful: Choose Two.* Retrieved 10 20, 2001, from Zooko: http://web.archive.org/web/20011020191610/http://zooko.com/distnames.html

Wile, R. (2013, 12 26). Researchers Retract Claim Of Link Between Alleged Silk Road Mastermind And Founder Of Bitcoin . *Business Insider* .

Winton, R. (2014, 03 07). Deputies: Newsweek Bitcoin story quoted Satoshi Nakamoto accurately. *Los Angeles Times* .

Wired. (2015, 01 01). The Most Dangerous People on the Internet Right Now. *WIRED* .

Wood, G. (2014). Ethereum: A secure decentralized generalised transaction ledger.

Yoo, S. Y. (n.d.). *About NASHX.* Retrieved 11 29, 2015, from NASHX: http://nashx.com/About

Zetter, K. (2013, 11 18). How the Feds Took Down the Silk Road Drug Wonderland. *WIRED* .

Zimmerman. (1995). *PGP: Source Code and Internals.* Massachusetts: The MIT Press.

Zimmerman, M. (2013, 03 18). *In Depth: The District Court's Remarkable Order Striking Down the NSL Statute.* Retrieved 11 24, 2015, from Electronic Frontier Foundation: https://www.eff.org/deeplinks/2013/03/depth-judge-illstons-remarkable-order-striking-down-nsl-statute

Zindros, D. (2014, 03 24). *A decentralized anonymous marketplace*. Retrieved 11 14, 2015, from LiberationTech: https://mailman.stanford.edu/pipermail/liberationtech/2014-March/013304.html

Appendix: Relicensing OpenBazaar under MIT

We are happy to announce that we decided to release OpenBazaar under the MIT license.

OpenBazaar had previously been released under AGPL. After lengthy discussions on the matter, the team arrived at the consensus to move forward with relicensing under MIT.

We believe the MIT license will help us better achieve our vision of building a platform of free trade, where anyone can exchange goods and services via the Internet in an uncensored and privacy-respecting manner. We feel it is crucial to publish our code under a very permissive open source license in order to let people freely build software upon our platform as they see fit.

While we're building the canonical OpenBazaar node software, we believe the power and accessibility of the OpenBazaar network will thrive through the use of a multitude of clients who have access to our secure, pseudonymous, and decentralized system of trade. We expect others to build both commercial and free clients that can access our network, leverage the versatility of the Ricardian contract system, utilize powerful blockchain-based transactions to ensure safety, and support their own systems with our trust and identity service.

By releasing under MIT, we are allowing these things to happen, and we hope this will, in turn, empower the OpenBazaar network to widen.

This is our way of saying we invite you to participate in our platform, whoever you may be. We hope this will encourage developers to start building code which makes use of these virtues and feel free to base it on our own code, so that, together, we can create a world where people can trade with freedom and safety.

Appendix: Creative Commons Attribution 4.0 International Public License