An Exploration of Privacy and Anonymity in Bitcoin

CS203: Network and Distributed System Security

Christopher A. Wood
Department of Computer Science
University of California Irvine
Email: woodc1@uci.edu

Chris H. Vu
Department of Computer Science
University of California Irvine
Email: ChrisHVu@gmail.com

***Abstract*—TODO**

I. Introduction

Electronic commerce would benefit greatly from the existence of a complete secure, private, and anonymous form of digital currency that did not rely on trusted third parties or external financial institutions to manage transactions. Indeed, there has been significant research invested in this problem in recent years resulting in various forms of cryptography-based digital payment systems, or cryptocurrencies for short, such as DigiCash [?], ecash [?], HashCash [?], Namecoin [?], Peercoin [?], Litecoin [?], Ripple [?], and perhaps the most popular variant, Bitcoin [1]. Each of these schemes offer different tradeoffs of security, privacy, and anonymity, and as such have varying popularity among users. However, it is the distribtued, decentralized nature of Bitcoin that has led to its leading popularity to the general public and research communities.

More specifically, Bitcoin is distinguished from other solutions by the fact that it does not rely on trusted third parties. Specifically, the global and publicly accessible ledger which stores records of financial transactions is maintained by a widely distributed, peer-to-peer network of (untrusted) users. Even though each transaction is linked to the public key of a particular user via a digital signature, rather than their real identity, user privacy and anonymity are still at risk because public keys must be owned by specific users. This is true even if a user has multiple public keys and uses them with caution to deter attackers looking for such links. Consequently, the privacy of Bitcoin is an open problem and illustrates the difficulty in achieving a distributed form of cryptocurrency, i.e., one that does not rely on trusted third parties, and one that provides sufficiently useful characteristics such as privacy and anonymity.

Currently, techniques to address such privacy issues with Bitcoin are rather limited and include the use of Chaumian's entirely independent e-cash system [2], which relies on trusted third parties, or Zerocoin [3], which achieves privacy and anonymity at the protocol-level by working in conjunction with Bitcoin, among others. The former is not ideal for several reasons, the most significant of which is that it directly conflicts with the decentralized nature of Bitcoin. That is, reliance on trusted third parties is generally unfavorable if at all feasible. The latter technique is very young, having only been published in the past year, and is just now starting to gain considerable attention. Of course, there exists other academic efforts to further the cause for Bitcoin user privacy, including studies by Ron and Shamir [5] and Androulaki et al. [4], and we can expect to see similar work publishing in the coming years.

In this work we survey Bitcoin and related forms of cryptocurrency with respect to the security, privacy, and anonymity guarantees provided by each. We assess proposed solutions that have and have not been implemented in practice, and offer critical insight into the open problems and difficulties in achieving complete security, privacy, and anonymity with minimal resource consumption (e.g., bandwidth, computational cycles, etc.). We hope that this survey will motivate continued research on this critical problem that has the potential to change financial instutitions and forms of currency for future generations.

TODO: outline the sections here

II. Preliminaries

In this section we introduce some common notation used in the literature involving security, privacy, and anonymity, and then present an overview of the Bitcoin system and underlying protocol for making payments (transactions).

A. *Security Definitions and Adversarial Models*

B. *Bitcoin Basics*

In what follows we distill a description of the Bitcoin system and underlying protocol from [1]; interested readers may acquire more specific details therein if required. To begin, Bitcoin is a distributed, decentralized form of cryptocurrency. Accordingly, this enables all (digitally signed) transactions between two parties to be conducted in a peer-to-peer fashion without the inclusion of a trusted third party, such as a bank or other financial institution. This form of decentralized exchange comes at a price, however, as there must be some way to prevent users from *double spending*, or using the same funds to simultaneously pay multiple parties. Bitcoin achieves this property by relying on its users to construct a history for every transaction that takes place in the system. If a majority of the users accept the validity of a particular transaction, or a set of transactions, the global history of the system is affirmatively updated and "confirmed" via a cryptographic hash digest that all users agree upon. This hash digest, referred to as a hash-based proof-of-work, is what constitutes the validity of the system. By the properties of the underlying hash function, the history of the system cannot be changed without breaking the function (i.e., finding collisions) or re-doing the proof-of-work, which is computationally infeasible for small groups of nodes. Therefore, so long as a majority of the Bitcoin users are honest, the system history is deemed correct and thus all signed transactions are verified, preventing double spending by potentially malicious users participating in direct, peer-to-peer transactions.

Unfortunately, while the above scheme is semantically correct and provides strong guarantees that all financial transactions are valid, there are inherent limitations in the amount of user privacy and anonymity that can be achieved in Bitcoin. In order to adequately define these limitations, we first describe how Bitcoin transactions are generated and how the system history is maintained. For simplicity, consider the scenario in which user A wants to send N BTCs (Bitcoins) to user B. Rather than identify users by name, Bitcoin uses *addresses* that are tied to specific users to use in such transactions. Denote addr_A and addr_B as the addresses of users A and user B used in this transaction. It is often convenient to think of Bitcoin addresses as public keys pk_A and sf_B, and as such there are corresponding private keys, which we denote as sk_A, and sk_B, respectively.

Structurally, a transaction T is a tuple comprised of the *source* transactions which supplied the funds necessary to make this transaction, denoted as source, the (public) address of the recipient, addr_B, the amount of BTCs to send, N, and a digital signature of these three properties, $\mathsf{Sign}_{\mathsf{sk_A}}(\mathsf{source}, \mathsf{pk}_B, N)$. In other words, we have

$$T = (\mathsf{source}, \mathsf{pk}_B, N, \sigma),$$

where $\sigma = \mathsf{Sign}_{\mathsf{sk_A}}(\mathsf{source}, \mathsf{pk}_B, N)$. Note that this signature is embedded in T so that any other Bitcoin user may verify the validity of the content using pk_A. Also note that source need not be a single transaction; user A is free to use multiple transactions in order to fund their transaction to B. In addition to the N BTC transfer from A to B, there is often C BTC amount specified in the transaction for a particular address, where C denotes the amount of change that will be given to this address as a result of the transaction. It is not required that the address to which C is addressed is the same as the address of A, though this often happens in practice. Figure 1 illustrates the input and output relation of our transaction from A to B, and Figure 2 illustrates the steps used in constructing this transaction. Note that, in both cases, source is comprised of two transactions T_1 and T_2, and the resulting transaction is denoted as T_3.

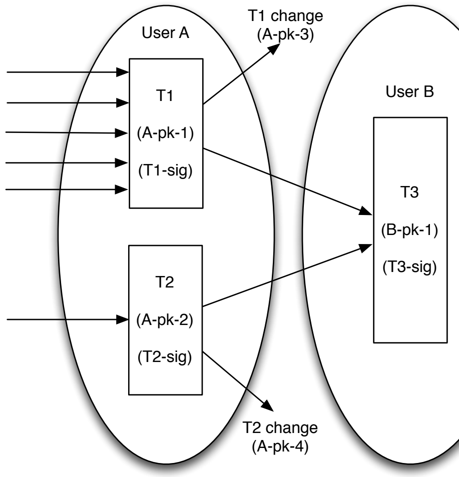

Fig. 1. Visual depiction of the input and output elements of a transaction from user A to user B.

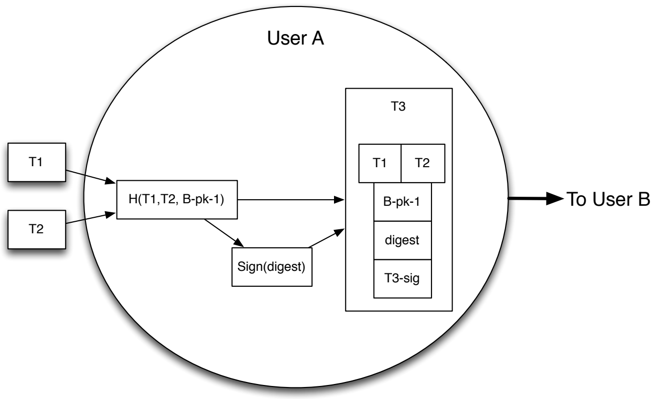

Fig. 2. Visual depction of the steps to create a transaction T_3 from user A to user B using two input transactions, T_1 and T_2.

and digest of the chain. The output digest of this Merkle hash, referred to as the challenge c in the proof-of-work protocol, is then used to find the proof p. Together, c and p have the property that, when concatenated and hashed using a cryptographically strong collision-resistant hash function H, the leading B bits of the output $x = H(c||p)$ are all 0. That is, $x = \{0,1\}^B\{0,1\}^{256-B}$. Given the collision resistant properties of H, finding a valid proof p for the challenge c is comptuationally difficult. Figure 3 illustrates the construction of c and p using a previously confirmed block chain B.

After a transaction has been created, it is broadcasted in the network. In order to prevent double spending, nodes must confirm this transaction and append it to the chain of accepted transactions in the system's history. This procedure is based on the aforementioned proof-of-work, which works as follows. Bitcoin miners will collect unconfirmed transactions into a buffer, along with the longest chain of system-wide accepted transactions, and compute a Merkle hash of the transactions

Once a miner finds a proof, it is broadcasted to the other nodes in the network along with the input transactions used by the miner, who can then easily recompute the challenge c and verify the correctness of p. Once verified, this new transaction "block" is appended to the block chain which the miner used in finding the proof. Figure 4 illustrates a snippet of the block chain, where the challenge c is the digest of the previous block and the proof p are embedded in each block. Miners will continually use the longest block chain to gather and verify

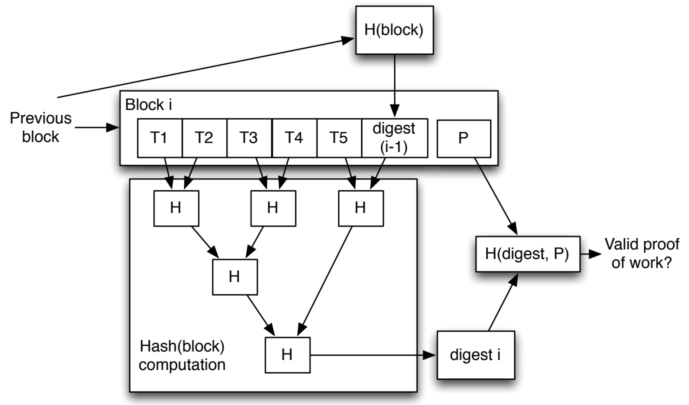

Fig. 3. Proof-of-work computational procedure using the transactions of a block, the digest of the previous block, and the sampled proof p.

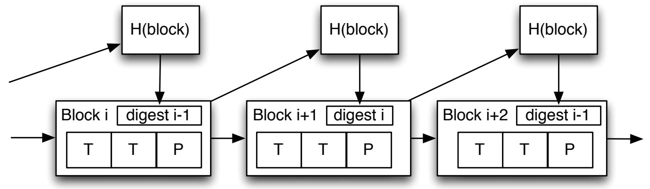

Fig. 4. A snippet of a Bitcoin transaction block chain, illustrating the groupings of transactions into a blocks, the declaration of the proof of the work p, and the digest of the previous block linking the blocks together.

transaction blocks. Since there is a particular subset of BTCs in each transaction that are paid to the miner who provides the proof-of-work for a block containing that transaction, referred to as the transaction fee, miners are financially incentivized to collect more transactions into a block and continually "mine" for valid proofs-of-work.

C. Bitcoin Privacy Measures

As the topic of the survey indicates, Bitcoin has serious privacy flaws. However, there are several standard practices that Bitcoin clients and users are recommended to follow in order to improve their overall privacy and decrease the likelihood of becoming the target of privacy- or anonymity-based attacks. First, clients (and users) should specify *shadow addresses* to collect change from a transaction [**?**]. Such addresses are distinct from the user's address associated used at the time of the transaction. Furthermore, since change need not always be returned to the user who provided the BTC funds, this disjoint address obfuscates the link between the address and the original user, thus helping improve overall privacy. Secondly, it is recommended that all users maintain and continually swap their addresses, and as a result, the underlying public and private key pairs, in order to deter attacks that stem from address re-use. We discuss attacks of this nature in the following sections.

III. Privacy and Anonymity

A. Definitions and Adversarial Models

Privacy refers to the inability of an adversary A to link a user U with any transactions involving U. The Bitcoin block chain links all transactions to addresses, therefore privacy is only preserved if A is unable to link U to any of the addresses involved in any transaction involving U. The inability of A to link U with any address defines the property *address unlinkability*. As shown below, analysis of privacy will be centered around address unlinkability.

In contrast to privacy is *anonymity*, which is captured and quantified with respect to *unlinkability* and *address* or *user profile indistinguishability* [4]. Activity unlinkability refers to the fact that an adversary A should not be able to link any set of transactions to any user. Given a record of all transactions, such as the Bitcoin block chain, A should not be able to identify any user. This is a much stronger security notion as it protects all users from being identified from any set of transactions. Furthermore, since transactions are linked to addresses, A should not be able to identify any user from a given set of addresses; this property is referred to as address unlinkability. Similar to address unlinkability is user profile indistinguishability. In fact, one may view it as an addition to the prior definition in that user profile indistinguishability holds if, given two addresses, an adversary A should not be able to determine if they have a common owner. Put another way, Bitcoin enjoys a measure of profile indistinguishability if an adversary is not able to group the addresses or transactions corresponding based on the original, underlying Bitcoin users.

Adversaries seeking to attack Bitcoin privacy or anonymity clearly have several distinct advantages. First, transactions are publicly broadcast throughout the network, and as users of the Bitcoin system or passive bystanders, they will therefore have access to this log. Additionally, adversaries may have access to the addresses associated with particular vendors that partake in transactions. That is, they may be able to identify and group transactions made by vendors (or other users) whose addresses are acquired via external means. Finally, for practical reasons, we also enforce that all adversaries are computationally bounded, i.e., any algorithm they may run or attack they may leverage must be carried out in polynomial time. Without this restriction it would be possible for an attacker to forge signatures, double-spend confirmed transactions by re-doing proofs-of-work, etc., among other scenarios.

IV. Attacks, Implications, and Proposed Solutions

In this section we survey anonymity attacks on Bitcoin

V. Attack Vectors

Given the design of the Bitcoin system, it may seem surprising that the surface for privacy- and anonymity-targeting attacks is quite large. In fact, there is a large amount of information available to attackers that may be, and has been, exploited to carry out such attacks. Perhaps most fruitful are the *transaction* and *user* graphs that can be constructed via network and transaction analysis. The transaction graph is a directed graph $\mathcal{T}$ with a vertex set $V(\mathcal{T})$ containing

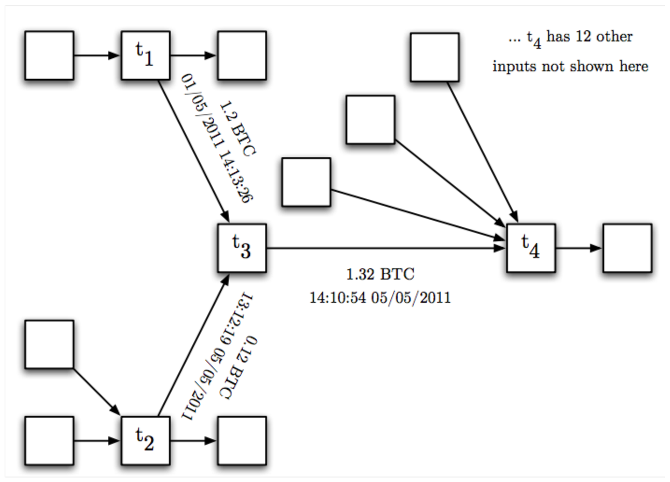

Fig. 5. fig:transaction-graph

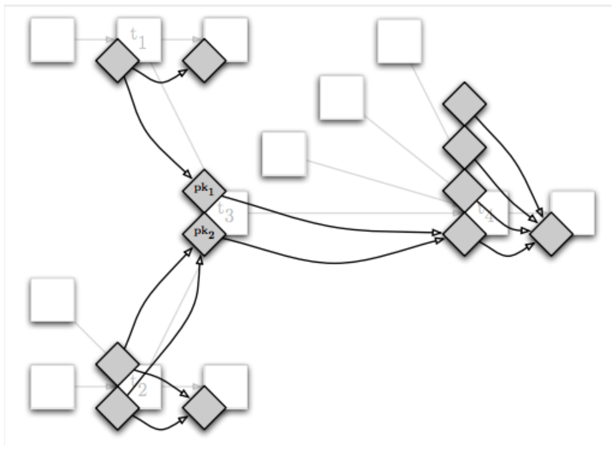

Fig. 6. fig:user-graph

all transactions in the Bitcoin history and edge set $E(\mathcal{T})$ containing directed edges between the source (sender) and target (recipient) for each transaction. An example transaction graph is illustrated in Figure **??**. The user graph is yet another directed graph $\mathcal{U}$ with a vertex set $V(\mathcal{U})$ corresponding to physical users, or entities, partaking in the Bitcoin system and edge set $E(\mathcal{U})$ corresponding to the flow of Bitcoins or funds between two users. An example user graph is illustrated in Figure **??**. With sufficient network analysis (i.e., eavesdropping on Bitcoin traffic in the network), one may also construct a *network address* graph, which is similar to the user graph with the exception that vertices represent physical IPs instead of particular users.

A. Attacks on Privacy

As a currency system, Bitcoin cannot have perfect privacy. Although the information originates outside the Bitcoin network, some address ownership information are public knowledge. For instances, a store needs to have a publicly identifiable address in order to accept payment for goods or services. Users may also disclose address ownership when asking for donations or posting on Bitcoin forums [8]. Large centralized Bitcoin services such as the Mt. Gox exchange service are also able to associate users with addresses as part of their service.

The trivial attack on privacy involve using the Bitcoin block chain to follow all the transactions associated with that address. As user commonly have many addresses, a more sophisticated attack requires the adversary to link the known address with other hidden addresses and then analyze the transactions associated with those addresses. The two major heuristics for linking addresses are *multi-address transactions* and *shadow or change addresses*.

Multi-address transactions are transactions with more than one source. Currently, Bitcoin allows for users to use more than one source address in a transaction, but does not allow multiple users to pay for one transaction. For example, suppose $\texttt{addr}_A$ has 3 Bitcoins (BTC) and $\texttt{addr}_B$ has 2 BTC. The user uses both addresses to pay 4 BTC to $\texttt{addr}_C$ and puts the remainder of 1 BTC to $\texttt{addr}_D$. Only one user can be the input to any transaction, therefore in this example, $\texttt{addr}_A$ and $\texttt{addr}_B$ belong to the same user.

Shadow addresses or *change accounts* are accounts created for change from a transaction. In the transaction above, $\texttt{addr}_D$ is the shadow account that belongs to the same user that controls $\texttt{addr}_A$ and $\texttt{addr}_B$. Although not directly related to the Bitcoin system, the way Bitcoin clients handle shadow accounts can break address indistinguishability [4]. However, because shadow accounts rely on user behavior instead of an inherent property of the Bitcoin system, the shadow account heuristic is not as robust [8].

Using these two heuristics, researchers have been able cluster addresses with a common owner in a user graph where every node is a user and every edge is a transaction [5], [6], [8]. In any node where the user has revealed ownership of an address, the user's privacy has been lost.

Another privacy loss channel is the TCP/IP layer. As previously mentioned, Bitcoin uses a peer-to-peer network to transmit transactions. Many services, such as Bit Faucet, will log and publish the IP address of users of the service. A more active attack would include malicious nodes scanning for Bitcoin clients listening to port TCP/8333 [6] and open a direct connection. While proxy services like Tor can hide outbound connections, an inbound connection will not be obfuscated. By listening to transaction announcements over time, the client that first reports a transaction is the one that initiated it. This allows the malicious nodes to link transactions to IP addresses.

B. Attacks on Anonymity

Researchers attempt to break the anonymity of Bitcoins in an attempt to study stolen Bitcoins [6][8]. In a similar manner to the attack on privacy, a user graph is created.

Intuitively, a successful attack on the anonymity of Bitcoin yields a mapping between Bitcoin addresses, or public keys, to their respective owners. Depending on the success criteria for such an attack, the attacker may seek to find a single mapping for a particular user or, quite oppositely, a mapping for as many users as possible. Accordingly, there has been substantial research investigating the degree to which anonymity is achieved [6], [7], [8], [5], [4]; proposed solutions presented in the literature are discussed in the following section.

Although the original Bitcoin proposal suggests that anonymity, or rather pseudonominity, is possible due to the expected difficulty of associating Bitcoin addresses with their physical owners, there have been ample works attempting to disprove this claim. In fact, one of the earliest anonymity analyses was done by Dan Kaminsky in 2011, a well-known security consultant and researcher. He assessed the anonymity claims of Bitcoin by performing the following experiment: He began by opening up a connection to a large set of peers in the Bitcoin network. Under the assumption that the first appearance of a transaction stems from the original sender (unless an anonymizing layer such as Tor is used to obscure the source), it became fairly easy to combine the peer connections with some elementary traffic analysis to derive a mapping between Bitcoin addresses and IP-layer addresses [9]. While IPs are rarely static due to DHCP and mobile devices, it is not a significant leap to suspect that such knowledge could reveal the underlying user's identity, especially if offline information is used (e.g., if the attacker is able to correlate the IP addresses to users using public forums or other applications).

While using Tor enables one to obfuscate the source of a transaction, it does not solve the problem entirely for several reasons. First, Tor does not provide perfect source secrecy. Tor is inherently susceptible to *end-to-end correlation attacks* in which the attacker controls both ends of a Tor circuit used to send data. Second, Tor was designed for low-latency, high throughput anonymous traffic. As such, it is susceptible to side channel timing attacks [10].

TODO: insert picture of Tor timing attack here

In an attempt to leverage readily accessible information in the Bitcoin system, i.e., not information acquired via traffic analysis or eavesdropping, Reid and Harrigen [6] followed up on Kaminsky's work with an analysis of Bitcoin anonymity. To do so, they constructed the Bitcoin transaction and user graphs (see Section V) as an alternative representation of the flow of information in Bitcoin, and use *passive* analysis techniques to derive user-to-address mappings.

C. Current and Proposed Solutions

The underlying idea behind the Zerocoin protocol is amazingly simple and intuitive, and it stems from the fact that Bitcoin transactions are publicly linked together and therefore susceptible to passive network analysis techniques (as previously discussed above). Zerocoin breaks the links between transactions using cryptographic techniques that yield the same property of Bitcoin transaction chains (namely, the flow of coins and current value of a transaction). Specifically, Zerocoin leverages accumulators, or primitives that enable a party to sign a collection of objects as opposed to a single object and accumulate the result into a single signature digest (in this case, the collection would be the set of previous transactions), and zero-knowledge proofs. The use of these cryptographic techniques is as follows. Whenever a user wants to spend Zerocoins they receive, they accumulate the results of all transactions used to finance the transaction into an accumulator, produce an accumulator witness w (the accumulated total of all bitcons in the transaction minus the value of the Zerocoins to be spent), and finally, generates (zero-knowledge) signature of knowledge π that enables other entities to *publicly* verify that the value in the accumulator was generated by the entity who derived this proof and "owned" the Zerocoins to spend. Zerocoin peers can then verify an accumulated value, representing virtually the same information as a Bitcoin transaction graph, using π.

From an anonymity perspective, Zerocoin ensures that given two Zerocoins and one "spend" coin, one is not able to determine, using publicly available knowledge, which coin was spent (i.e., the adversary's advantage in correctly identifying the spent coin is negligible in the security parameter of the system). With respect to the size of anonymity set, this means that $k = 2$, which is ideal. However, in practice, Zerocoin is inherently limited in that the size of the spender anonymity set can be significaly reduced under certain circumstances. For example, if a user mints and subsequently spends 5 Zerocoins, it is clear to an adversary participating in the system that all 5 coins were spent since they can simply verify the transactions; the adversary does not, however, know which coint was spent in which transaction, implying that the anonymity set cardinality is $k = 5$. Now, if the same user not mints and spends another coin, one would hope that the anonymity set would increase to $k = 6$. Clearly, this is not the case, as the attacker can easily deduce that the last minted coin was spent in the last transaction, and thus $k = 1$. Therefore, the anonymity set cardinality for a single coin is clearly bounded below by the number of coints minted between the time of the candidate coin's mint and spend, and upper bounded by the total number of minted coins in the system. Other anonymity issues of Zerocoin relate to the fact that all minted and spent coins are public knowledge, and that all transaction denominations are also publicly available. These can easily be addressed in practice, however.

The mathematical details of accumulation and witness verification are not discussed for brevity. However, we note that security of both schemes relies on the hardness of the Strong RSA and Discrete Logarithm assumptions. As such, the operations required for accumulation and verification come at the cost of additional computational overhead. Furthermore, Zerocoin requires a preliminary, potentially offline, setup phase in which the parameters for the accumulator and zero-knowledge scheme. Aware of these pitfalls, the authors propose a variety of optimizations for improving Zerocoin performance. Namely, they introduce the notion of accumulator checkpoints, which are essentially timestamps that cryptographically capture the value of an accumulator after each transaction in a recently mined block, thus removing the need for a verifier to recompute the entire accumulator value from scratch upon every invocation.

Secondly, the authors discuss methods in which the zero-knowledge proof scheme can be improved. Their preliminary experiments revealed that the size of proofs often exceeded the MTU of Bitcoin transactions, which prohibits its immediate use since Zerocoin relies on Bitcoin for the initial source of Zerocoin funds (i.e., Bitcoins are used to mint Zerocoins) and transaction block graphs to timestamp the state of the system. Rather than modify this MUT limitation so that proofs can be stored alongside accumulator checkpoints in the transaction block chain, the authors propose to store and access proofs using a distributed hash table or non block-chained storage mechanism in Bitcoin. With respect to the computational cost of proofs, the authors propose a distribtued verification strategy so as to not make every node verify the proof of every new transaction block. Doing so would be a large computational burden on verifiers, which should be quick, as opposed to miners, whose efforts are justified via transaction fees. Internally, the chosen zero-knowledge signature of knowledge scheme consists of n repetitions of the same proof to reduce the probability of forgery to $\mathcal{O}(2^{-n})$, and so by requiring that each node *randomly* selecting a subset of these n proofs to compute and it is possible to achieve the same proof security guarantee with a significantly reduced amount of duplicated effort.

CoinJoin [11] is a method to have multi-user inputs in a Bitcoin transaction. Users agree on a common output size and then combine all their transactions of that size into one big transaction. The aim of this method is to obfuscate which user is the input to an output and prevent the association of multiple addresses in a transaction to one user.

It is not be clear which user is associated to a certain output if multiple transactions with the same sized output are combined together. Any user could have paid for any output. For instance, if Alice has a 1 Bitcoin transaction to Carol and Bob has a 1 Bitcoin transaction to Dave, they can combine their transactions together to create one transaction with two outputs for 1 Bitcoin each. Now it is unclear if Alice paid Carol or Dave. With more simultaneous users, the ability to accurately track transactions would decrease.

CoinJoin allows for multiple inputs per user and combines them all into one transaction. With enough users, the transaction would hide which of the many accounts belong to which user. Although some analysis can be done on the inputs and outputs, it would not have nearly the same accuracy as the current heuristic of one user per transaction.

The simplest implementation of CoinJoin can be done with a "meet-up" server to coordinate transactions. A decentralized version can be used, but the issues with coordinating transactions cause complexity.

The most notable feature of CoinJoin, aside from privacy, is that it works today *without* any modification to the Bitocoin protocol, which is untrue for solutions akin to Zerocoin. The transactions from CoinJoin are identical to normal Bitcoin transactions. A potential development involves moving away from a centralized server that knows all the mappings to one that doesn't and eventually a de-centralized system. It is also unknown how many sessions does the protection of privacy extends. Finally, CoinJoin similar to other financial systems, CoinJoin does not hide the user's IP address; an anonymity layer such as Tor is needed to obfuscate this information from a sender's peers.

VI. Open Problems

TODO: identify issues in the bitcoin protocol that need to be addressed, and propose some solutions here

VII. Privacy-Preserving Alternatives

TODO: summary of related work and different currency systems (e.g., zerocoin) that achieve privacy

References

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Consulted* 1 (2008).

[2] David Chaum. Blind Signatures for Untraceable Payments. *Crypto* 82 (1982).

[3] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. *IEEE Symposium on Security and Privacy* (2013).

[4] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating User Privacy in Bitcoin. *IACR Cryptology ePrint Archive* 596 (2012).

[5] Dorit Ron and Adi Shamir. Quantitative Analysis of the Full Bitcoin Transaction Graph. *IACR Cryptology ePrint Archive* 584 (2012).

[6] Fergal Reid and Martin Harrigan. An Analysis of Anonymity in the Bitcoin System. *Security and Privacy in Social Networks*, Springer New York (2013), 197-223.

[7] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to Better – How to Make Bitcoin a Better Currency. *Financial Cryptography and Data Security*, Springer Berlin Heidelberg (2012), 399-414.

[8] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoyy, Geoffrey M. Voelker, and Stefan Savage. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, ACM (2013).

[9] Dan Kaminsky. Black Ops of TCP/IP Presentation. *Black Hat, Chaos Communication Camp* (2011).

[10] Tor. Bitcoin Wiki. Available online at https://en.bitcoin.it/wiki/Tor. Last accessed: 1/25/14.

[11] G. Maxwell. Coinjoin: Bitcoin Privacy for the Real World. Available online at https://bitcointalk.org/index.php?topic=279249.0. Last accessed: 1/31/14.