An Exploration of Bitcoin Anonymity

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*—Bitcoin is rapidly becoming one of the most popular forms of cryptocurrency available today. Its growth is largely due to its decentralized and distributed design. Unfortunately, however, such a design has serious implications on the anonymity of its users. Motivated by the inherent limitations of user anonymity, this survey contains a detailed overview of all scholarly works studying the anonymity properties of Bitcoin, as well as all known proposed solutions to remediate its shortcomings. Our goal is to illuminate such problems to scholarly researchers in this field and cultivate ideas for new solutions or alternative designs that can help bring cryptocurrencies like Bitcoin one step closer to global acceptance.**

I. Introduction

Electronic commerce would benefit greatly from the existence of a completely secure, private, and anonymous form of digital currency that does not rely on trusted third parties or external financial institutions to manage transactions. Motivated by this ideal type of currency, there have been many research efforts focused on generating suitable cryptography-based digital payment systems, or cryptocurrencies, such as DigiCash [23], E-Cash [22], HashCash [24], Namecoin [25], Peercoin [26], Litecoin [27], Ripple [28], 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 widespread popularity among the general public and research communities.

Bitcoin is distinguished from other cryptocurrencies by the fact that it does not rely on trusted third parties. Specifically, the global and publicly accessible ledger that stores records of all financial transactions, thereby serving as a verifiable history of all Bitcoin funds in circulation, is maintained by a widely distributed, peer-to-peer network of (untrusted) users. Transactions are linked to specific identities, or pseudonyms, via digital signatures used to ensure the validity of each transaction. In this context, it is often convenient to associate specific pseudonomous addresses with a single public and private key pair owned by a particular user. Unfortunately, these pseudonoyms are very weak masks for the underlying user's identity - user privacy and anonymity are still at risk even with the use of such pseudonomous identities. This is true even if a user has multiple pseudononyms and uses them with caution to deter attackers looking for such links. Consequently, user deanonymization is a major problem for Bitcoin users, and there have been several academic efforts to further the cause for Bitcoin user privacy and anonymity, including studies by Reid and Harrigan [7] and Androulaki et al. [5], and we can expect to see similar work publishing in the coming years. Elias [29] also discussed some legal, and moral, aspects of the anonymity, or lack thereof, in Bitcoin. We do not focus on such legality issues here, and merely operate under the assumption that spender anonymity is an ideal property that any currency system should have.

Currently, techniques to address such anonymity issues with Bitcoin are rather limited and include the use of Chaumian's entirely independent e-cash system [3], which relies on trusted third parties, and Zerocoin [4], which achieves privacy and anonymity properties based on strong cryptographic assumptions at the protocol-level by working *on top of* 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. The latter technique is very young, having only been published in the past year, and is just now starting to gain considerable attention [13].

In this work we survey Bitcoin and related forms of cryptocurrency with respect to their privacy and anonymity properties. We analyze proposed solutions and offer critical insight into the open problems and difficulties in achieving perfect privacy and anonymity with minimal resource consumption (e.g., bandwidth, CPU 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.

The rest of this survey is outlined as follows. Section II presents the fundamentals of Bitcoin needed to understand its inherent anonymity limits. Section III discusses the main properties of adversaries in deanonymization attacks, and we give an overview of such attacks and studies on Bitcoin anonymity in Section IV-B. We then follow-up with proposed solutions to improve Bitcoin anonymity in Section V. We then conclude by elaborating on the fundamental problems uncovered throughout the development of this survey and offer a new approach to consider for Bitcoin anonymity in VI, and then briefly discuss related cryptocurrencies and their anonymity properties in Section VII.

II. Bitcoin Preliminaries

To appreciate the simplicity of the anonymity flaws in Bitcoin system, all that is required is an understanding of transaction generation and verification. To that end, we now provide an overview of the Bitcoin transaction system and these two procedures, which are presented with an emphasis on the protocol-level and structural properties that render the currency vulnerability to many deanonymization attacks.

A. Transaction Generation

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 reiterate, 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, which is referred to as the system ledger. If a majority of the users accept the validity of a particular transaction based on provided cryptographic hash digest (to be discussed later), or a set of transactions, it is "confirmed" and the global ledger of the system is affirmatively updated. This validation hash digest, referred to as a hash-based proof-of-work, is the fundamental technique underlying the correctness and accuracy of the ledger contents. By the properties of the hash function, the system ledger cannot be changed without breaking the function (i.e., finding a collision) or re-doing the proof-of-work *faster* than honest nodes working to verify and update the ledger, 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 accepcted as correct and all signed transactions are considered valid, which prevents double spending by potentially malicious users (the ledger can be examined to see if funds have been previously spent elsewhere).

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 ledger is maintained. For simplicity, consider the scenario in which user A wants to send N Bitcoins (BTCs) to user B. Rather than identify users by name, Bitcoin uses *pseudonomous 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, respectively. It is often convenient to think of Bitcoin addresses as public keys pk_A and pk_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, $\mathsf{Sign}_{\mathsf{sk_A}}(\mathsf{source}, \mathsf{addr}_B, N)$. In other words, we have

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

where $\sigma = \mathsf{Sign}_{\mathsf{sk_A}}(\mathsf{source}, \mathsf{addr}_B, N)$. Note that this signature is embedded in T so that any other Bitcoin user may verify the validity of the content using pk_A, which is implicitly tied to the transaction. 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 change address, where C denotes the amount of change, in BTCs, 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 sometimes happens in practice. One final piece of the output of a transaction is a transaction fee, which is a small subset of N that is granted to the mining node that successfully verifies this transaction. We discuss this verification process in the following section.

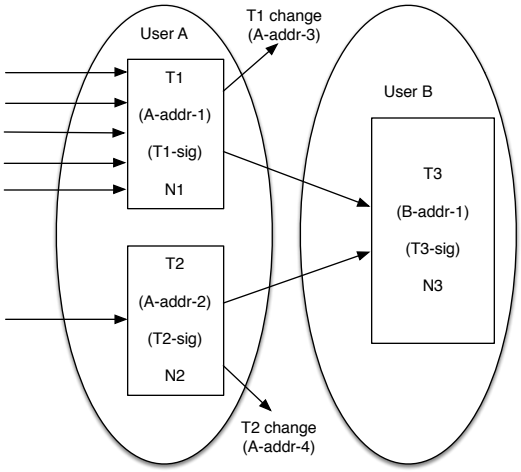

Fig. 1. Visual depiction of the input and output elements of a transaction from user A to user B.

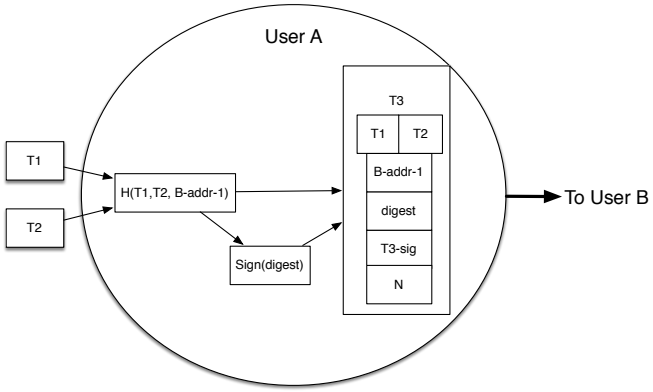

Fig. 2. Visual depction of the steps to create a transaction T_3 from user A to user B using two input transactions, T_1 and T_2.

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, where $N = \mathsf{Val}(T_1) + \mathsf{Val}(T_2)$, and the resulting transaction is denoted as T_3.

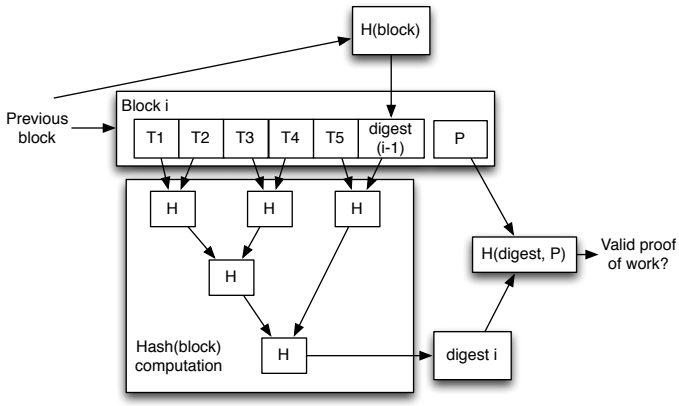

Fig. 3. Proof-of-work computational procedure using the transactions of a block, the digest of the previous block, and the sampled proof p.

B. Transaction Verification

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 longest chain of accepted (confirmed) transactions in the system's ledger. This procedure is based on the aforementioned proof-of-work, which works as follows. Bitcoin miners (i.e., verifying nodes) will collect unconfirmed transactions into a block or buffer, along with the longest chain of system-wide accepted transactions, and compute a Merkle hash of the transactions 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, $x = 0^B\{0,1\}^{256-B}$. Given the collision resistant properties of H, finding a valid proof p for the challenge c is comptuationally difficult, thus making it highly improbable for malicious nodes to alter the contents of the ledger. Figure 3 illustrates the construction of c and p using a previously confirmed block chain B.

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. Other nodes 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 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.

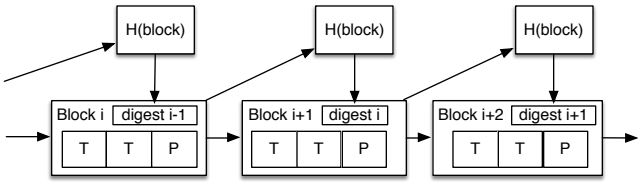

Fig. 4. A snippet of a Bitcoin transaction block chain, illustrating the groupings of transactions into a blocks, the declaration of the proof of the work p, and the digest of the previous block linking the blocks together.

C. Standard Bitcoin Anonymity Practices

As the topic of the survey indicates, Bitcoin has serious anonymity flaws. However, there are several standard practices that Bitcoin clients and users are recommended to follow in order to improve their overall anonymity and decrease the likelihood of becoming the target of deanonymization attacks. First, clients (and users) should specify *shadow addresses* to collect change from a transaction. Such addresses are intended to be distinct from the user's address 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 helps obfuscate the link between the address and the original user, thus helping improve overall anonymity. Secondly, it is recommended that all users continually generate new transaction addresses and corresponding 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. Thirdly, it is often recommended that users connect to the Bitcoin network using an anoymizing layer such as Tor [19], [11] so as to obfuscate their network-layer identities. Beyond these simple practices, there does not exist any other standardized techniques that may be used to help improve user anonymity.

III. Definitions and Adversarial Assumptions

Privacy refers to the inability of an adversary A to link a user U with any transactions involving U. The Bitcoin block chain[1] 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. In contrast to privacy is *anonymity*, which is captured and quantified with respect to *unlinkability* and *address* or *user profile indistinguishability* [5]. Activity unlinkability refers to the property that an adversary A should not be able to link any set of transactions to any user. Given a record of transactions, perhaps acquired from the global ledger, A should not be able to identify any specific user. Note that 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,

[1]We use the term block chain, transaction history, and ledger interchangeably in this survey.

Bitcoin users enjoy a measure of profile indistinguishability if an adversary is not able to group the addresses or transactions based on the underlying Bitcoin users.

An attack on privacy or anonymity is referred to as a deanonymization attack. Adversaries seeking to perform such attacks 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 specific users whose addresses are acquired via external means. However, 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. Deanonymization Attacks and Their Implications

In this section we survey deanonymization attacks on Bitcoin users. Such attacks can be roughly characterized into the following four categories based on the source of information used to carry out the attack:

1) Multi-input transaction attacks,
2) Bitcoin value flow attacks,
3) Implementation-specific side channel attacks that rely on timing and network-layer (e.g., IP addresses) information, and
4) Auxiliary information linkability attacks (e.g., entity-to-address linking using information acquired via external mediums).

Our analysis is based on the small body of work primarily analyzing the first three classes of attacks. The last class is generally not discussed in scholarly work, but rather in popular media. Before discussing any attacks, we first describe the attack vectors leveraged by malicious users and researchers while conducting such attacks.

A. Attack Vectors

Given the design of the Bitcoin system, it may seem surprising that the surface for deanonymization 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. We discuss only a few here. 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 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. The user graph is yet another directed graph $\mathcal{U}$ with vertex set $V(\mathcal{U})$ corresponding to physical users, or entities, partaking in Bitcoin transactions and edge set $E(\mathcal{U})$ corresponding to the flow of Bitcoins or funds between two users. 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. As previously mentioned, other sources of information include auxiliary information gathered from offline sources (e.g., public vendors or merchants) or side channel attacks.

B. Anonymization Attacks and Analysis Techniques

Intuitively, a successful attack on the anonymity of Bitcoin users 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 a mapping for as many users as possible. Accordingly, there has been substantial research investigating the degree to which user anonymity is achieved [7], [8], [9], [6], [5]; proposed solutions presented in the literature are discussed in the following section.

1) Network-Layer Attacks: 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 several works that disprove this claim. One of the earliest anonymity analyses was done in 2011 by Dan Kaminsky, a well-known security consultant and researcher. He assessed the anonymity properties of Bitcoin by performing the following experiment: First, he opened up a direct P2P 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 [10]. While IPs are rarely static due to DHCP-based IP address acquisition and the increasing ubiquity of mobile devices, it is not a significant leap to suspect that such knowledge could reveal the underlying user's identity, especially if offline auxiliary information is used (e.g., if the attacker is able to correlate the IP addresses to users using public forums or other applications). With this observation, a more active deanonymization attack, as carried out by Kaminsky [10], [7], would involve malicious nodes scanning for Bitcoin clients listening to port TCP/8333 and opening a direct connection. While proxy services like Tor can hide outbound connections, an inbound connection will not be obfuscated. Again, by listening to transaction announcements over time, the client that first reports a transaction is the very likely one from whom it was initiated. This allows the malicious nodes to link transactions to IP addresses.

Using Tor enables one to obfuscate the source of a transaction. Unfortunately, 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 (see Figure 11) [21]. Second, Tor was designed for low-latency, high throughput anonymous traffic. As such, it is susceptible to side channel timing attacks [11]. For this reason, the Chaum-like mixnets is often suggested but is not used in practice due to the lack of such services.

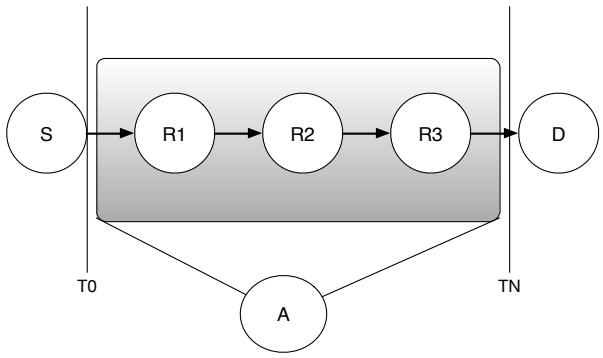

Fig. 5. Illustration of an end-to-end correlation attack on Tor.

2) Protocol-Layer Attacks: Trivial attacks on anonymity involve using the Bitcoin block chain to follow all the transactions associated with that address. As users 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 a single user owns two addresses, addr_A and addr_B, and wishes to make a payment to another user via address addr_C. Also suppose that addr_A has 3 Bitcoins (BTC) and addr_B has 2 BTC. The user uses both addresses to pay 4 BTC to addr_C and puts the remainder of 1 BTC to addr_D. Since only one user can be the input to any transaction, anyone can deduce that addr_A and addr_B belong to the same user. *Shadow addresses* or *change accounts* are accounts created for change from a transaction. In the transaction above, addr_D is the shadow account that belongs to the same user that controls addr_A and 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 [9]. Using these two heuristics, researchers have been able cluster addresses using the transaction graph [6], [7], [9]. In any node where the user has revealed ownership of an address, the user's anonymity is lost.

Motivated by this observation, and relying on the accessibility of publicly accessible offline information relating to the Bitcoin system, Reid and Harrigen [7] 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 IV-A) as an alternative representation of the flow of information in Bitcoin, and used *passive* analysis techniques to derive user-to-address mappings. Such mappings were made by a combination of these dynamic graphs and offline auxiliary information acquired via other means. For instance, 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 [9]. Large centralized Bitcoin services such as the Mt. Gox exchange service are also able to associate users with addresses as part of their service. Reid et al. also pointed out that by colluding the information of multiple accounts that participated in a transaction details about the owner can in fact be recovered.

Shamir et al. [6] also analyzed the transaction graph, deriving some global statistics, including an estimate that 78% of the issued Bitcoins are not circulating, and an in depth analysis of a highly active region in the transaction graph. Their study was not focused on user anonymity, however.

Recently, Androulaki et al. [5] studied the limits of user anonymity through the computational analysis of real-world and simulation-driven data. Their analysis was based on the assumption that users employ several standard practices for making transactions: multi-input transactions are allowed (although discouraged) and fresh shadow addresses are used to collect transaction change. The novelty in this study is the introduction of new quantifiable metrics by which user privacy and anonymity can be assessed. In particular, the notions of *activity unlinkability* and *profile indistinguishability* are defined, and metrics which can be used to measure the degree of unlinkability and indistinguishability are discussed. For completeness, we define these measurements and their metrics here.

Activity unlinkability is a general condition in which an adversary is not able to link two different addresses (address unlinkability) or transactions (transaction unlinkability) to the same user of her choice. It is clear that address unlinkability implies transaction unlinkability, as it is simple to link two transactions together if the addresses bound to these transactions are linked. Standard game-theoretic definitions of system-wide activity unlinkability are also given, which essentially state that an adversary cannot link two activities (addresses or transactions) with non-negligbly better probability than some adversary making uniformly random guesses of such links.

To quantify activity unlinkability by effectively measuring an adversary's success probability in this type of game, the authors propose the following technique: Let $\mathbf{E}$ and $\mathbf{GT}$ be $n \times n$ matrices, where n is the number of addresses in the system according to the transaction history. The i, j entries of $\mathbf{E}$ are 1 if an adversary guesses that addresses i and j are owned by the same user, and 0 otherwise. Similarly, the i, j entries of $\mathbf{GT}$ are 1 if addresses are actually owned by the same user, and 0 otherwise. The error in an adversary's estimate for each address $i = 1, \ldots, n$ is intuitively then the norm of the vector $\mathbf{E}[i, *] - \mathbf{GT}[i, *]$ (i.e., the difference between the guess and the real value), and the maximum error in this estimate, which is analogous to an upper bound on the probability of the adversary's success in the activity unlinkability game, is then the maximum error over all addresses. For the "random guess" adversary, $\mathbf{E}$ is constructed such that each $\mathbf{E}[i, j] = \pi_{i,j}$ if there is some external knowledge available to this attacker and $\pi_{i,j}$ represents the probability, or confidence, based on this apriori knowledge, and $\mathbf{E}[i, j] = \rho + (1 - \rho)/2$ otherwise, where ρ is the fraction of addresses in the transaction history that cannot possibly be linked to multiple users. The latter scenario occurs when the address is owned by some user who only has one address. Given these definitions, the overall advantage

of an adversary in this activity linkability game is then the difference of the two error estimates.

Profile indistinguishability is a condition in which an adversary is not able to reconstruct the profiles (collection of representative addresses or transactions) of *all users* who participated at some point in the block chain. The fact that all profile indistinguishability holds if and only if the adversary cannot reveal profiles for all users makes it an intuitively stronger notion of privacy than address unlinkability, which only requires some information linkage with respect to a single user. Analogous to the above definition, a system deployment is said to enjoy profile indistinguishability if the global transaction history does not provide any non-negligible advantage in deriving the correct (address or transaction) profiles when also given the total number of users in the system.

The metrics used to quantify profile indistinguishability are related to those used to quantify address unlinkability, though somewhat more contrived. In particular, they are based on the definition of some similarity function Sim, which takes as input the adversary's estimate of the user profiles, $\mathbf{E}_{\mathsf{prof}}$, and the actual user profiles $\mathbf{GT}_{\mathsf{prof}}$. If we define the output of this function given inputs $\mathbf{E}_{\mathsf{prof}}$ and $\mathbf{GT}_{\mathsf{prof}}$ as $\mathsf{Sim}(\mathbf{E}_{\mathsf{prof}}, \mathbf{GT}_{\mathsf{prof}})$, then the advantage of an adversary in the profile indistinguishability game is simply $\mathsf{Sim}(\mathbf{E}_{\mathsf{prof}}, \mathbf{GT}_{\mathsf{prof}}) - \mathsf{Sim}(\mathbf{E}^{R}_{\mathsf{prof}}, \mathbf{GT}_{\mathsf{prof}})$ (where the second term denotes the estimate made by a "random guess" adversary). Computing Sim relies on *normalized mutual information* (NMI) and *adjusted mutual information* (AMI). According to [16], [17], NMI increases in magnitude as the similarity of the two input profile groupings increases. Conversely, AMI decreases in magnitude (towards 0) as the input of the adversary's guessed profile grouping appears to be uniformly random (i.e., it is highly similar to a random grouping). Details of these computations are omitted for brevity.

Given these privacy measurements and supporting metrics, the authors then quantitatively assessed user privacy in a simulation of Bitcoin transactions in a university setting. This experiment used the simulator to gather the "ground truth" information about users and their addresses in a system, performed various machine learning behavior-based analysis techniques such as k-means clustering (KMC) and Hierarchichal Agglomerative Clustering (HAC) to formulate an adversary's guess in deanonymizing clients, and then calculated the resulting success (adversarial advantage) in the address unlinkability and profile indistinguishability games. The interesting results for these calculations after varying the fraction of "privacy-aware" users, i.e., those that avoid multi-input transactions and use fresh shadow addresses, and the total number of users in the simulation are shown in Table I. Observe that in nearly all cases the adversary has a clearly non-negligible advantage when given some apriori knowledge about the users in the network. These results highlight the extent to which anonymity is not upheld in Bitcoin.

V. Proposed Solutions

Generally speaking, solutions to address the anonymity issues in Bitcoin roughly fall into one of the following three categories:

1) Use an anonymity layer to obfsucate the source IP address of Bitcoin users,
2) Use mixing services to unlink transactions from their owners, and
3) Modify the Bitcoin protocol using cryptographic techniques that improve guarantees of anonymity.

A. Anonymity Layers

The basic design and use of mix networks is credited to Chaum and dates back to more than three decades [2]. The essential idea of a mix network is illustrated in Figure 8. Senders wrap their message in layers of encryption using the public keys of randomly selected mixing service nodes (mix node), who then receive a set of encrypted messages, decrypt and shuffle the messages, and then send them to another mix node or the intended recipient. Clearly, the role of each mix node is to hide the source of original messages by obfuscating their flow through the network. To initiate an untraceable message (email) in the original design, which was intended to be used for untraceable email, Alice encrypts a message to Bob with Bobs public key, appends Bobs address, and then encrypts the entire message with the mix servers public key. The mix server receives the message, decrypts it with the mix servers private key, and then sends the message to Bob after waiting some non-deterministic amount of time. The order of outgoing messages is hidden by the fact that the mix server receives messages that are uniformly sized and rearranges them in lexicographical order. Bob receives the message from the mix server and decrypts it with his private key.

A "cascade" or series of mixes (see Figure 8) can increase the secrecy of the messages as all of them must be compromised before a message can be traced. The protocol is similar to a single mix with the exception that messages are encrypted with all the mix servers private keys.

Since mixnet services are not generally available, Bitcoin users are often suggested to use Tor [19] to achieve similar anonymity properties with strong (but not perfect) protection against eavesdropping and traffic analysis attacks. To use Tor with Bitcoin, a separate software client must be installed which establishes circuits from a pool of anonymizing routers through which all Bitcoin traffic can flow. To redirect this traffic, the user's Bitcoin client must be configured to send all traffic through the local Tor proxy using the SOCKS protocol (i.e., 127.0.0.1:9050, where 9050 is the default Tor port number). Despite the popularity of Tor as a network anonymity layer, it is intended for low-latency, high-throughput traffic that is susceptible to targeted attacks by active adversaries [20], [21].

B. Coin Mixing Services

Mixing services in the context of Bitcoin are distinct from traditional mixnets. In particular, mixers are used to anonymously collect or aggregate coins from clients, mix them internally (i.e., among several different Bitcoin addresses), and then redistribute coins back to the sources in equal denominations after some predefined amount of time. The role of a mixing service is illustrated in Figure 7. The obvious goal is to break, to the maximum extent possible, the link between the change address and signing address, thus improving the sender's overall anonymity. Also, it is important to emphasize the element of time with mixing services; contrary to anonymity layers like Tor, which were designed for

TABLE I. ADVERSARIAL ADVANTAGE WHEN GIVEN PARTIAL KNOWLEDGE OF USERS IN THE (SIMULATED) BITCOIN NETWORK.

	100 (50%)	200 (0%)	200 (50%)	200 (100%)	400 (50%)
$\mathbf{Link}_A$	0.91 ± 0.01	0.90 ± 0.01	0.91 ± 0.01	0.92 ± 0.01	0.93 ± 0.01
$\mathbf{Prof}_A^a$	0.76 ± 0.01	0.87 ± 0.01	0.79 ± 0.01	0.70 ± 0.01	0.80 ± 0.01

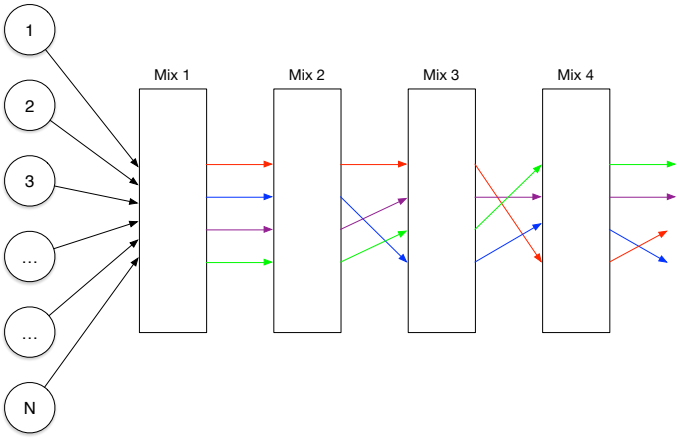

Fig. 6. Mixnet cascade.

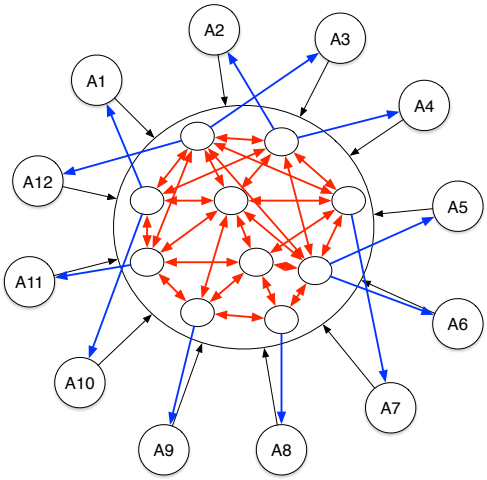

Fig. 7. Bitcoin fund mixing service node.

low-latency, high throughput anonymous connections and are therefore susceptible to timing side channel attacks, mixing services that introduce a mandatory delay prohibit (to some degree) such attacks.

Unfortunately, Bitcoin mixing services naturally require a certain level of trust on behalf of the user. In particular, there is nothing in the current Bitcoin system that prevents a malicious mixing service from aggregating a large collection of coins and then abandoning their duty as a mixing service, thereby stealing all of its client's coins. Of course, such financial losses can be minimized if the user only sends funds to the mixing services in small amounts, but it does not eliminate the problem; users would then need to sequentially small small amounts to the mixer, thus drastically increasing the payout time. Therefore, we identify mixing *theft* as a primary threat to users, as well as deanonymization, since the mixing service naturally learns the client's input transaction-signing and output change address.

Given these problems with simple mixing services in the context of Bitcoin, there has been several attempts to modify their deployment and usage in order to avoid the aforementioned threats. One work by Barber et al. [8] explored such a modification under the assumption that there does not exist an underlying trust architecture that can be relied upon to prevent the attacks; their conclusion was that the only feasible alternative is to use a *fair exchange protocol* between the users and mixing service to ensure that all input coins are eventually paid back. Unfortunately, their proposed protocol must be integrated with the existing Bitcoin protocol - it is not natively supported. For completeness, we highlight the main features of their approach here. The fair exchange protocol between a sender and mixing service, denoted as parties A and B, respectively, requires three types of transactions:

1) Commitment transactions to commit a party to a coin exchange.
2) Refund transactions to refund a party's committed coins at a future date in case the other party aborts the protocol.
3) Claim transactions to claim the other party's committed money.

After establishing a set of shared secrets using offline or online techniques akin to Diffie Hellman key exchange, A and B then begin the fair exchange protocol, which proceeds as follows:

1) B generates a committment and refund transaction, with the help of A, and broadcasts both transactions. The committment transaction is constructed such that its respective funds can be redeemed using either the corresponding refund transaction (after the protocol "times out", or one party fails to complete their duties within a pre-defined) or a to-be-generated claim transaction. [2]
2) Simultaneously, A generates a committment and refund transaction, also with the help of B, and broadcasts both to the network.
3) After both parties are committed to the exchange, they must claim their coins. In order to do so fairly, B first claims A's coins using A's refund transaction, and by doing so, enables A to claim B's coins through B's refund transaction. This refund process works as follows: B modifies the output of A's refund script (a part of the transaction block) to be redirected to B's recently generated address, and also modifies the input of the script to reveal the "secret" values that enables A to modify her refund transcation. Once B publishes his claim transaction, thus taking A's funds,

[2]The refund transactions are constructed with the output coins being redirected back to their original owner so that the coins are not lost in the event that the protocol does not complete.

A can then receive the secret values from B's claim transaction to modify B's refund transaction output to point to her own recently generated address. A then broadcasts her claim transaction to claim B's coins, thus completing the exchange.

Note that if a protocol "timeout" occurs, then the refund transactions were never properly modified by both parties to steal the other party's coins, and no exchange took place.

With an existing public key infrastructure and means to securely transfer the secrets used in the transaction generation, this protocol will prevent malicious mixing services from stealing user's coins. However, it requires modifications to the underlying Bitcoin protocol that aren't readily supported. Most importantly, it requires that transactions support the notion of timelock (i.e., points in time when they can no longer be changed).

Motivated by the desire to achieve the same level of trust without modifying the underlying Bitcoin protocol, Bonneau et el. [14] recently proposed Mixcoin, a natively-supported protocol for building accountability into mixing services. The principal idea used to achieve mixing accountability is to rely on simple economics, rather than cryptographic solutions, to ensure that mixing services will benefit more from honest participation in mixing rather than from theft. Properly aligned financial incentives ensure that theft is not likely to happen, else the mixing service's reputation is permanently destoryed. However, as the authors note, there is no (protocol-level) mechanism that can be used to gaurantee that a mixing service is not linking client input and output addresses; software implementations of mixing services may purposefully or accidentally store this information and thus render it susceptible to exposure if compromised. Such means of acquiring auxiliary information are outside the scope of this survey and we do not discuss them further.

The key insight to achieving business-incentivized accountability through financial incentives is that mixing nodes will provide warranties stating that they claim to complete a mixing operation prior to some agreed-upon deadline. If the mixing node operates dishonestly or steals the client's funds, the client then simply broadcasts the warranty to the network, which is signed by the mixing node using a long-term private key, which can then be verified by all nodes in the network using the corresponding public key. The net effect is that no more (intelligent) clients will do business with the mixing node. With an appropriate fee selection it is therefore economically sensible to behave honestly rather than attempt to steal funds since mixing nodes are paid for their efforts by means of a mixing fee.

The fundamental protocol used to achieve this accountability operates as follows:

1) A client A generates a tuple $T = (v, \mathsf{addr}_{\mathsf{in}}, \mathsf{addr}_{\mathsf{out}}, t_1, n)$, where v is the BTC amount to send, t_1 is a time by which A will transfer the v funds, and n is a random nonce. This tuple is then sent to the mixing node M.
2) If M accepts the parameters, a fresh escrow address $\mathsf{addr}_{\mathsf{escrow}}$ is generated at which the funds can be acquired by M, a deadline t_2 by which the funds will be transferred, and a mixing fee rate p. These parameters are then signed using the long-term key K_M to generate a warranty, and both are then sent to A.
3) If A accepts t_2 and p, A transfers the funds to $\mathsf{addr}_{\mathsf{escrow}}$.
4) After receipt of the funds at $\mathsf{addr}_{\mathsf{escrow}}$, M will return the same amount of funds, minus the agreed-upon transaction fee, to $\mathsf{addr}_{\mathsf{out}}$.
5) Both parties delete the parameters used in the transaction.

Observe that A may choose to opt out of the protocol at any point prior to the transfer of the funds by A. If M does not act honestly, A simply broadcasts the signed warranty to other nodes so that they may see A was cheated. It is also important to note that, just as in traditional mixing schemes, these mixing nodes may be used sequentially to increase anonymity through multiple mixing rounds. This is done by using the output address of the i-th round of mixing as the input address of the $(i+1)$-th round, thereby creating a chain through which the client's funds flow through potentially separate mixes.

While simple, there are several underlying issues that must be addressed in practice to ensure anonymity:

- All Bitcoin chunk sizes should be uniform.
- The mixing service should choose escrow addresses at random when transferring funds from a client's input address to their specified output address.
- Transaction fees should be randomized (from an appropriate source of randomness, such as a Beacon [15] or some PRG computed based on the transaction block chain).
- There should never be two outstanding warranties for the same input address issues at the same time.
- Clients should not publicize their freshly created input addresses until mixing has initiated, else a malicious party could perform a DoS attack (due to the above requirement) by requesting a warranty for some observed or predicted input address.
- Clients should use separate mixes when aggregating their mixed funds to make a transaction, or else their anonymity set reduces to the intersection of the anonymity set for each of the mixed funds.

A brief analysis of the anonymity properties provided by Mixcoin was also presented. Of the most trivial observations are that mixing message replay is impossible due to the single-warranty issuance property stated above, and also that blocking client-to-mix or mix-to-client transactions are not feasible under the standard assumption that the majority of Bitcoin users and miners are not malicious. They also derived a value for the maximum mix delay δ_{max} (i.e., the time between receiving input funds and transferring output funds) to optimize the client's anonymity set. If a mix receives input transactions at a stable rate where Q chunks are mixed in a single round (i.e., the total number of funds mixed is equal to $Q \times v$, where v is the baseline uniform mixing input), then the anonymity set of each client is roughly $Q(\delta_{max} - w + 1)$,

since fresh input and output keys are used for each of the Q chunks for a particular user, and where w is the minimum difference between t_2 and t_1 (observe that the final mix delay, uniformly generated, thus falls in the interval $[w, \delta_{max}]$). For $98 \leq Q \leq 4096$, it was found that $\delta_{max} = 7$ since the anonymity set grows by

$$1 + \frac{\log_2 Q(\delta_{max} - w + 1)}{1 + w + \frac{\delta_{max} - w}{2}}.$$

It should also be clear that the anonymity set of a particular chunk *is not* the size of the fund chunks in the mixing node. Rather, it is the size of the set of fund chunks that were mixed at the same time, i.e., those which overlap in the same round within the mixing service. For this reason, longer mixing delays or multiple mixing nodes in sequence benefit all mixing participants, and it is conjectured that these delays will converge to reasonably steady values in practice. Simulations performed by the authors showed that as the number of overlapping mixing rounds for a set of m mixes increases, where each round transfers Q chunks, the statistical distance between the PDF of the anonymity set of each chunk rapidly approaches the uniform distribution. Put simply, this means that out of all Qr chunks (where r is the number of rounds), the probability that an adversary can correctly associate a chunk with its input client is roughly $(Qr)^{-1}$.

There is one more subtle yet important issue concerning the use of these mixing services. Recall that the input from one client A_1 may be used to provide output funds for another client A_2, and each of these inputs are associated with their providers, A_2 will learn that A_1 has interacted with the mixing service. Therefore, to deanonymize clients, an active adversary can request the mixing of many funds in hopes of probabilistically determining whether or not a particular user A has used a mixing service. This type of deanonymization attack is probabilistic because the input provided by A is only routed to the adversary's output according according to some probability distribution. Fortunately, since each use of the mixing service costs a certain fee, an attacker will, on average, have to invest a great deal of funds in hopes of learning whether or not A has interacted with the mixing service. Therefore, as previously mentioned, clients should use separate mixes whenever possible if their mixed funds (chunks) are to be used together in a single transaction. Beyond the immediate fact that her anonymity set reduces to the intersection of the anonymity set for each mixed chunk, it also reduces the likelihood that compromised or malicious mixes can deanonymize the client.

Finally, we note that there are several protocol-level side channel attacks that can be leveraged by an adversary to decrease a client's anonymity, as follows:

- Passive observance of a client requesting a certain amount of mixed funds and then witnessing that same amount appear in a single transaction in the future effectively provides a probabilistic link from the later transaction to the observed client.
- Bitcoin transactions (and therefore, mixed funds) are timestamped, and the provenance information of transaction block chains can therefore be used to link clients to future transactions if both the client who originally owned the unmixed funds and the time at which those funds were mixed is known. This can be avoided by paying with funds that were mixed at the same time, paying to re-mix all chunks every time a payment is to be made, and time-delayed continual mixing, among other possibilities. Observe that the client will incur additional mixing costs for the latter two solutions, which may or may not be acceptable given the level of anonymity desired, and so the client will have to intelligently choose when to mix funds at her own discretion.

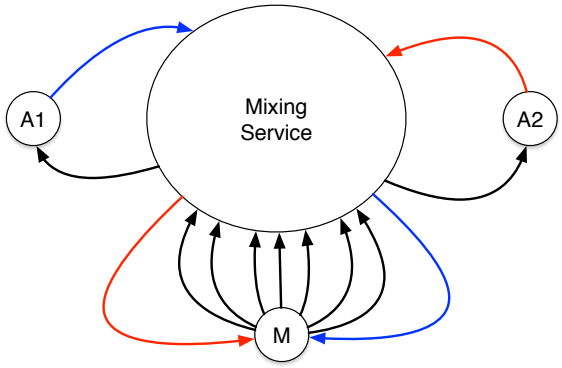

Fig. 8. Visual depiction of an active deanonymization attack in Mixcoin.

CoinJoin [12] is another method inspired by mixing services to support multi-user inputs in a Bitcoin transaction without sacrificing user anonymity. In CoinJoin, 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 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 proportionally.

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 increased anonymity, is that it works *without* any modification to the Bitocoin protocol, which is untrue for solutions like the fair-exchange mixing service and Zerocoin (see the following section). 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 decentralized system. It is also unknown how many sessions does the protection of privacy extends. Finally, 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.

C. Cryptographically Enhanced Anonymization Techniques

We now describe solutions that build upon the Bitcoin protocol to improve sender anonymity. To our knowledge, the earliest such attempt (which is actually quite recent) is Zerocoin, a distributed, anonymous cryptocurrency using Bitcoin as a form of backing currency. At a high level, 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). Ultimately, Zerocoins are not identified by public keys as in the case of the backing Bitcoin currency; rather, they are identified by a commitment C to randomly generated secrets (i.e., a serial number S associated with the coin and "opening value" r) only maintained by the owner of the coins. To do this, Zerocoin leverages cryptographic 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 primitives as follows. Whenever a user wants to spend Zerocoins they receive associated with some committment C, the owner must reveal S publicly and then provide a proof that they are also in possession of r capable of opening some commitment $C_i \in \{C_0, \ldots, C_{n-1}\}$, the set of all committments made in the system. The owner uses r to generate a (zero-knowledge) "signature of knowledge" π that is equivalent to stating that the owner can open some committment (coin) $C_i \in \{C_0, \ldots, C_{n-1}\}$, without revealing which coin they actually own (hence, the signature of knowledge is actually a zero-knowledge proof). Technically, to generate this proof, the owner accumulates the results of all coins used to finance the transaction into an RSA-based accumulator, produces an accumulator witness w (the accumulated total of all bitcoins in the transaction minus the value of the Zerocoins to be spent), and finally, generates π that enables other entities to *publicly* verify that the value in the accumulator was generated by the entity who derived this proof and was in possession of the Zerocoins coins (committments) necessary to fund this transaction. 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). 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 coin was spent in which transaction, implying that the anonymity set cardinality is 5. Now, if the same user not mints and spends another coin, one would hope that the anonymity set would increase to 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 the cardinality of the anonymity set is exactly 1. Based on this observation, the anonymity set cardinality for a single coin is clearly bounded below by the number of coins 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 out of the scope of this survey. 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 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 backing 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 MTU 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.

Pinocchio Coin [13] is a very new attempt to optimize the Zerocoin protocol. Whereas Zerocoin uses an accumulator based on the Strong RSA assumption and proofs founded on double discrete logarithms, Pinocchio Coin uses elliptic curves and bilinear pairings to achieve virtually the same behavior with significantly less overhead. As such, Pinocchio Coin follows the same process as Zerocoin for the setup, mint, spend, and verify protocols: The setup phase involves creating a suitable pairing-friendly elliptic curve in the security parameter and then configuring the parameters for the Pinocchio proof system. As previously stated, minting, spending, and verification are analogous to the Zerocoin protocol with the exception that proofs are generated without the need to accumulate past commitments (though the proof generation still requires all commitments $C_0, \ldots, C_{n-1}$ as input to ensure correctness).

The Pinocchio proof of work was originally designed as a generic proof of work to be used in applications such as cloud computing. The system takes a set of operations and converts them into proof of work that can be seen as a zero knowledge proof. The primary advantage of using Pinocchio Coin over Zerocoin is that the size of each proof is significantly smaller; Pinocchio proofs are less than 400 bytes as opposed to Zerocoin's 50kB proofs, which are well within the Bitcoin MTU limit. One potential disadvantage of using Pinocchio Coin is that there is no security analysis as of yet [13]. Futhermore, although Pinocchio serves as an efficient general proof compiler, it is unknown whether or not specialized systems such as ZKPDL [18], a general zero-knowledge proof system intended to be used with e-cash technologies, would exhibit better performance in the Zerocoin framework.

VI. Fundamental Problems and New Ideas

Based on our review of the literature, it is quite clear that Bitcoin is far from being a completely anonymous form of currency. Before first attempting to derive such a currency, it is important to explicitly capture what it means for a currency system to provide its users anonymity.Ideally, an anonymous form of (digital) currency would have the following properties:

1) Users should be able to validate the monetary value associated with a coin (or bill) without learning any additional information. This is akin to validating the monetary value of a $20 dollar bill found on the ground - one cannot determine (with reasonable effort) where the bill came from or who previously owned said bill.
2) The system should enable users to possess coins without learning who owns what coins. Using the dollar metaphor, this is again similar to the case where the any person is able to acquire and spend bills as they see fit, but the US treasury and other people are not able to see what bills are owned by a specific user unless they attempt to make a payment.
3) Users should be able to prove ownership of a coin without revealing their identity.

Enumerating the goals in this manner makes it quite easy to see why the Zerocoin extension to Bitcoin is such a natural solution to anonymous currency. Zerocoin supports all three properties through cryptographic techniques such as accumulators and zero knowledge proofs of knowledge, which, when used according to the protocol, enable coins to be spent anonymously. Recall that Zerocoin achieves this through the use of a virtual "bulliten board" of transaction committments, where the value of a transaction is verified by churning out the result of an accumulator and the proof of ownership is done via a zero-knowledge proof of knowledge of the secret information needed to reveal *one* committment on the bulliten board. However, the development of Pinocchio Coin [13] to enhance the Zerocoin protocol with alternative cryptographic primitives that yield the same behavior is an indication that this anonymity comes at a price. Furthermore, since these techinques are more sophisticated than native Bitcoin protocol "operations," the chance of an implementation or engineering error leading to an exploitable side channel through which to conduct a deanonymization attack is greatly increased.

Such risks may be unavoidable, however, as even the next-best, low-cost, natively supported solutions for anonymity in Bitcoin (e.g., mixing services) do not provide perfect anonymity. In particular, while mixing serices may bend and break the links in transaction graph, masking the original generator of a particular transaction, compromised services can still lead to client input and output information leakage if implemented poorly.

To date, we believe that Zerocoin is the most popular form of completely anonymous currency proposed. While it is still too early to tell whether or not Zerocoin will see widespread adoption. However, reports indicate that the first Zerocoins will become available in May 2014 [30]. Until then, it may still be worthwhile to investigate alterantive means for creating anonymious currency. We now propose one such idea: Suppose that instead of transactions being associated with a single sender, they were associated with a group of k users. That is, each transaction was signed by k distinct users using some aggregate signature technique [31]. Among this set of k users there exists exactly one user who is the generator of the transaction, but the members of the group cannot identify said user. This property can be achieved by having the single generating user U send the transaction T to each member of the group using the Dining Cryptographer's Protocol [32]. This protocol enables U to multicast T to the other $k-1$ members of the group with *complete information-theoretic anonymity*. When T has been retrieved by all k users, they would then engage in an aggregate signature computation over T before broadcasting it to the rest of the network. Alternatively, one single user may inject a *group signature* in T where the verification key is associated with the k users. This approach has the drawback that k-size groups must be predefined and keys must be carefully managed.

Observe that, if the transaction T is distributed throughout the k-group using an anonymity-preserving protocol, then the anonymity set of the transaction is effectively k, the set of all users who participate in the aggregate (or group) signature. Based on prior transaction graph analysis attacks, this would help circumvent deanonymization attacks on single users based on transaction clustering. Of course, this property comes at the cost of efficiency, as the communication and computation complexity of performing an anonymous multicast and aggregate signature is quite high depending on the particular techniques used. If, however, nodes can spare these cycles and network

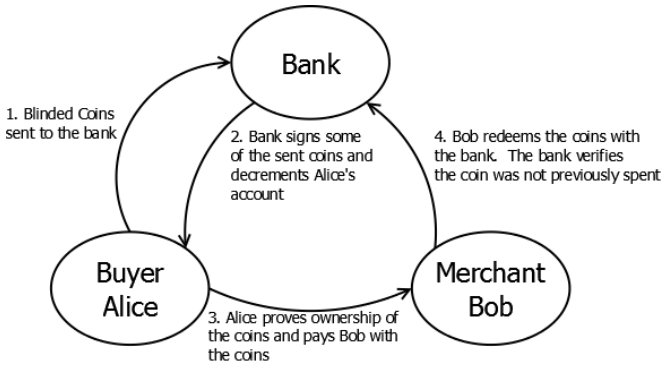

Fig. 9. Digicash protocol.

resources, this may be a viable solution worth pursuing.

VII. Related Cryptocurrencies

Although Bitcoin is inarguably the most popular cryptocurrency to date, it is by no means the first such technology proposed. Other popular variants include DigiCash [23], E-Cash [22], HashCash [24], Namecoin [25], Peercoin [26], Litecoin [27], and Ripple [28]. In this section we briefly describe some of these variants with respect to their anonymity properties. Our discussion begins with E-Cash [22].

Chaum, Fiat, and Naors offered two methods to create untraceable electronic cash. Both methods require a central bank organization. In all methods, the coins that are issued by the bank are blinded in a way to prevent the bank from identifying the source of honestly spent coins. The first method uses untraceable coins in that each transaction is of set amounts. Alice creates random commitments and sends them to the bank.The bank uses a subset of those commitments to create cryptographic coins by signing them. The bank then sends those coins to Alice after decrementing the amount from Alices account. Everyone can verify the coins structure and the banks signature. Alice can then spend those coins to pay Bob. To do so, Alice sends Bob the information on the coin as well as a zero-knowledge proof that Alice initiated the creation of the coin. When Bob tries to redeem the coin with the bank, Bob sends both pieces of information to the bank. The bank will not have enough information to identify the coin came from Alice unless Alice tries to double spend. In that case, there is a high probability one of the challenges in the zero-knowledge proofs from the merchants will be different allowing the bank to reconstruct Alices identity. This method underpins the DigiCash system.

The second method outlined is “untraceable checks”. The idea builds upon the untraceable coins idea by giving Alice has a roll of coins. Alice can spend them by indexing the roll of coins by the purchase amount and revealing that to the merchant. Alice is then refunded the rest of the amount by the bank by creating a separate transaction to pay herself.

Compact E-Cash builds upon the framework that Chaum creates with Untraceable Electronic Cash by creating a coin that can be used repeatedly a limited number of times. The system is comprised of a withdrawal protocol, a spending protocol, and a double spending check. The withdrawal protocol involves the user creating a private key as well as the coins serial number and blinding value. The bank then signs the values and decrements the users account accordingly. The spending protocol has the user give the merchant the coins serial number, the merchants random number challenge, and a double-spending value based on the users private key, the random number challenge, and the blinding value. The user also gives the merchant two non-interactive proofs. The first proof is that the committed coin was signed by the bank. The second proof verifies that the coins serial number and double spending number correspond to the commitment as well. The merchant then reveals this information to the bank for payment. In order to protect the users privacy, the serial number that is revealed to the merchant is encrypted using the users private key. In order to prevent the user from cheating, if the same coin is used over the limited number of uses, the bank is able to infer the secret key of the user, decrypt the serial number of the coin, and identify the user. This also means that the user can re-use coins a number of times without the bank being able to identify the user. The Compact E-Cash system allows for users to create coins that can be used multiple times. The coins do not reveal the spending habits of the user if the user does not try to cheat and re-use the same coin too many times.The drawbacks are that there must be a central bank to issue coins and verify transactions. Also, each time a coin is used is a separate transaction even if the user spends multiple coins with the same merchant.

In both systems, privacy is maintained by blinded coins. If the user uses the system honestly and does not double spend, then the bank cannot reveal the identity of the user based on the coins used. Furthermore, because the coins are indistinguishable, the bank is unable to create profiles based on the location the coins were spent.

A mature form of the untraceable coins is the Mondex Smart Card. The bank issues secure cards embedded with an integrated circuit. The cards store value on the card. The value is incremented when money is transferred onto the card and decremented when the card is used for payment or deposit. In essence, if Alice pays Bob 5 coins, Alices card will destroy 5 coins and Bobs card will create 5 coins. This makes the individual coins impossible to track. However, both Alice and Bobs cards will have a transaction ledger that ties Alices and Bobs cards to the transaction. Similar to a Bitcoin address, the card provides limited privacy as it separates the user from the coins being spent. However, merchants are able to create purchasing profiles based on transaction logs and the cards internal identification. Furthermore, the system does not prevent merchants from sharing purchasing information with the issuing bank to match card identification numbers with an identity. In this way, the Mondex Smart Card system is less anonymous than other cryptocurrencies and does not provide any guarantees of privacy.

HashCash was originally designed to stop the waste or abuse of internet resources. HashCash requires the actor, Alice, to provide a proof-of-work before allowing an action such as sending an email. The idea is to add a cost to the action to prevent abuse. HashCash now has been incorporated into Bitcoin and similar cryptocurrency systems in the mining process as the proof-of-work protocol, but is not used as a currency system by itself. There are three protocols and two public variables in the original interactive HashCash system.

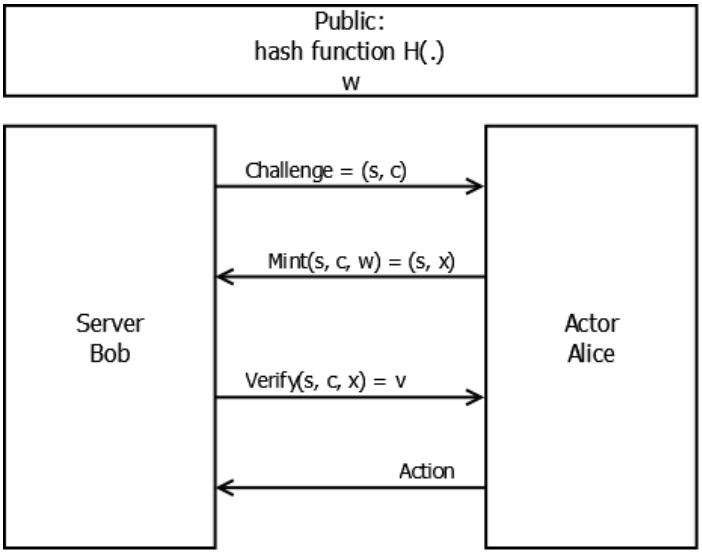

Fig. 10. Hashcash protocol.

The public variables are a hash function, $H(\cdot)$, and the length parameter w. In the Challenge protocol, the server, Bob, sends Alice the service name s and a random challenge value c. Alice runs the Mint protocol to run a brute-force search for the value x. The hash of $s|c|x$, where $|$ represents concatenation, creates a value with w leading zeroes. After Alice sends x to Bob, Bob runs the Verify protocol to verify Alices work. If Verify succeeds, Alice is allowed to perform her action using the requested service. HashCash improvements include replacing the service name and challenge with a fixed output string and requiring Alice to find a collision. The proof-of-work can be considered a currency as it is used to create a transaction. If used in this way, HashCash does provide slightly better privacy than Bitcoins. Proofs-of-work are public knowledge, but HashCash does not have a ledger and therefore new nodes do not have access to the history of work.

Other alternative cryptocurrencies are based off the Bitcoin system, but only Zerocoin increases privacy. The other systems do not improve or reduce the privacy as compared to Bitcoin. Litecoin is the most similar to Bitcoin as it only differs in a few aspects. The first difference is that Litecoin uses scrypt instead of SHA-256, which is used in Bitcoin. Second, Litecoin has faster block times which decreases the time needed for a transaction confirmation. Finally, Litecoin increases the total number of coins available to be mined. Peercoin combines the HashCash proof-of-work method of mining coins with a second method, proof-of-stake. Proof-of-stake uses the notion of coin age. The coin age of a wallet is the sum of the lengths of time since a coins previous transaction. If Alice received 4 coins 2 days ago, then her wallets coin age would be 8 coin-days. In proof-of-stake, one hash per unspent wallet-output is created each second. Whereas the proof-of-work protocol uses a fixed hash target, proof-of-stake uses a variable hash target that scales inversely with coin age. If Alice finds an accepted value, she creates a transaction paying herself and is awarded one percent of her transaction as a reward. Being that her coins were used in a transaction, the mining process resets the coin age of Alices wallet. Namecoin is a cryptocurrency based off Bitcoin in the sense that the coins are used to register and transfer domain names. Namecoin is designed as a decentralized DNS. The ledger, which already contains coin transactions, will also contain DNS transactions such as the creation or purchase of a domain name and the associated IP address.

Unlike the previous cryptocurrencies which are based on the transference of value, Ripple is based on the transference of debt. Ripple is comprised of two parts. The first part involves a web-of-trust between nodes. Alice determines how much she trusts Bob and Charlie and then extends the maximum debt she is willing to accept from them. Suppose Charlie owes Alice 5 coins and Alice wanted to pay Bob 5 coins, but Bob only trusts Alice with a debt of 2 coins. If Bob trusts Charlie with a debt of 3 coins, Alice can send an IOU for 2 coins from her and an IOU for 3 coins from Charlie. Bob receives an IOU totaling 5 coins from which he can collect from Alice and Charlie at a later date or use to make payments to someone else.

The second part of Ripple is to facilitate transactions where no web-of-trust exists. This is done through gateway nodes. Gateways act as publically known trusted nodes analogous to banks. If Alice wanted to pay Bob 5 coins, but Bob only trusts a gateway, then Alice can send 5 coins to the gateway and the gateway will credit Bob 5 coins. Unlike the first part of Ripple, gateways use stores of value instead of debt. Ripple is as private as Bitcoins as transfers of debt or credit between addresses is public knowledge, but no registry ties addresses to people. The exception is if Ripple deals with real world currency. Ripple is currency agnostic and therefore gateways are free to use real world currency as an exchange medium. In this case, gateways require personal information from the user and are able to link users to addresses and transactions.

VIII. Conclusion

In this survey we provided an extensive study of the anonymity properties of Bitcoin. We discussed all known attacks published in scholarly literature, elaborated on possible solutions to improve anonymity, and also discussed potential avenues for future work. We also briefly discussed related cryptocurrencies to provide some reference against which Bitcoin can be compared.

References

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

[2] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24(2) (1981), 84-90.

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

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

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

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

[7] 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.

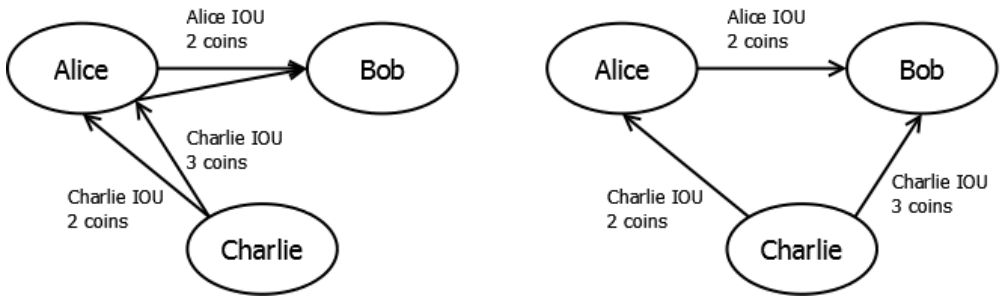

Fig. 11. Ripple debt network.

[8] 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.

[9] 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).

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

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

[12] 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.

[13] George Danezis, Cédric Fournet, Markulf Kohlweiss, Bryan Parno. Pinocchio Coin: Building Zerocoin from a Succinct Pairing-Based Proof System. In *Proceedings of the First ACM Workshop on Language Support for Privacy-Enhancing Technologies*, ACM (2013).

[14] Joseph Bonneau, Arvind Narayan, Andrew Miller, Jeremy Clark, and Joshua A. Kroll. Mixcoin - Anonymity for Bitcoin with accountable mixes. Preprint available online at: http://cs.umd.edu/~amiller/mix.pdf.

[15] NIST Randomness Beacon. Available online at: http://www.nist.gov/itl/csd/ct/nist_beacon.cfm. Last accessed: 2/5/14.

[16] N. X. Vinh, J. Epps, and J. Bailey. Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary? In *26th Annual International Conference on Machine Learning (ICML)* (2009).

[17] N. X. Vinh, J. Epps, and J. Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. *Journal of Machine Learning Research* (2010).

[18] Sarah Meiklejohn, C. Chris Erway, Alptekin Køupcøu, Theodora Hinkle, and Anna Lysyanskaya. ZKPDL: A Language-Based System for Efficient Zero-Knowledge Proofs and Electronic Cash. *USENIX Security Symposium*, (10) (2010).

[19] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router. *Naval Research Lab Washington DC* (2004).

[20] Steven J. Murdoch and George Danezis. Low-Cost Traffic Analysis of Tor. *IEEE Symposium on Security and Privacy* (2005).

[21] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. Timing Attacks in Low-Latency Mix Systems. *Financial Cryptography, Springer Berlin Heidelberg* (2004).

[22] David Chaum, Amos Fiat, and Moni Naor. Untraceable Electronic Cash. *Proceedings on Advances in Cryptology, Springer-Verlag New York, Inc.* (1990).

[23] Digicash. Available online at: http://www.digicash.com/. Last accessed: 3/1/14.

[24] Adam Back. Hashcash A Denial of Service Counter-Measure. (2002). Available online at: http://www.hashcash.org/papers/hashcash.pdf

[25] Namecoin. Available online at: https://www.namecoin.org/. Last accessed: 3/1/14.

[26] Peercoin - The Secure and Sustainable Cryptocoin. Available online at: http://www.peercoin.net/. Last accessed: 3/1/14.

[27] Litecoin. Available online at: https://litecoin.org/. Last accessed: 3/1/14.

[28] Ripple - The Future of Payments. Available online at: https://ripple.com/. Last accessed: 3/1/14.

[29] Matthew Elias. Bitcoin: Tempering the Digital Ring of Gyges or Implausible Pecuniary Privacy. *Available at SSRN 1937769* (2011).

[30] Andy Greenberg. Bitcoin Anonymity Upgrade Zerocoin To Become An Independent Cryptocurrency. *Forbes - Security* (2014). Available online at: http://www.forbes.com/sites/andygreenberg/2014/01/13/bitcoin-anonymity-upgrade-zerocoin-to-become-its-own-cryptocurrency/. Last accessed: 3/4/14.

[31] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, Brent Waters. Sequential Aggregate Signatures and Multisignatures without Random Oracles. *Advances in Cryptology - EUROCRYPT 2006, Springer Berlin Heidelberg* (2006), 465-485.

[32] David Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology* (1.1) (1988), 65-75.