# Evaluating User Privacy in Bitcoin

Elli Androulaki[1], Ghassan O. Karame[2], Marc Roeschlin[1], Tobias Scherer[1], and Srdjan Capkun[1]

[1] ETH Zurich, 8092 Zuerich, Switzerland
elli.androulaki@inf.ethz.ch, romarc@student.ethz.ch,
schereto@student.ethz.ch, capkuns@inf.ethz.ch
[2] NEC Laboratories Europe, 69115 Heidelberg, Germany
ghassan.karame@neclab.eu

**Abstract.** Bitcoin is quickly emerging as a popular digital payment system. However, in spite of its reliance on pseudonyms, Bitcoin raises a number of privacy concerns due to the fact that all of the transactions that take place are publicly announced in the system.
In this paper, we investigate the privacy guarantees of Bitcoin in the setting where Bitcoin is used as a primary currency for the daily transactions of individuals. More specifically, we evaluate the privacy that is provided by Bitcoin *(i)* by analyzing the genuine Bitcoin system and *(ii)* through a simulator that faithfully mimics the operation of Bitcoin in the context where Bitcoin is used for all transactions within a university. In this setting, our results show that the profiles of almost 40% of the users can be, to a large extent, recovered even when users adopt privacy measures recommended by Bitcoin. To the best of our knowledge, this is the first work that comprehensively analyzes, and evaluates the privacy implications of Bitcoin. As a by-product, we have designed and implemented the first simulator of Bitcoin; our simulator can be used to model the interaction between Bitcoin users in generic settings.

## 1 Introduction

Bitcoin [14] is an emerging digital currency that has, as of September 2011, approximately 60,000 users [8]. Shortly after its downfall in November 2011, Bitcoin has quickly recovered [1] and is currently integrated across a number of businesses [2] and exchange markets (e.g., [3]).

Bitcoin is a Proof-of-Work (PoW) based currency that allows users to generate digital coins by performing computations. Bitcoin users execute payments by digitally signing their transactions and are prevented from double-spending their coins (i.e., signing-over the same coin to two different users) through a distributed time-stamping service [14]. This service operates on top of the Bitcoin Peer-to-Peer (P2P) network and ensures that all transactions and their order of execution are available to the public. To strengthen the privacy of its users, Bitcoin users participate in transactions using virtual pseudonyms—referred to as *Bitcoin addresses*. Generally, each user has hundreds of different Bitcoin addresses that are all stored and transparently managed by its client.

In spite of the reliance on pseudonyms, the public timestamping mechanism of Bitcoin raises serious concerns with respect to the privacy of users. In fact, given that

Bitcoin transactions basically consist of a chain of digital signatures, the expenditure of individual coins can be publicly tracked [24]. Note that any decentralized currency is likely to have similar limitations.

Until now, the privacy threats and their implications in Bitcoin have not been studied. Given the increasing popularity of Bitcoin, we argue that a thorough evaluation of user privacy in Bitcoin should precede the large scale deployment of Bitcoin across businesses. In this work, we evaluate the privacy that is provided by Bitcoin *(i)* by analyzing the genuine Bitcoin system and *(ii)* through a novel simulator that mimics existing Bitcoin clients implementation in the special case where Bitcoin is used as the primary currency within a university setting. We believe that, while Bitcoin is still not being used in such environments, it is highly likely that a university environment would be one of the first candidate environments to "host" Bitcoin if it were to be adopted as an alternative currency in the near future. Finally, we discuss possible measures that can be used to enhance the privacy of users in Bitcoin. To the best of our knowledge, this is the first work that comprehensively analyzes, and evaluates the privacy guarantees of Bitcoin. More specifically, our contributions in this paper can be summarized as follows:

- We investigate the privacy-enhancing measures that are used in current Bitcoin implementations.
- We design and implement the first simulator of Bitcoin; we rely on our simulator to faithfully emulate the functionality of Bitcoin, while providing us with the corresponding "ground truth" that we use to evaluate the accuracy of our evaluation.
- We empirically evaluate the privacy guarantees of Bitcoin in a realistic university setting. Our results show the profiles of 40% of the university users can be constructed using behavior-based clustering techniques with 80% accuracy, even in the case when users manually transfer their Bitcoins among their addresses in an attempt to enhance their privacy.
- We explore possible measures that can be used by Bitcoin developers to enhance the privacy of users in Bitcoin.

The remainder of this paper is organized as follows. In Section 2, we present a background on Bitcoin. In Section 3, we introduce metrics that we use to measure privacy in Bitcoin. In Section 4, we present our Bitcoin simulator and our evaluation results. In Section 5, we discuss the implications of our findings and we explore possible countermeasures for enhancing privacy in Bitcoin. In Section 6, we overview the related work and we conclude the paper in Section 7.

## 2 Background on Bitcoin

Bitcoin is a decentralized P2P payment system [14] that relies on PoW. Payments are performed by generating *transactions* that transfer Bitcoin coins (BTCs) between Bitcoin users. Users participate in transactions using virtual pseudonyms—referred to as *Bitcoin addresses*. Generally, each user has hundreds of different Bitcoin addresses that are all stored and managed by its (digital) *wallet*. Each address is mapped through a

transformation function to a unique public/private key pair. These keys are used to authorize the transfer of the ownership of BTCs among addresses.

**Transactions:** Users transfer coins to each other by issuing a transaction. A transaction is formed by digitally signing a hash of the previous transaction where this coin was last spent along with the public key of the future owner and incorporating this signature in the coin [14]. More specifically, a transaction taking place between two addresses $a_{\mathsf{S}}$ and $a_{\mathsf{R}}$ has the following form: $\tau(a_{\mathsf{S}} \rightarrow a_{\mathsf{R}}) = \{\text{source}, B, a_{\mathsf{R}}, \mathsf{SIG}_{\text{sk}_{a_{\mathsf{S}}}}(\text{source}, B, a_{\mathsf{R}})\}$. Here, $\mathsf{SIG}_{\text{sk}a_{\mathsf{S}}}$ is the signature using the key $\text{sk}a_{\mathsf{S}}$ associated with $a_{\mathsf{S}}$, $B$ is the amount of BTCs transferred, and source is a reference to the most recent transaction that $a_{\mathsf{S}}$ acquired the $B$ BTCs from. After their creation, Bitcoin transactions are released in the Bitcoin network. Once the validity/correctness of $\tau$ is *confirmed* (as described later in this section), $a_{\mathsf{R}}$ can subsequently use this transaction as a reference to spend the acquired BTCs. Consequently, Bitcoin transactions form a public record and any user can verify the authenticity of a BTC by checking the chain of signatures of the transactions in which the BTC was involved. Note that transaction fees may be included within any transaction transferring BTCs from one address to another [4].

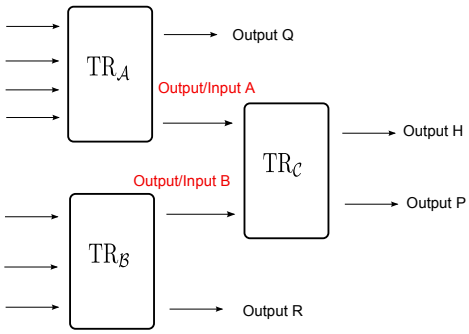


**Fig. 1.** Example of referencing input/output in transactions.

In the case where $a_{\mathsf{S}}$ needs to spend a value that exceeds the maximum value of a BTC that it possesses, then its Bitcoin client will automatically combine a number of its BTCs (and their corresponding transactions) as multiple inputs of the same outgoing transaction. We analyze the impact of "multi-input" transactions on the privacy of users in Bitcoin in Section 4.1. Figure 1 depicts an example of multiple-input transactions. Here, $\text{TR}_{\mathcal{C}}$ uses two inputs that are outputted by $\text{TR}_{\mathcal{A}}$ and $\text{TR}_{\mathcal{B}}$.

**Shadow Addresses:** In the current implementation of Bitcoin, a new address—the "shadow" address [5]—is automatically created and used to collect back the "change" that results from any transaction issued by the user. Besides the reliance on pseudonyms, shadow addresses constitute the only automated mechanism through which Bitcoin strengthens the privacy of its users.

**Confirmation of Transactions:** As mentioned before, transactions are broadcasted in the Bitcoin network and are subject to validity checks by users in the system. Valid transactions are included by a special type of users, the *miners*, in Bitcoin *blocks* that are also broadcasted in the network. More specifically, to generate a new block, miners must find a nonce value that, when hashed with additional fields (i.e., the Merkle hash of all valid and received transactions, the hash of the previous block, and a timestamp), results in a value below a given threshold. If such a nonce is found, miners then include it in a block thus allowing any entity to verify the PoW. Upon successful block generation, a miner is granted 50 new BTCs. This provides an incentive for miners to continuously

support Bitcoin. Table 2 in Appendix A shows the information included in Bitcoin block number 80,000 as reported in the Bitcoin block explorer [6]. The resulting block is forwarded to all users in the network, who can then check its correctness by verifying the hash computation. If the block is deemed to be "valid"[3], then the users append it to their previously accepted blocks, thus growing the Bitcoin block *chain*. Bitcoin relies on this mechanism to resist double-spending attacks; for malicious users to double-spend a BTC without being detected, they would not only have to redo all the work required to compute the block where that BTC was spent, but also they need to recompute all the subsequent blocks in the chain.

## 3 Modelling Privacy in Bitcoin

In this section, we introduce our adversarial model and we define a number of metrics that can be used to quantify privacy in Bitcoin.

### 3.1 Adversarial Model

We observe the public log of Bitcoin, denoted by $\mathsf{pubLog}$, within a period of time $\Delta t$. During this period, $\mathrm{n_U}$ users, $\mathrm{U} = \{u_1, u_2, \ldots, u_{\mathrm{n_U}}\}$, participate in $\mathsf{pubLog}$ through a set of $\mathrm{n_A}$ addresses: $\mathsf{A} = \{a_1, a_2, \ldots, a_{\mathrm{n_A}}\}$. We assume that within $\Delta t$, $\mathrm{n_T}$ transactions have taken place as follows: $\mathrm{T} = \{\tau_1(\mathsf{S}_1 \rightarrow \mathsf{R}_1), \ldots, \tau_{\mathrm{n_T}}(\mathsf{S}_{\mathrm{n_T}} \rightarrow \mathsf{R}_{\mathrm{n_T}})\}$, where $\tau_i(\mathsf{S}_i \rightarrow \mathsf{R}_i)$ denotes a transaction with (unique) ID number $i$ and $\mathsf{S}_i$ and $\mathsf{R}_i$ denote the sets of senders' addresses and recipients' addresses, respectively.

We assume that the adversary $\mathcal{A}$ is motivated to acquire information about the addresses/transactions pertaining to all or a subset of Bitcoin users. As such, $\mathcal{A}$ does not only have access to $\mathsf{pubLog}$, but is also *part of the Bitcoin system* and can also incur one or more transactions through Bitcoin. Furthermore, we assume that $\mathcal{A}$ can have access to the (public) addresses of some vendors along with (statistical) information such as the pricing of items or the number of their clients within a specified amount of time. We, however, assume that $\mathcal{A}$ is computationally bounded and as such cannot construct ill-formed Bitcoin blocks, double-spend confirmed transactions, or forge signatures, etc.

Throughout this paper, we consider the "privacy" measures adopted by existing Bitcoin clients. Namely, we assume that *(i)* new shadow addresses are used to collect change that results from issued transactions, *(ii)* users own many Bitcoin addresses, and *(iii)* users are encouraged to frequently change their addresses (by transferring some of their BTCs to newly created addresses); this conforms with the current practices adopted in Bitcoin.

### 3.2 Quantifying Privacy in Bitcoin

In what follows, we define activity unlinkability and profile indistinguishability and we provide metrics to appropriately quantify these notions. In Section 4.3, we provide an empirical evaluation of these metrics by means of our Bitcoin simulator.

[3] That is, the block contains correctly formed transactions that have not been previously spent, and has a correct PoW.

We define these notions using privacy games in which a challenger $\mathcal{C}$ issues a challenge to $\mathcal{A}$, who in turn may use the public pubLog to respond to the challenge.

**Activity Unlinkability:** We distinguish between address and transaction unlikability.

*Address Unlinkability*. Informally, address unlinkability refers to the fact that $\mathcal{A}$ should not be able to link two different addresses pertaining to the same user.

We construct the $\mathrm{AddUnl}$ game in Bitcoin as follows. We assume that originally $\mathcal{A}$ does not know pubLog but has a priori knowledge of a subset of addresses that belong to some users in the Bitcoin system. These addresses could, for example, pertain to the set of vendors known by $\mathcal{A}$ in the system[4]. Let $\overline{g_i}$ denote the set of addresses that are known a priori to $\mathcal{A}$ and pertaining to $u_i$. We denote by $\overline{G} \leftarrow \{\overline{g_1} \cup \overline{g_2} \cup \cdots \cup \overline{g_{\mathrm{n_U}}}\}$. Given $\overline{G}$, $\mathcal{A}$ can construct an $\mathrm{n_A} \times \mathrm{n_A}$ matrix, $\mathrm{PriorKnowledge}$, as follows.

$$\mathrm{PriorKnowledge}(i,j) = \begin{cases} 0, \text{ if } a_i \in \overline{g_x} \text{ and } a_j \in \overline{g_y},\ x \neq y \text{ and } x, y \in [1, \mathrm{n_U}], \\ 1, \text{ if } a_i \in \overline{g_x} \text{ and } a_j \in \overline{g_x},\ x \in [1, \mathrm{n_U}], \\ \frac{1}{2}, \text{otherwise.} \end{cases}$$

$\mathrm{PriorKnowledge}$ denotes the estimate of $\mathcal{A}$ with respect to the association of different addresses in A in the case where $\mathcal{A}$ does not know pubLog, or other information besides that reflected in her prior knowledge $\overline{G}$.

$\mathcal{C}$ then sends pubLog to $\mathcal{A}$. Given pubLog and $\overline{G}$, $\mathcal{A}$ constructs another $\mathrm{n_A} \times \mathrm{n_A}$ matrix, $\mathrm{Estimate}$, as follows:

$$\mathrm{Estimate}(i,j) = \begin{cases} 0, \text{ if } a_i \in \overline{g_x} \text{ and } a_j \in \overline{g_y},\ x \neq y \text{ and } x, y \in [1, \mathrm{n_U}], \\ 1, \text{ if } a_i \in \overline{g_x} \text{ and } a_j \in \overline{g_x},\ x \in [1, \mathrm{n_U}], \\ p_{i,j}, \text{ otherwise,} \end{cases}$$

where $p_{i,j}$ denotes $\mathcal{A}$'s estimate on the probability that addresses $a_i$ and $a_j$ belong to the same user or not. As such, $\mathrm{Estimate}$ denotes $\mathcal{A}$'s estimate on the address association, which includes her prior knowledge G and additional information that $\mathcal{A}$ could extract from pubLog (e.g., by means of clustering, statistical analysis, etc.)

We measure the success of $\mathcal{A}$, $\mathrm{Adv_{unl}}$, in the $\mathrm{AddUnl}$ game by evaluating the additional knowledge that $\mathcal{A}$ can derive from pubLog given the initial knowledge of $\overline{G}$. Namely, $\mathrm{Adv_{unl}}$ is computed as follows: [5]

$$\mathrm{Adv_{unl}} = \frac{\|\mathrm{PriorKnowledge} - \mathrm{Genuine}\|}{\|\mathrm{Genuine}\|} - \frac{\|\mathrm{Estimate} - \mathrm{Genuine}\|}{\|\mathrm{Genuine}\|},$$

where $\|X\|$ denotes the norm II of vector $X$, and $\mathrm{Genuine}$ is an $\mathrm{n_A} \times \mathrm{n_A}$ matrix that denotes the genuine assignment matrix between addresses in the Bitcoin network. Here $\frac{\|\mathrm{Estimate-Genuine}\|}{\|\mathrm{Genuine}\|}$ and $\frac{\|\mathrm{PriorKnowledge-Genuine}\|}{\|\mathrm{Genuine}\|}$ denote the (normalized) distance of the estimate of $\mathcal{A}$ from the genuine address association matrix.

[4] Recall that the addresses of Bitcoin vendors are generally public.

[5] Note that the smaller is the distance of $\mathcal{A}$'s estimate from the truth, the greater is the success of $\mathcal{A}$.

*Remark 1.* Similarly, *transaction unlinkability* refers to the fact that $\mathcal{A}$ should not be able to link two transactions that pertain to the same user's wallet. Similar to the $\mathrm{AddUnl}$ game, we can define a variant game, $\mathrm{TranUnl}$, to capture the transaction unlinkability property. Note that the advantage of $\mathcal{A}$ in the $\mathrm{AddUnl}$ and the variant $\mathrm{TranUnl}$ game is comparable. We therefore do not consider the advantage of $\mathcal{A}$ in the $\mathrm{TranUnl}$ game separately in our evaluation.

**User Profile Indistinguishability:** The profile indistinguishability property denotes the fact that the adversary should not be able to group addresses or transactions corresponding to Bitcoin users.

We define the profile indistinguishability property by means of the $\mathrm{ProfInd}$ game as follows. Here, $\mathcal{C}$ sends to $\mathcal{A}$ the number of users $\mathrm{n_U}$ participating in $\mathsf{pubLog}$[6]. $\mathcal{A}$ responds with $\mathrm{n_U}$ (non-overlapping) sets of clusters $G = \{\mathsf{g}_1\}_{i=1}^{\mathrm{n_U}}$ each representing the set of activities of a distinct user in the system.

We measure the success of $\mathcal{A}$ in the $\mathrm{ProfInd}$ game using the metrics $(\mathrm{Adv_{prof}}, \mathsf{H}, \mathsf{C})$. Here, $\mathrm{Adv_{prof}}$ represents the Adjusted Rand Index (ARI) [22], and $\mathsf{H}$, $\mathsf{C}$ stand for the V-Measure coordinates, i.e., Homogeneity and Completeness [15]. $\mathrm{Adv_{prof}}$ measures the expected value of assigning addresses to groups and reflects the *advantage* of $\mathcal{A}$ in grouping addresses over the random assignment of addresses to groups. $\mathrm{Adv_{prof}}$ takes values in $[-1, 1]$ such that the further it is from 0, the further is $\mathcal{A}$'s estimate from the random assignment of addresses to groups; in addition, the closer $\mathrm{Adv_{prof}}$ is to 1, the closer is the adversarial estimate to the true grouping of addresses. In particular, $\mathrm{Adv_{prof}} = \frac{\sum_{i,j}\binom{n_{(i,j)}}{2} - \frac{\sum_i \binom{n_{(i,*)}}{2}\sum_j \binom{n_{(*,j)}}{2}}{\binom{\mathrm{n_A}}{2}}}{\frac{1}{2}[\sum_i \binom{n_{(i,*)}}{2} + \sum_j \binom{n_{(*,j)}}{2}] - \frac{\sum_i \binom{n_{(i,*)}}{2}\sum_j \binom{n_{(*,j)}}{2}}{\binom{\mathrm{n_A}}{2}}}$, where $\mathrm{n_A}$ is the number of Bitcoin addresses, $n_{(i,j)}$ is the number of $u_i$'s elements, which are assigned to group $\mathsf{g}_j$, $n_{(i,*)}$ and $n_{(*,j)}$ are the number of addresses of $u_i$ and $\mathsf{g}_j$ respectively.

On the other hand, the homogeneity ($\mathsf{H}$) and completeness ($\mathsf{C}$) are entropy metrics that measure the clustering quality. $\mathsf{H}$ measures the degree to which the groups of addresses that are produced by $\mathcal{A}$ each contain addresses of the same user, i.e., are homogeneous; $\mathsf{H} = 1$ refers to the case featuring completely homogenous clusters. Complementary to $\mathsf{H}$, $\mathsf{C}$ measures the degree by which the addresses of a user are included in a single cluster. Therefore, $\mathsf{C} = 1$ denotes the case where all the addresses of each user are assigned to the same cluster. $\mathsf{H}$ and $\mathsf{C}$ are computed as follows:

$$\mathsf{H} = \begin{cases} 1 & \text{if } H(\mathrm{U}, G) = 0 \\ 1 - \frac{H(\mathrm{U}|G)}{H(\mathrm{U})} & \text{otherwise} \end{cases}, \quad \text{where:}$$

$$H(\mathrm{U}|G) = -\sum_{j=1}^{|G|}\sum_{i=1}^{|\mathrm{U}|}\frac{\mathrm{n}_{(i,j)}}{\mathrm{n_A}} log \frac{\mathrm{n}_{(i,j)}}{\mathrm{n}_{(*,j)}}, \quad H(\mathrm{U}) = -\sum_{i=1}^{|\mathrm{U}|}\frac{\mathrm{n}_{(i,*)}}{\mathrm{n_A}} log \frac{\mathrm{n}_{(i,*)}}{\mathrm{n_A}},$$

$$\mathsf{C} = \begin{cases} 1 & \text{if } H(G, \mathrm{U}) = 0 \\ 1 - \frac{H(G|\mathrm{U})}{H(G)} & \text{otherwise} \end{cases}, \quad \text{where:}$$

$$H(G|\mathrm{U}|) = -\sum_{i=1}^{|\mathrm{U}|}\sum_{j=1}^{|G|}\frac{\mathrm{n}_{(i,j)}}{\mathrm{n_A}} log \frac{\mathrm{n}_{(i,j)}}{\mathrm{n}_{(i,*)}}, \quad H(G) = -\sum_{j=1}^{|G|}\frac{\mathrm{n}_{(*,j)}}{\mathrm{n_A}} log \frac{\mathrm{n}_{(*,j)}}{\mathrm{n_A}}.$$

[6] The number of users participating in the Bitcoin network is public.

## 4 Evaluating Privacy in Bitcoin

In what follows, we show how the adversary, given pubLog, can gather some knowledge about Bitcoin users by exploiting the properties of existing Bitcoin client implementations. We then evaluate, by means of our Bitcoin simulator, the success of the adversary in the aforementioned privacy games given this knowledge.

### 4.1 Exploiting Existing Bitcoin Client Implementations

Current Bitcoin client implementations enable $\mathcal{A}$ to link a fraction of Bitcoin addresses that belong to the same user.

**Heuristic I—Multi-input Transactions:** As mentioned earlier, multi-input transactions occur when $u$ wishes to perform a payment, and the payment amount exceeds the value of each of the available BTCs in $u$'s wallet. Clearly, if Bitcoin transactions only featured a single input, then $u$ would have to perform multiple transactions to complete the payment, i.e., each with a different BTC from her wallet. However, this would endure losses in terms of transaction fees. To circumvent such scenarios, existing Bitcoin clients choose a set of BTCs from $u$'s wallet (such that their aggregate value matches the payment) and perform the payment through multi-input transactions. It is therefore straightforward to conclude that if these BTCs are owned by different addresses, then the input addresses belong to the same user. Note that, currently, Bitcoin clients do not provide support for different users to participate in a single transaction; to achieve this, users would have to modify the Bitcoin protocol themselves.

**Heuristic II—"Shadow" Addresses:** As mentioned earlier, Bitcoin generates a new address, the "shadow" address [5], on which each sender can collect back the "change".

This mechanism suggests a distinguisher for shadow addresses. Namely, in the case when a Bitcoin transaction has two output addresses, $a_{\mathsf{R}_n}$, $a_{\mathsf{R}_o}$, such that $a_{\mathsf{R}_n}$ is a new address (i.e., an address that has never appeared in pubLog before), and $a_{\mathsf{R}_o}$ corresponds to an old address (an address that has appeared previously in pubLog), we can safely assume that $a_{\mathsf{R}_n}$ constitutes a shadow address for $a_i$. Note that, in current Bitcoin implementations, users cannot issue transactions with two different outputs (that correspond to two different users) without modifying the Bitcoin protocol.

**Evaluating Heuristics I and II:** In what follows, we evaluate the implications of these heuristics on the user-privacy in Bitcoin. Given that the number of Bitcoin users in September 2011 was estimated by about 60,000 users [8], we created a C++ parser that parses the first 140,000 blocks (till August 2011) in the Bitcoin block explorer [7].

Our C++ parser extracts all the addresses in each block and categorizes them in clusters of Generic Addresses, GAs, given the two aforementioned heuristics. The parser then outputs a list of addresses organized in different GAs.

Our results show that there were 1,632,648 unique addresses in the first 140,000 blocks. Given Heuristic I, we could classify these addresses into 1,069,699 distinct GAs. Given Heuristic II, this number decreases to 693,051 GAs; this corresponds to grouping approximately 58% of Bitcoin addresses with an average of 11.55 addresses

per GA. This clearly shows that $\mathcal{A}$'s advantage in the $\mathrm{AddUnl}$ game is considerable given Heuristics I and II.

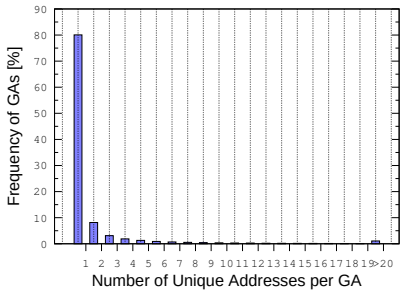


**Fig. 2.** Number of addresses per GA in the first 140,000 Bitcoin blocks after applying Heuristics I and II.

Figure 2 shows the number unique addresses per GAs; almost 82% of the 693,051 GAs have only a single address. We believe that this is the case since most users initially used Bitcoin solely to mine and invest in BTCs; this explains why a number of users did not perform any transaction with their generated BTCs.

### 4.2 Behavior-Based Analysis

Besides exploiting current Bitcoin implementations, $\mathcal{A}$ could also make use of behavior-based clustering techniques, such as K-Means (KMC), and the Hierarchical Agglomerative Clustering (HAC) algorithms. Let $\mathsf{U}$ be the set of users populating Bitcoin and $(\mathsf{GA}_1, \ldots, \mathsf{GA}_{\mathrm{n_{GA}}})$ denote the GAs that $\mathcal{A}$ has obtained by applying the two aforementioned heuristics on $\mathsf{pubLog}$, respectively. Given this, the goal of $\mathcal{A}$ is to output a group of clusters of addresses $G = \{\mathsf{g}_1, \ldots, \mathsf{g}_{n_\mathsf{U}}\}$ such that $G$ best approximates $\mathsf{U}$. Since each GA is owned by exactly one user, the estimate on the assignment of each $\mathsf{GA}_i$ can be modeled by a variable $z_i$ such that $z_i = k$, if and only if, $\mathsf{GA}_i$'s owner is $\mathsf{g}_k$.

In fact, HAC assumes that initially each GA represents a separate cluster ($\{z_i = i\}_{i=1}^{\mathrm{n_{GA}}}$) and computes similarity values for each pair of clusters. Clusters with higher similarity value are combined into a single cluster and cluster-to-cluster similarity values are re-computed. The process continues until the number of created clusters equals the number of users $\mathrm{n_U}$. KMC is then initialized using the output of HAC and assumes that each user is represented by the center of each cluster. The algorithm iterates assignments of GAs to clusters and aims at minimizing the overall distance of GAs to the center of the cluster they have been assigned to. The centers of the clusters and the GA-to-cluster distances are re-computed in each round.

In our implementation (cf. Section 4.3), we represent each transaction that appears within a GA using: *(i)* the time at which the transaction took place, *(ii)* the indexes of the different GAs that appear within the transaction (as senders or recipients), and *(iii)* the values of the BTCs spent by the transaction. Let $\tau_x$ denote the set of transactions of $\mathsf{GA}_x$. The degree of similarity between $\mathsf{GA}_i$ and $\mathsf{GA}_j$, denoted by $\mathsf{Sim}(\mathsf{GA}_i, \mathsf{GA}_j)$, is then represented by the cosine similarity of lists $\tau_i$ and $\tau_j$, i.e., $\mathsf{Sim}(\mathsf{GA}_i, \mathsf{GA}_j) = \frac{\sum_{\forall \tau \in \tau_i \cap \tau_j} (f_{(\tau,i)} \cdot f_{(\tau,j)})}{\|\tau_i\| \cdot \|\tau_j\|}$, where $f_{(\tau,i)}$, $f_{(\tau,j)}$ are the occurrences of item $\tau$ in lists $\tau_i$ and $\tau_j$ respectively, and $\|X\|$ denotes the norm II of vector $X$. Given this, the resulting distance metric in KMC is $\mathsf{Dist}(\mathsf{GA}_i, \mathsf{g}_k) = \frac{2}{1+\mathsf{Sim}(\mathsf{GA}_i, \mathsf{g}_k)} - 1$.

Our implementation also accounts for constraints that are posed by real-world settings. Namely, since users cannot be physically located in two different places at the same time, they cannot participate in two different (physical) exchanges of goods at the

same time. To account for this case (cf. Section 4.4), we apply different weighting for similarity of GAs who participate in transactions concurrently.

### 4.3 Simulating the use of Bitcoin in a University

To evaluate the success of $\mathcal{A}$ in the $\mathrm{AddUnl}$ and the $\mathrm{ProfInd}$ games, we simulate a realistic case of using Bitcoin in a university environment among Computer Science students[7]. Here, we assume that the shops located around the university also accept BTCs as a currency. Given the lack of details and statistics about the current use of Bitcoin, this was one of the few "workable" uses of Bitcoin that we could try to accurately model and through which we could evaluate the advantage of $\mathcal{A}$ in the system. We believe that, while Bitcoin is still not being used in such environments, if Bitcoin would be adopted as an alternative currency in the near future, then it is highly likely that a university environment would be one of the first candidate environments to "host" Bitcoin. In Section 5, we discuss the implications of our findings in generic uses of Bitcoin.

**Experimental Setup:** To evaluate the privacy implications of using Bitcoin in a university environment, we devised the setup shown in Figure 3. Our Bitcoin simulator takes an XML configurations file as input and outputs: *(i)* a log that details the events that were simulated, the "ground truth", as well as *(ii)* the resulting simulated public Bitcoin log, pubLog. The XML configurations file contains all the necessary parameters to run the simulator. These include the number of users, the number of miners, the simulation time, the difficulty in block generation, as well as usage configurations for creating user profiles and Bitcoin sellers/buyers. Further details about our simulator can be found in Appendix C.

As shown in Figure 3, the outputs of our simulator are used to evaluate $\mathcal{A}$'s success. In fact, once the simulations terminate, a Perl-based parser uses the simulated Bitcoin block as input and pre-classifies the simulated addresses into GAs according to Heuristics I and II. The resulting "pre-filtered output" is then fed into our clustering algorithms, the HAC and the KMC algorithms (both implemented in C). The output of these algorithms is then compared using another Perl-based script with the "ground truth" generated by the simulator in order to compute the success of $\mathcal{A}$ in the $\mathrm{AddUnl}$ and the $\mathrm{ProfInd}$ games.

We tuned our simulator to match a real-world scenario that reflects the actual behavior of the staff and student members of a Computer Science Department of a university in the Fall 2012 semester. In our setting, we consider a variable number of users, 5.2% of which are "Professors", 42.0% are "Staff" and the remaining 52.8% are "Students". We consider a total of 6 events, each having several options: lunching/dining (12 options), buying groceries (2 options), buying from vending machines (4 options), online shopping (5 options), purchasing books (2 options), and performing barters with other users, totalling 25 different Bitcoin vendors present in our system. For each user, we assign a probability that the user undergoes each of the possible options of each event. These probabilities are assigned according to the "category" of the user; that is, if the user is a "Professor", then it is more likely that he/she would eat lunches at more expensive

[7] Details are anonymized due to the anonymous reviewing process.

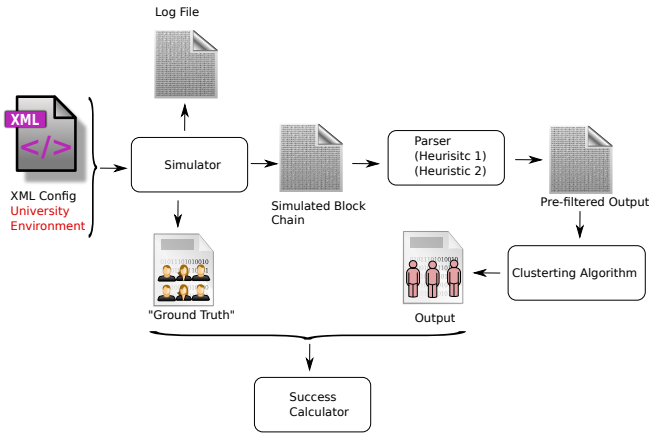


**Fig. 3.** Experimental setup used throughout our simulations. The outputs of our Bitcoin simulator are pre-filtered according to Heuristics I and II and then fed as input to our clustering algorithm. The clustering result is then compared with the "ground truth" that is emulated by our simulator.

restaurants, when compared to the case where the user falls in the "Student" category. For each event, we specify in the XML configuration the following parameters: the frequency of the event, and the price range per option of the event. Note that each option is assigned a rating that would reflect its popularity. The probability of performing an option is interpolated from the frequency of occurrence of the event per week, and from the rating of the option. To ensure a large variety of profiles in our user base, we specify in the XML configuration a minimum and maximum value for the frequency, rating and price fields. These bounds depend on the category of the user, the event and option in question. Listing 1 in Appendix B depicts an example of configuration parameters for a "lunch" event. At the start of our experiments, users originally have few ($< 10$) Bitcoin addresses; as they issue new transactions, new (shadow) addresses are created in their wallets. In the XML file, we also model the behavior of "privacy-aware" users. We assume that these users are motivated to increase their privacy in Bitcoin; for that purpose, these users create new Bitcoin addresses in their wallets and send some of their BTCs from their old to their new addresses. Our simulator also supports the transfer of BTCs among users.

### 4.4 Experimental Results

Throughout our experiments, we emulated two different scenarios for each simulation round. In the first scenario denoted by "Partial Knowledge", we assume that $\mathcal{A}$ is aware of the location/service of all Bitcoin vendors and as such can distinguish whether a transaction was performed in exchange of a physical good. In this case, we include the vendors's addresses in the prior knowledge of $\mathcal{A}$ when computing $\mathrm{Adv}_{\mathrm{unl}}$; we also assume that $\mathcal{A}$ can tune the clustering algorithm to take into account that the same user performing this transaction cannot appear in other transactions that takes place at the same time. This case emulates the realistic setting where $\mathcal{A}$ can extract a subset of the addresses owned by geographically co-located Bitcoin users/vendors from the overall public Bitcoin log; for example, $\mathcal{A}$ can extract from the Bitcoin log all the addresses that

| | **100 (50%)** | **200 (0%)** | **200 (50%)** | **200 (100%)** | **400 (50%)** |
|---|---|---|---|---|---|
| $\mathrm{Adv}_{\mathrm{unl}}$ | $4.558 \pm 0.142$ | $8.849 \pm 0.140$ | $6.764 \pm 0.149$ | $6.225 \pm 0.070$ | $9.393 \pm 0.068$ |
| $\mathrm{Adv}_{\mathrm{prof}}$ | $0.523 \pm 0.015$ | $0.579 \pm 0.014$ | $0.512 \pm 0.011$ | $0.421 \pm 0.012$ | $0.477 \pm 0.006$ |
| H | $0.761 \pm 0.006$ | $0.868 \pm 0.008$ | $0.787 \pm 0.005$ | $0.699 \pm 0.013$ | $0.795 \pm 0.003$ |
| C | $0.778 \pm 0.009$ | $0.941 \pm 0.002$ | $0.812 \pm 0.005$ | $0.707 \pm 0.008$ | $0.825 \pm 0.003$ |

**(a)** Results in the "Partial Knowledge" scenario.

| | **100(50%)** | **200(0%)** | **200(50%)** | **200(100%)** | **400(50%)** |
|---|---|---|---|---|---|
| $\mathrm{Adv}_{\mathrm{unl}}$ | $4.808 \pm 0.113$ | $8.832 \pm 0.189$ | $6.757 \pm 0.132$ | $6.230 \pm 0.039$ | $9.263 \pm 0.153$ |
| $\mathrm{Adv}_{\mathrm{prof}}$ | $0.566 \pm 0.008$ | $0.633 \pm 0.010$ | $0.521 \pm 0.019$ | $0.430 \pm 0.015$ | $0.489 \pm 0.014$ |
| H | $0.787 \pm 0.011$ | $0.885 \pm 0.004$ | $0.791 \pm 0.011$ | $0.707 \pm 0.013$ | $0.800 \pm 0.010$ |
| C | $0.802 \pm 0.011$ | $0.943 \pm 0.002$ | $0.817 \pm 0.008$ | $0.712 \pm 0.010$ | $0.826 \pm 0.009$ |

**(b)** Results in the "No Knowledge" scenario.

**Table 1.** Behavior-based clustering results in the "Partial Knowledge" and "No Knowledge" scenarios. A column entitled X (Y%) denotes an experiment featuring X users among which $Y\%$ are privacy-aware. Each data point in our plots is averaged over five rounds of experiments; we also present the corresponding $95\%$ confidence intervals (shown after the "$\pm$" sign).

interact with a known address of a vendor located within the university environment. In the second scenario denoted by "No Knowledge", we consider the case where $\mathcal{A}$ does not know the location or service of the vendors, and as such does not have any prior knowledge, but assumes that up to 10% of the transactions are performed in exchange of goods delivered over the Internet.

Given this setup, we evaluate the metrics $\mathrm{Adv}_{\mathrm{prof}}$, H, C, and $\mathrm{Adv}_{\mathrm{unl}}$ with respect to *(i)* the fraction of "privacy-aware" users and *(ii)* the number of users $\mathrm{n_U}$. By privacy-aware users, we refer to users that manually generate new Bitcoin addresses (following a configuration in the XML file) to enhance their privacy in the system.

Table 1 depicts our findings. Our results show that both the "Partial Knowledge" and the "No Knowledge" configurations exhibited comparable results.

In the first round of experiments, we evaluate the success of $\mathcal{A}$ with respect to the fraction of "privacy-aware" users. More specifically, we run our clustering and privacy evaluation algorithm in a setting featuring $200$ users, among which $0\%, 50\%$, and $100\%$ of the users are privacy-aware. Table 1 shows the $\mathrm{Adv}_{\mathrm{unl}}$, $\mathrm{Adv}_{\mathrm{prof}}$, H, and C with respect to the different fractions of privacy-aware users.

When none of users in the system are privacy-aware, the performance of our clustering algorithms is high. In particular, $\mathrm{Adv}_{\mathrm{unl}}$ is in both configurations around $8.84$ which means that $\mathcal{A}$ performs much better than in the case where $\mathcal{A}$ assumes that all addresses (that are not part of the prior knowledge of $\mathcal{A}$) have $\frac{1}{2}$ probability of being owned by the same user. The resulting $\mathrm{Adv}_{\mathrm{prof}}$ reaches $0.58$ and $0.63$ in the "No Knowledge" and "Partial Knowledge" scenarios, respectively, which shows that the estimate chosen by $\mathcal{A}$ is close to the true assignment of users to clusters. Conforming with these findings, both H and C are close to 1, which signifies that the clusters constructed by $\mathcal{A}$ are homogenous, and that each user's set of addresses is almost completely clustered. As the fraction of privacy-aware users increases, $\mathrm{Adv}_{\mathrm{unl}}$ and $\mathrm{Adv}_{\mathrm{prof}}$ slightly decrease. In this case, $\mathrm{Adv}_{\mathrm{unl}}$ drops from values close to $8.84$ to $6.23$, while $\mathrm{Adv}_{\mathrm{prof}}$ drops to $0.43$.

C, H also decrease from $0.89$ and $0.94$, respectively, to $0.71$. Nevertheless, these results indicate a considerable advantage over the adversary that makes random guesses.

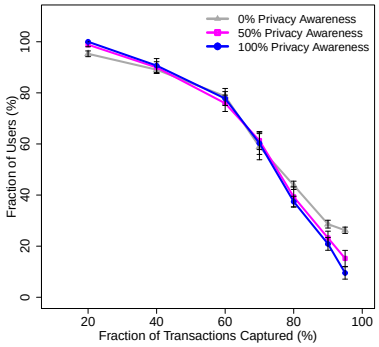


**Fig. 4.** Fraction of transactions captured by our clustering algorithms in the "No Knowledge" case.

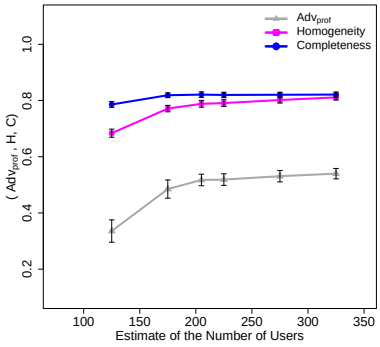


**Fig. 5.** The case where $\mathcal{A}$ cannot accurately estimate $\mathrm{n_U}$. We assume the 'No Knowledge" case where $\mathrm{n_U} = 200$.

Figure 4 depicts the overall fraction of user profiles (measured by means of the similarity of transactions appearing in a user's wallet and the corresponding cluster) that are captured by $\mathcal{A}$. Our results show that in the case when $\mathrm{n_U} = 200$ users with 0% privacy awareness, almost $42\%$ of the users have their preferences captured with $80\%$ accuracy. In the case featuring 100% privacy-aware users, this fraction drops to 35% of the users whose profile was correctly clustered with an accuracy of at least $80\%$. We therefore conclude that the privacy of users in Bitcoin can be compromised, even if users manually create new addresses in order to enhance their privacy in the system.

Furthermore, our results show that $\mathrm{Adv_{prof}}$, H, and C slightly decrease when the number of users increases from $100$ to $400$. This is the case since, as the number of users increases, the probability that two users perform similar transactions (comparable collaborator, time and amount), and therefore have similar *profiles*, increases. Since addresses with similar behavior can be easily misclassified, the higher the number of users that participate within a given period of time is, the lower are the resulting privacy metrics (see discussion in Appendix D for more details). On the contrary, $\mathrm{Adv_{unl}}$ increases considerably to values around $9.30$ for $\mathrm{n_U}= 400$. As shown in Appendix E, this stems from the fact that the distance between an adversary that makes random guesses (the first term in $\mathrm{Adv_{unl}}$ definition) from the true assignment of addresses increases when $\mathrm{n_U}$ increases.

In Figure 5, we evaluate the case where $\mathcal{A}$ does not have an accurate estimate of the number of users in the university setting. Our findings show that even if $\mathcal{A}$'s estimate of the number of users is not accurate, the privacy of a considerable fraction of users is still compromised. We therefore argue that our findings apply to other possible uses

of Bitcoin, since $\mathcal{A}$ can simply extract from the Bitcoin log a subset of the addresses of users that interact with a Bitcoin vendor at any given location (cf. Section 5).

## 5 Discussion

So far, our analysis focused on the current implementation of Bitcoin when used in a university setting. In what follows, we analyze the implications of our findings in generic implementations/uses of Bitcoin.

**Evading Heuristic I:** We start by showing that Heuristic I cannot be easily evaded in any future implementation of Bitcoin without compromising the basic operation of Bitcoin. Indeed, the combination of multiple inputs ensures that coins with "large" values can be recreated from existing smaller BTCs; this process is necessary otherwise the value of coins will be continuously deprecated following every issued transaction until the value of these coins reaches the minimum amount. At that point in time, the only way for Bitcoin users to issue transactions without combining their previous coins is to perform multiple transactions with single-input, one coin at a time. This clearly does not solve the problem since *(i)* this process can still be tracked by the adversary (transactions will be linked by time) and *(ii)* this will incur more transaction fees (one fee per transaction). Another alternative would be for Bitcoin developers to provide support for different users to transparently participate in a single transaction. While this would increase the use-cases where Bitcoin finds applicability (e.g., performing contracts [9]), the collaborative construction of transactions by different users is unlikely to be predominately used in the network.

**Evading Heuristic II:** Similarly, it is easy to see that evading Heuristic II can only decrease the privacy of Bitcoin users. That is, if "shadow" addresses were not utilized, and the change of coins is simply put back in the sender's address then users' activities can be traced in an easier way. We point out, that while Heuristic II is indeed a best-effort solution to increase the privacy of users, it results in the dispersion of the coins among several addresses of the users. This makes the privacy leakage due to Heuristic I even more considerable. One possible way to "evade" Heuristic II would be for Bitcoin users to *(i)* first divide the coins according to the required payment in one transaction and *(ii)* then make the payment with zero change. Another possible solution to "harden" the reliance on Heuristic II would be for Bitcoin to support Mutli-Input Multi-Output (MIMO) transactions[8].

**Implications to Generic Uses of Bitcoin:** We argue that our findings are not specific to the studied university setting and apply to other generic-uses of Bitcoin. More specifically, we believe that the adversary can, to a large extent, extract from the public Bitcoin log a small set of addresses that correspond to geographically co-located users; the adversary can subsequently run our clustering algorithms on the extracted set of addresses. For instance, if Bitcoin were to be used across shops, then the adversary can

[8] Currently, Bitcoin only enables the construction of transactions that have two outputs, when change is needed.

simply extract all the addresses that interacted with a specific set of Bitcoin vendors[9] that are located within a specific geographic region. The larger is the number of addresses of (physical) vendors that the adversary knows, the more complete is the view of the adversary of the geographic sub-network. This impacts, in turn, the accuracy of the clustering results (cf. Figure 5); when $\mathcal{A}$ knows the entire network, then she has an accurate estimate of the number of active users.

One possible technique to enhance privacy in Bitcoin would be therefore to require that vendors use a newly generated recipient address in each transaction that they receive; this address should be communicated to clients using a private channel (e.g., via email). This ensures that an adversary cannot extract the set of addresses used by geographical co-locate users; this, in turn, results in the fact that the input set of addresses/transaction fed to the clustering algorithms is large, which can distort the advantage of the adversary in the system. In most cases however, the amount of BTCs paid in a transaction can be used to acquire knowledge about a given vendor.

We therefore conclude that our techniques and analysis are not specific to a given Bitcoin implementation but apply to the generic Bitcoin protocol. Based on this analysis, it is clear that the reliance on third-party trusted entities (e.g., Bitcoin banks, FlexCoin [10]) emerges as one of the few workable solutions to increase the privacy of Bitcoin clients. By storing the coins of a large number of users, these entities can hide the direct relationship between the inputs and outputs of a transaction within a sufficiently large anonymity set—and therefore better support the privacy of users. Clearly, this solution comes at odds with the main intuition behind Bitcoin, originally planned as a completely decentralized digital payment system.

## 6 Related Work

First introduced in 2008, Bitcoin has recently attracted the attention of the research community. In [13], Elias investigates the legal aspects of privacy in Bitcoin. In [16], Babaioff *et al.* address the lack of incentives for Bitcoin users to include recently announced transactions in a block, while in [12], Syed *et al.* propose a user-friendly technique for managing Bitcoin wallets.

Finney [11] describes a double-spending attack in Bitcoin where the attacker includes in her generated blocks transactions that double-spend some coins between her own addresses. In [23], Karame *et al.* thoroughly investigate double-spending attacks in Bitcoin and show that double-spending fast payments in Bitcoin can be performed in spite of the measures recommended by Bitcoin developers. Clark *et al.* [21] propose the use of the Bitcoin PoW to construct verifiable commitment schemes. Barber *et al.* [17] analyze possible ways to enhance the resilience of Bitcoin against a number of security threat. Reid and Harrigan [24] analyze the flow Bitcoin transactions in a small part of Bitcoin log, and show that external information, i.e., publicly announced addresses, can be used to link identities and organizations to some transactions.

ECash [18–20] and anonymous credit cards were the first attempts to provide privacy-preserving transactions. ECash offers strong anonymity and accountability guarantees

[9] Recall that the addresses of Bitcoin vendors are generally public.

for buyers by relying on a set of cryptographic primitives that ensure that payments issued pertaining to the same user cannot be linked to each other and anonymous, provided that the user does not misbehave.

## 7 Conclusion

In this paper, we explored the privacy implications of Bitcoin. More specifically, we evaluated the privacy measures adopted in Bitcoin and we analytically and empirically quantified the advantage of an adversary in defeating these measures by means of a simulation that mimics the use of Bitcoin in a realistic university setting.

Our findings show that the current measures adopted by Bitcoin are not enough to protect the privacy of users if Bitcoin were to be used as a digital currency in realistic settings. More specifically, our results show that if Bitcoin is used as a digital currency to support the daily transactions of users in a typical university environment, then behavior-based clustering techniques can unveil, to a large extent, the profiles of $40\%$ of Bitcoin users, even if these users try to enhance their privacy by manually creating new addresses. Finally, we explored a number of solutions that could be integrated by Bitcoin developers to enhance the privacy of users.

## References


1. Bitcoin, Monetarists Anonymous, Available from http://www.economist.com/node/21563752?fsrc=scn/tw/te/pe/monetaristsanonymous.
2. Trade - Bitcoin, Available from https://en.bitcoin.it/wiki/Trade.
3. Bitcoin Charts, Available from http://bitcoincharts.com/.
4. Transaction Fees - Bitcoin, Available from https://en.bitcoin.it/wiki/Transaction_fees.
5. Bitcoin — Accounts Explained, Available from https://en.bitcoin.it/wiki/Accounts_explained.
6. Bitcoin Block 80000, Available from http://blockexplorer.com/b/80000.
7. Bitcoin Block Explorer, Available from http://blockexplorer.com/.
8. Bitcoin – Wikipedia, Available from https://en.bitcoin.it/wiki/Introduction.
9. Contracts - Bitcoin, Available from https://en.bitcoin.it/wiki/Contracts.
10. Flexcoin –The Bitcoin Bank, Available from http://www.flexcoin.com/.
11. The Finney Attack, Available from https://en.bitcoin.it/wiki/Weaknesses#The_.22Finney.22_attack.
12. Bitcoin Gateway, A Peer-to-peer Bitcoin Vault and Payment Network, 2011. Available from http://arimaa.com/bitcoin/.
13. Bitcoin: Tempering the Digital Ring of Gyges or Implausible Pecuniary Privacy, 2011. Available from http://ssrn.com/abstract=1937769ordoi:10.2139/ssrn.1937769.
14. Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System.
15. Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, 2007.
16. M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On Bitcoin and Red Balloons. *CoRR*, 2011.

17. S. Barber, X. Boyen, E. Shi, and E. Uzun. Bitter to Better - How to Make Bitcoin a Better Currency. In *Proceedings of Financial Cryptography and Data Security*, 2012.
18. S. Brands. Electronic Cash on the Internet. In *Proceedings of the Symposium on the Network and Distributed System Security*, 1995.
19. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact E-Cash. In *Proceedings of Advances in Cryptology - EUROCRYPT*, 2005.
20. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings on Advances in Cryptology - CRYPTO*, 1990.
21. J. Clark and A. Essex. (Short Paper) CommitCoin: Carbon Dating Commitments with Bitcoin. In *Proceedings of Financial Cryptography and Data Security*, 2012.
22. Douglas Steinley. Properties of the hubert-arabie adjusted rand index. *Psychol Methods*, 2004.
23. G. Karame, E. Androulaki, and S. Capkun. Double-Spending Fast Payments in Bitcoin. In *Proceedings of ACM CCS*, 2012.
24. F. Reid and M. Harrigan. An Analysis of Anonymity in the Bitcoin System. *CoRR*, 2011.

## A Bitcoin Block Explorer

**Hash:** 000000000043a8c0fd1d6f726790caa2a406010d19efd2780db27bdbbd93baf6
**Previous block:** 00000000001937917bd2caba204bb1aa530ec1de9d0f6736e5d85d96da9c8bba
**Next block:** 00000000000036312a44ab7711afa46f475913fbd9727cf508ed4af3bc933d16
**Time:** 2010-09-16 05:03:47
**Difficulty:** 712.884864
**Transactions:** 2
**Merkle root:** 8fb300e3fdb6f30a4c67233b997f99fdd518b968b9a3fd65857bfe78b2600719
**Nonce:** 1462756097

| Input/Previous Output | Source & Amount | Recipient & Amount |
|---|---|---|
| N/A | Generation: 50 + 0 total fees | Generation: 50 + 0 total fees |
| f5d8ee39a430...:0 | 1JBSCVF6VM6QjFZyTnbpLjoCJ...: 50 | 16ro3Jptwo4asSevZnsRX6vf..: 50 |

**Table 2.** Example Block of Bitcoin. The block contains 2 transactions, one of which awards the miner with 50 BTCs.

## B Example Configuration File

**Listing 1** Example of XML configuration parameters for a "lunch" event with 12 options corresponding to the profile of a "Professor".

```
<!– lunch, eventid="0" refers to event with id="0" from above –>
<ProfileEvent eventid="0" minFreqPerWeek="5" maxFreqPerWeek="5" >
<Store storeid="0" maxPref="1" minPref="0" maxPrice="10.0" minPrice="8.0" />
<Store storeid="1" maxPref="1" minPref="0" maxPrice="13.0" minPrice="10.0" />
<Store storeid="2" maxPref="1" minPref="0" maxPrice="15.0" minPrice="13.0" />
<Store storeid="3" maxPref="4" minPref="2" maxPrice="25.0" minPrice="15" />
<Store storeid="4" maxPref="2" minPref="0" maxPrice="20.0" minPrice="15.0" />
<Store storeid="5" maxPref="2" minPref="0" maxPrice="17.0" minPrice="12.0" />
<Store storeid="6" maxPref="0.5" minPref="0" maxPrice="20.0" minPrice="8.0" />
<Store storeid="7" maxPref="0.5" minPref="0" maxPrice="10.0" minPrice="7.5" />
<Store storeid="8" maxPref="4" minPref="2" maxPrice="25.0" minPrice="15" />
<Store storeid="9" maxPref="0.5" minPref="0" maxPrice="25.0" minPrice="10" />
<Store storeid="10" maxPref="0.5" minPref="0" maxPrice="10.0" minPrice="5.0" />
<Store storeid="11" maxPref="3" minPref="1" maxPrice="12.0" minPrice="9.0" />
</ProfileEvent>
```

## C Bitcoin Simulator

Our simulator is round-based; in each simulation round (defined as a "weekly timestepping" interval), events are added to a priority queue with a probability dictated by the configuration file. These events correspond to one of the following operations:

– *Issue a new transaction:* Users might issue new Bitcoin transactions whose time, value, beneficiary and purpose stem from the XML configurations file. The process of transaction issuance in our simulator fully mimics its counterpart in the genuine Bitcoin system.

– *Generate a new Bitcoin address:* Here, in addition to the automatically generated addresses in Bitcoin (cf. Section 3), "privacy-aware" users might decide to generate a number of new addresses to further "obfuscate" their usage of Bitcoin.

Conforming with the current use of Bitcoin, only few users in our setting were miners (i.e., mining is currently mostly performed by dedicated mining pools).

Our Bitcoin simulator abstracts away network delays, congestion, jitter, etc.. We also assume that all transactions in the system are well-formed and we do not model transaction fees that are incurred in the network. While malformed transactions and double-spending attempts [23] can be indeed witnessed in the genuine Bitcoin system, we believe that malicious behavior in Bitcoin is orthogonal to our privacy investigation—which explains the reason why we did not model such a misbehavior in our simulator. Moreover, our simulator relies on a variant greedy algorithm that closely approximates the genuine algorithm used in Bitcoin. Note that while the distribution of generated blocks in our simulator matches that in Bitcoin [23][10], we increased the average time between the generation of successive blocks in the simulator to better cope with simulated network dynamics[11].

## D Captured Transactions w.r.t. $\mathrm{n_U}$

Figure 6 shows the impact of the number of users $\mathrm{n_U}$ on the performance of our profiling algorithms.

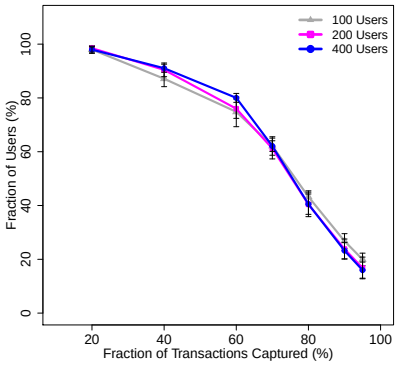


**Fig. 6.** Fraction of captured transactions with respect to $\mathrm{n_U}$. Here, we consider the "No Knowledge" case where $50\%$ of the users are privacy-aware.

Our results show that the fraction of users whose transactions are correctly captured by our algorithms is not considerably affected by the number of users in the system. For instance, the fraction of users whose profiles were captured with an accuracy of at least $80\%$ is approximately $40\%$ when $\mathrm{n_U} = 100, 200, 400$.

Note that this conforms with our previous observations in Section 4.4; as shown in Figure 5, $\mathrm{Adv_{prof}}$ decreases as $\mathrm{n_U}$ increases. This is because, as the number of users increases, the probability that users (and their corresponding addresses) witness similar behavior (i.e., transactions) increases. In our clustering algorithms, this can result in the misclassification of similar addresses (that belong to different users), which would result in a smaller $\mathrm{Adv_{prof}}$. However, since similar addresses consist of similar transactions,

[10] It was shown in [23] that the block generation in Bitcoin follows a shifted geometric distribution with parameter 0.19.

[11] Throughout our experiments, we considered that new blocks were generated every 20 minutes on average.

| | **100 (50%)** | **200 (0%)** | **200 (50%)** | **200 (100%)** | **400 (50%)** |
|---|---|---|---|---|---|
| Partial Knowledge | $0.774 \pm 0.007$ | $0.823 \pm 0.005$ | $0.832 \pm 0.004$ | $0.830 \pm 0.005$ | $0.871 \pm 0.003$ |
| No Knowledge | $0.785 \pm 0.004$ | $0.838 \pm 0.003$ | $0.832 \pm 0.004$ | $0.831 \pm 0.003$ | $0.872 \pm 0.003$ |

**Table 3.** $\mathrm{Adv}_{\mathrm{unl}}^{\mathrm{norm}}$ with respect to the number of users and the fraction of privacy-aware users.

the fraction of correct transactions captured per user is not affected in this case (as shown in Figure 6).

## E Evaluation of the Normalized Variant of $\mathbf{Adv_{unl}}$

$\mathrm{Adv}_{\mathrm{unl}}^{\mathrm{norm}}$ denotes the comparative advantage of $\mathcal{A}$ in the $\mathrm{AddUnl}$ game with respect to that of an adversary who chooses her answers at random, i.e., who assumes that all addresses (that are not part of her prior-knowledge) are equally likely to be owned by the same user. $\mathrm{Adv}_{\mathrm{unl}}^{\mathrm{norm}}$ is computed as follows:

$$\mathrm{Adv}_{\mathrm{unl}}^{\mathrm{norm}} = \frac{\frac{\|\mathrm{PriorKnowledge}-\mathrm{Genuine}\|}{\|\mathrm{Genuine}\|} - \frac{\|\mathrm{Estimate}-\mathrm{Genuine}\|}{\|\mathrm{Genuine}\|}}{\frac{\|\mathrm{PriorKnowledge}-\mathrm{Genuine}\|}{\|\mathrm{Genuine}\|}}.$$

where $\|X\|$ denotes the norm II of vector $X$, and $\mathrm{Estimate}$, $\mathrm{PriorKnowledge}$ are two $\mathrm{n_A} \times \mathrm{n_A}$ matrices, that are computed as shown in Section 3.1. $\mathrm{Estimate}[i,j]$ denotes $\mathcal{A}$'s guess on the probability that addresses $a_i$ and $a_j$ belong to the same user, while $\mathrm{PriorKnowledge}[i,j]$ reflects the guess of an adversary who has the same prior-knowledge as $\mathcal{A}$ but assigns a probability of 0.5 to associate addresses that are not part of her prior-knowledge. $\mathrm{Genuine}$ denotes the genuine assignment matrix between addresses in the entire network.

Table E depicts the normalized version of $\mathrm{Adv}_{\mathrm{unl}}$, $\mathrm{Adv}_{\mathrm{unl}}^{\mathrm{norm}}$, with respect to the number of users and the fraction of privacy-aware users.