Design of Privacy-preserving Mobile Bitcoin Client Based on γ-Deniability Enabled Bloom Filter

Kota Kanemura, Kentaroh Toyoda, Tomoaki Ohtsuki
Keio University, 3-14-1, Hiyoshi, Kohoku-ku,
Yokohama, 223-8522, Japan
Email: kanemura@ohtsuki.ics.keio.ac.jp, toyoda@ohtsuki.ics.keio.ac.jp, ohtsuki@ics.keio.ac.jp

***Abstract*— Bitcoin is a decentralized currency system that does not need any central authorities. All transactions issued by users have been recorded in the common ledger, called blockchain, which is shared by all users. In Bitcoin, an SPV (Simplified Payment Verification) client, which is a lightweight client that does not possess the entire blockchain, are developed for storage constrained devices such as a mobile phone. For an SPV client to check if there are transactions related to it, a Bloom filter where their Bitcoin addresses are involved is sent to a full client that possesses the entire blockchain. The full client only transfers transactions of which Bitcoin addresses are positive on the received Bloom filter. However, it is necessary to preserve the privacy of SPV clients when designing a Bloom filter because SPV clients' Bitcoin addresses will be identified by a full client with high probability. In this paper, we propose a privacy-preserving Bloom filter design for SPV clients based on γ-Deniability. γ-Deniability is a privacy metric that shows how much true positive Bitcoin addresses are hidden by the false positives in a Bloom filter. Furthermore, in order to design a Bloom Filter that satisfies a certain γ-Deniability, it is necessary to know the number of unique Bitcoin addresses that appear for the first time since the queried time. Based on our manual inspection, we propose to estimate it based on the linear regression. We show that our scheme achieves good estimation accuracy and γ through the simulation with a real Bitcoin blockchain.**

I. Introduction

A revolution of cryptocurrencies is very trending in the world [1]. In particular, Bitcoin is the most successful one and can be exchanged with a real currency [2], [3]. Bitcoin is a digital currency system maintained by users in the P2P (Peer-to-Peer) network and does not need any central authorities, e.g., bank [4]. Bitcoin is transferred in between Bitcoin addresses via a formatted message called transaction. The legitimacy of transactions is verified by participating users and if verified, such transactions are recorded in the everlasting ledger called *blockchain*.

Recently, more and more services accept Bitcoin as a payment method [5]. For instance, a user can buy some items with Bitcoin through an advanced vending machine [6]. Since it is quite common to pay with mobile phones in daily life, more mobile Bitcoin client apps will be available in the near future. However, at the end of January 2017, the size of Bitcoin's blockchain is more than 100 GB and thus it is not wise for storage constrained devices to keep the entire blockchain. To solve this problem, an SPV (Simplified Payment Verification) client that does not download the entire blockchain has been developed [7]. When an SPV client needs to confirm transactions that involve its Bitcoin addresses, it is necessary to tell its Bitcoin addresses to a full client. However, this can cause two problems. The first problem is that the size of a request message gets larger according as the number of SPV client's Bitcoin addresses increases. The second one is a privacy issue, because a full client may identify which Bitcoin addresses are used by an SPV client and thus his/her personal information such as purchase habit may be leaked. To solve these problems, it has been proposed that a Bloom filter that involves SPV client's Bitcoin addresses is sent to a full client [8]. A Bloom filter is a space-efficient probabilistic data structure and can be used to quickly check whether an element is contained in a group [9], while allowing some false positives. A full client checks all Bitcoin addresses involved in the transactions among blocks from the last checkpoint, to the latest, and sends transactions that only match the Bloom filter. Since the Bloom filter can generate false positives, it makes it somewhat difficult to identify Bitcoin addresses owned by the same user [8]. However, the preserved privacy level is not constant by the time, because a fixed target false positive rate is used in designing a Bloom filter. The root cause of this problem is that none of the specific privacy metrics have been considered in designing the Bloom filter for SPV clients, and thus there is no specific rule to set the target false positive rate.

In this paper, we propose a privacy-preserving Bloom filter design based on γ-Deniability for an SPV client. γ-Deniability is a privacy metric that quantifies how truly involved elements are "hidden" by the false positive ones in a Bloom filter [10]. By applying γ-Deniability into the

978-1-5386-3531-5/17/$31.00 © 2017 IEEE

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on April 15,2023 at 06:49:36 UTC from IEEE Xplore. Restrictions apply.

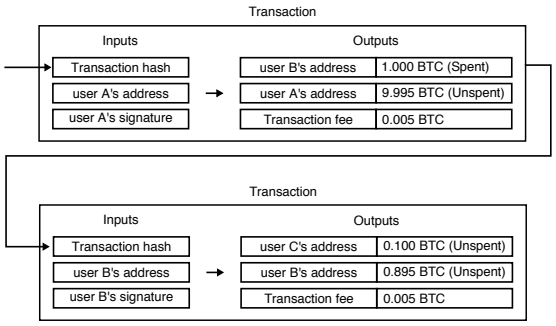


Fig. 1. An illustration of two typical blocks in the blockchain.

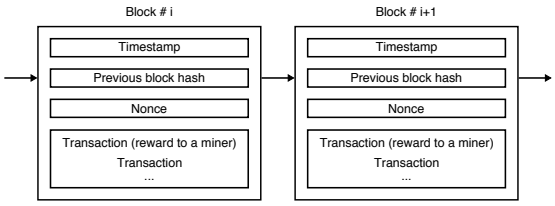


Fig. 2. An illustration of two typical blocks in the blockchain.

privacy metric of designing SPV client's Bloom Filter, the preserved privacy level can be controlled by the parameter, γ. To design a Bloom filter that satisfies a specific γ, it is necessary to know the number of unique Bitcoin addresses in blocks that appear from the last checkpoint to the latest, denoted as N_u. However, it is impossible for an SPV client to calculate N_u without possessing the entire blockchain. For SPV clients to know N_u, we propose an estimation scheme with a linear regression technique. Based on the investigation of a real Blockchain, the number of unique Bitcoin addresses that have appeared from the last checkpoint can be predicted with a simple linear regression with high probability. Although the coefficients of a linear regression model should be periodically estimated, e.g., once in week, with the latest blocks, this process can be delegated to the app developer of our scheme enabled SPV client. To show the effectiveness of our scheme, we evaluate the average estimation error rate of N_u and achievable γ value with real blockchain data. In particular, the average estimation error rate is less than 0.062 after one week from a checkpoint, when N_u is estimated with a simple linear regression model trained with transactions for four weeks. We also show that the specified γ value can be achieved when N_u is estimated by our scheme.

The remainder of this paper is organized as follows. In Section II, we introduce the fundamental of Bitcoin, SPV client and its privacy. In Section III, we describe the proposed scheme. In Section IV, we discuss the performance evaluation. Finally, we conclude the paper in Section V.

II. Preliminaries

A. The Fundamental of Bitcoin

In Bitcoin, a data structure referred to as *transaction* is used to identify the amount to be transferred, sender, and recipients. When spending Bitcoin, a user generates a new transaction of which input addresses refer to the unspent output address(es) of transactions. The user signs a transaction with a private key that corresponds to a pairwise public key specified in the unspent outputs of transactions to prove that he/she actually possesses and is willing to transfer such his/her unspent Bitcoin. Fig. 1 illustrates an example of Bitcoin transactions, where the user *A* transfers his/her Bitcoin to the user *B* and then the user *B* transfers his/her Bitcoin to the user *C*. For the user *A* to transfer his/her Bitcoin to the user *B*, the user *A*'s address calculated from the *A*'s public key, the source of Bitcoin (unspent balance user *A* owns), and its signature calculated by the user *A*'s private key are specified in the inputs part of a transaction as indicated in the first transaction in Fig. 1. At its outputs part, the user *A* specifies the amount to be transferred and user *B*'s address which is calculated from the user *B*'s public key. User *A* also specifies his/her own Bitcoin address or possibly his/her newly created address owned by user *A* to receive the change. When user *B* transfers Bitcoin to user *C*, user *B* follows the same procedure that user *A* did as illustrated in the second transaction in Fig. 1. Since any user can check every transaction through the blockchain, if the same address is used for several transactions, this could be a privacy issue. To reduce the effect of this privacy issue, it is recommended for users to generate a fresh address for every transaction.

In Bitcoin, when issuing a transaction, a user sends it to the P2P network. A broadcast transaction is then verified whether Bitcoin in inputs has not been spent ever and its signatures are valid by the P2P nodes. If found to be valid, such a transaction will be broadcast. Since Bitcoin is decentralized, a ledger that involves all the confirmed transactions must be shared by the all the users with a certain consensus algorithm. Bitcoin achieves (probabilistic) consensus with the concept of the *blockchain*, which is an ever-growing chain of blocks that contain confirmed transactions. Fig. 2 illustrates an example of blocks in a blockchain. A block is created by voluntary users, which are known as *miners*, to solve the computational puzzle that is difficult to solve but is easy to check. More specifically, miners are required to find a nonce (number used once) that satisfies its (SHA-256) hash value together with a reference to the previous block, and a set of the unapproved transactions lowers the target value. Since the previous block is required to generate the next block, this creates an ever-growing chains of blocks, i.e., blockchain. In order for the miners to follow such protocol, an incentive mechanism is adopted: The first solver of the block acquires the newly minted Bitcoin and all transaction fees included in the

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on April 15,2023 at 06:49:36 UTC from IEEE Xplore. Restrictions apply.

block as a reward.

B. SPV Client and Bloom Filter

However, at the end of January 2017, the size of Bitcoin's blockchain is more than 100 GB[1]. It is infeasible for storage constrained devices such as smartphone to maintain the full blockchain. To solve the above problem, the concept of lightweight clients so-called SPV clients was proposed [4]. SPV clients do not maintain the transactions in the blockchain but maintains only block headers. A block header is a primary identifier of a block and consists of several fields such as a previous block hash, a timestamp, a nonce, and Merkle root. An SPV client requests a full client send transactions that include its Bitcoin addresses and appear from checkpoint. A checkpoint is set when an SPV client generates a new Bitcoin address and is renewed every time a new Bitcoin address is generated. For an SPV client to receive any transactions that involve its Bitcoin addresses, it has to tell its Bitcoin addresses and the checkpoint to a full client. However, there are two problems when an SPV client sends a request to receive only transactions that its Bitcoin addresses appear. First, the size of the requesting message gets increased as the number of the Bitcoin addresses an SPV client possesses increases. Second, it has a privacy issue to naively telling their Bitcoin addresses to a full client, because it brings about privacy leakage, such as the disclosure of a purchasing habit. To solve these problems, using Bloom filter was proposed in [8]. A BIP (Bitcoin Improvement Proposal) describes a new feature to improve the functionality of Bitcoin. When an SPV client requests a full client to forward blocks that contain their Bitcoin addresses, it constructs a Bloom filter that involves its Bitcoin addresses and sends it to the connected full client instead of directly giving Bitcoin addresses. Bloom filter is a space-efficient probabilistic data structure and can be used to check whether an element is contained with allowing some false positives [9]. A Bloom filter is composed of m-bit array and m is calculated as follows.

$$m = -\frac{n \ln(P_t)}{(\ln(2))^2} \tag{1}$$

where n is the maximum number of Bitcoin addresses inserted in a Bloom filter and P_t is a target false positive rate, respectively. n is typically set as $n = n_a + 50$, where n_a is the actual number of Bitcoin addresses inserted in a Bloom filter and 50 Bitcoin addresses are considered as margin. Similarly, in some major SPV clients, e.g., [7], a fixed P_t is set ($P_t = 0.1\%$ [7]). An SPV client inputs its Bitcoin addresses into k hash functions $H_i(\cdot) : \{0,1\}^* \to [0, m-1]$, and $1 \leq i \leq k$, meaning that each hash function $H_i(\cdot)$ maps an input string x to an index of m-bit array. The number of hash functions, denoted as k, is calculated as follows.

$$k = \ln(2)\frac{m}{n} \tag{2}$$

When a full client receives a Bloom filter, it checks an SPV client's Bitcoin addresses involved in the transactions of blocks that appear from the checkpoint, and sends transactions and their Merkle tree that match the Bloom filter. Merkle tree [11] is used to check whether a specific transaction is truly involved in a specific block. A Merkle tree is a binary tree of which nodes have a hash value. The hash value of each node is calculated by hashing the concatenation of its child nodes. For an SPV client to check whether a transaction exists in a block, only its transaction, the block header, and some values in the Merkle tree are required, since an SPV client can hash them and check if the hash value matches a Merkle root involved in the block header.

C. Privacy Issue of SPV clients

Since a Bloom filter can generate false positives, utilizing a Bloom filter makes a full client difficult to identify what positive Bitcoin addresses are truly involved in a Bloom filter [8]. However, it was reported that it is not enough to preserve privacy of SPV clients by relying on the false positive of a Bloom filter [12]. If a fixed target true positive rate P_t is set, the privacy level preserved by false positives significantly depends on the number of Bitcoin addresses tested to a Bloom filter. In other words, the privacy achieved by false positives depends on not only the target false positive but also the total number of unique Bitcoin addresses invloved in blocks that appear from a fast catchup time to the latest block, which we denote as N_u. For instance, we consider a Bloom filter of which target false positive rate is set to $P_t = 0.1\%$. When $N_u = 10^5$, the Bloom filter will generate 100 false positive Bitcoin addresses on average. However, if $N_u = 100$, no false positive may be obtained. This means that it is not wise to fix P_t to a specific value without considering the trend of N_u. In other words, the current design of a Bloom filter for an SPV client does not provide a certain level of privacy. In Bitcoin, the number of unique addresses in a block steadily increases and thus N_u also increases as time goes on. Therefore, it is necessary to decide a criteria for designing the Bloom filter for an SPV client.

III. Proposed Scheme

We propose a privacy-preserving Bloom filter design for an SPV client based on γ-Deniability. γ-Deniability is a metric to measure how much Bitcoin addresses truly inserted into a Bloom filter are kept preserved by false positive Bitcoin addresses. To do so, it is necessary for an SPV client to know N_u, the number of unique Bitcoin addresses that have appeared from the fast catchup time. However, it is impossible for SPV clients to know N_u,

[1]Blockchain size is available at https://blockchain.info/ja/charts/blocks-size (Accessed on 29 May 2017.)

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on April 15,2023 at 06:49:36 UTC from IEEE Xplore. Restrictions apply.

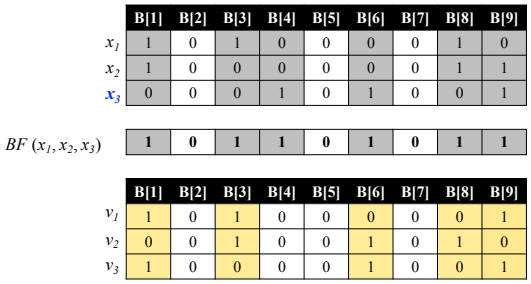


Fig. 3. An example of Bloom filter including three true positive elements and three false positive elements.

because they do not keep the entire blockchain. To address this problem, we also propose its estimation scheme by using a linear regression technique. Although the coefficients of a linear regression model should be periodically estimated, e.g., once in a week, with the latest blocks, this process can be delegated to the app developer of our scheme enabled SPV client.

In what follows, we describe (i) how γ-Deniability is applied for the construction of Bloom filter for an SPV client and (ii) the estimation method of N_u with the linear regression model.

A. γ-Deniability

γ-Deniability is a privacy metric that indicates how much true positive elements in a Bloom filter are hidden by false positive ones [10]. A Bitcoin address x inserted in a Bloom filter can be said to be deniable if $\forall i \in \{1...k\}$, there exists at least one false positive Bitcoin address v, such that $\exists j \in \{1...k\}$ s.t. $H_i(x) = H_j(v)$. A Bloom filter is said to be γ-deniable, whenever a Bitcoin address x is deniable with the probability of γ. γ of m-bit array Bloom filter is calculated as follows.

$$\gamma \approx \left(1 - \exp\left(-\frac{lk}{m(1 - e^{\frac{-kn}{m}})}\right)\right)^k \tag{3}$$

being $l = (N_u - n)P_t$ the average false positive addresses cardinality. Equation (3) can be approximated as the following equation by applying the Eqs (1) and (2).

$$\gamma \approx \left(1 - \exp\left(\frac{2(N_u - n)P_t \ln 2}{n}\right)\right)^{-\ln P_t / \ln 2} \tag{4}$$

Fig. 3 shows an example of a Bloom filter and its bit array. $< x_1, x_2, x_3 >$ are truly inserted Bitcoin addresses. $< v_1, v_2, v_3 >$ are false positive Bitcoin addresses. The Bitcoin address x_1 is deniable, because its bits $B[1], B[2], B[3]$ are covered by v_1 and $v2$. Similarly, the Bitcoin address x_2 is also deniable. On the other hand, the Bitcoin address x_3 is not deniable, because its bit $B[4]$ is not covered by any false positive Bitcoin addresses. In this example, 2 of 3 Bitcoin addresses are deniable, and thus $\gamma = 0.66$. Obviously, better privacy is provided according as γ approaches 1.

B. Estimation Scheme of N_u

In our scheme, N_u is estimated with a linear regression model. Hence, N_u can be modelled as $N_u \approx f(x) = ax^n + bx^{n-1} + \cdots$. The coefficients of $f(x)$ is calculated by using the least square method that minimizes the error against the true value of past N_u. This means that N_u is necessary to train the coefficients of $f(x)$, which contradicts the assumption that an SPV client does not possess the entire Blockchain. Actually, this step is delegated to an SPV client vendor. In this case, although it requires a periodical update of the coefficients, once in a week is typically enough as it will be later clarified.

C. The Flow of SPV Client's Bloom Filter Construction

To summarize, an SPV client constructs its Bloom filter as follows.

1) An SPV client is periodically given a fitting model of N_u by an SPV client developer.
2) An SPV client calculates the elapsed time T since the last checkpoint and estimates N_u by using the given linear regression model $N_u \approx f(T)$.
3) The array size of a Bloom filter m is calculated with N_u and the target γ by using equation (4).

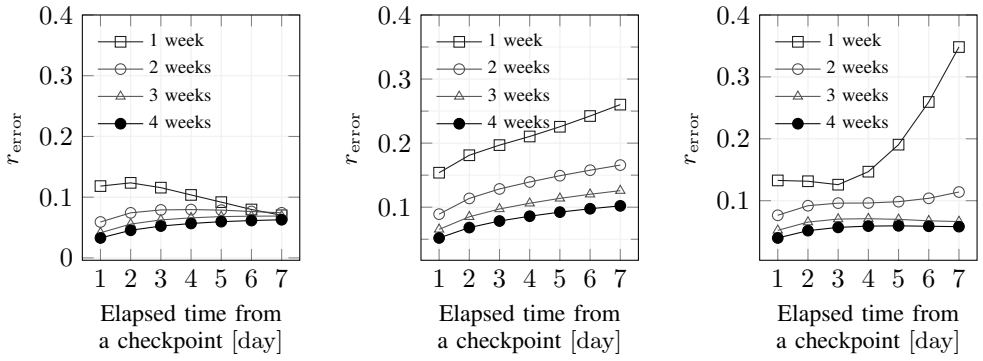
IV. Performance Evaluation

To show the effectiveness of our scheme, we measure two performance metrics, which are (i) the average error rate of predicting the unique addresses from a checkpoint to a certain time point and (ii) achievable γ. These metrics are evaluated through simulation with a real Bitcoin blockchain. The entire simulation is executed in a workstation running Ubuntu 14.04. *bitcoind*[2] and *blockparser* [13] to download the blockchain in the workstation, and to extract Bitcoin transactions and addresses, respectively. For the evaluation, the blocks from 1st January 2014 to 31st December 2015[3] are used. Both of an SPV client and full client are simulated on the workstation. For the evaluation, the following steps are simulated: At first, a checkpoint is randomly chosen from a time within the blocks. 100 Bitcoin addresses are then randomly retrieved from the blocks before the chosen checkpoint and they are assumed to be owned by an SPV client. A Bloom filter is constructed with the retrieved Bitcoin addresses. For the proposed scheme, when constructing a Bloom filter, an SPV client is given the coefficients of linear regression model to estimate N_u. Furthermore, in the proposed scheme, P_t is calculated with the estimated N_u and the given γ value. In this evaluation, γ is fixed to 0.995. In contrast, P_t is fixed to 0.1% in the conventional scheme. Every Bitcoin

[2]Available at https://bitcoin.org/en/download (Accessed on 29 May 2017.)

[3]Although we wanted to use blocks in 2016 for evaluation, our workstation failed to process blocks in 2016 due to the lack of required memory.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on April 15,2023 at 06:49:36 UTC from IEEE Xplore. Restrictions apply.



(a) $N_u \approx ax + b$ (b) $N_u \approx ax^2 + bx + c$ (c) $N_u \approx ax^3 + bx^2 + cx + d$

Fig. 4. The comparison of averaged error rate r_{error} with three linear regression models.

address in transactions in the blocks after the checkpoint is tested on the received Bloom filter to calculate the estimation error rate of N_u and the achieved γ value. The estimation error rate of N_u, denoted as r_{error}, is defined as eq (5).

$$r_{\text{error}} = \frac{|\text{actual } N_u - \text{estimated } N_u|}{\text{actual } N_u} \tag{5}$$

If not stated otherwise, the above procedures are repeated 100 times and both the performance metrics are averaged. The elapsed time since the checkpoint is varied from one to seven days.

A. Estimation Accuracy of N_u

We first discuss the estimation accuracy of N_u. We evaluated three types of linear regression models (i) $N_u = ax + b$, (ii) $N_u = ax^2 + bx + c$, and (iii) $N_u = ax^3 + bx^2 + cx + d$, where x is the elapsed time from the last checkpoint to the time when a Bloom filter is constructed. The number of weeks to find the coefficients in each model is varied from one week to four weeks. The least squares method is used to learn the coefficients. Fig. 4 shows the comparison of averaged r_{error} with three regression models. From this figure, two notable results can be seen. First, as the learning duration gets longer, the averaged estimation error gets smaller regardless of the regression models. For example, when the coefficients of the simple linear regression model (Fig. 4(a)) are trained for four weeks, the averaged r_{error} is 0.062 after seven days are passed. In contrast, one week is not enough to accurately estimate N_u. We argue that taking four weeks to learn is not a big problem, because this process is done by an app vendor and thus its cost can be typically ignored. Second, on average, the smallest error rate is achieved when N_u is estimated with the simple linear regression model $N_u \approx ax + b$. From this result, it can be said that the number of unique Bitcoin addresses that appear in blocks almost linearly increases in a short term, e.g., within a week. In the practical use case, this means that it is good enough to update the coefficients a and b once in a week, which is usually not costly. Note that future research needs to investigate whether the estimation scheme will work in the other periods.

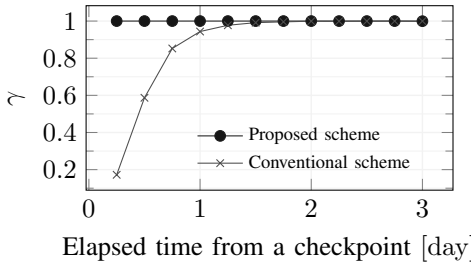


Fig. 5. The achieved γ versus the elapsed time from a checkpoint.

B. Achieved γ

We then discuss the achieved γ. In this evaluation, N_u is estimated with the simple linear regression model of which coefficients are trained with four weeks for the proposed scheme. Fig. 5 shows the comparison of the averaged γ. As can be seen from this figure, the γ of the proposed scheme is always higher than that of the conventional scheme. Obviously, more unique addresses will appear as time has elapsed. As mentioned before, this is because the conventional scheme does not consider N_u in designing a Bloom filter. Furthermore, the target false positive rate $P_t = 0.1\%$ is not suited for the evaluated block range by considering the fact that γ achieved by the conventional scheme is less than 0.6 within half a day. From this result, our proposal provides a rigid level of privacy by introducing γ-Deniability as the privacy metric.

V. Conclusion

In this paper, we have proposed a privacy-preserving Bloom filter design for Bitcoin's SPV (Simplified Payment Verification) client based on γ-Deniability. Although it has been said that introducing Bloom filter

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on April 15,2023 at 06:49:36 UTC from IEEE Xplore. Restrictions apply.

improves the privacy level of an SPV client, none of the specific privacy metric is specified. Therefore, we introduced γ-Deniability as a privacy metric and discussed how privacy-preserving Bloom filter should be constructed. Since the number of unique addresses that have appeared from the last checkpoint is required, we proposed its estimation scheme with a linear regression technique by counting N_u. From the simulation with a real blockchain data, we have clarified that the average estimation error rate is less than 0.062 after one week from a checkpoint, when N_u is estimated with a simple linear regression model trained with transactions for four weeks. We have also shown that the acheived γ value of the proposed scheme is higher than that of the conventional scheme when N_u is estimated by our scheme.

REFERENCES

[1] L. Kristoufek, "BitCoin meets Google Trends and Wikipedia: Quantifying the relationship between phenomena of the Internet era," *Scientific reports*, vol. 3, 2013.

[2] M. Van Alstyne, "Why Bitcoin has value," *Communications of the ACM*, vol. 57, no. 5, pp. 30–32, 2014.

[3] P. Ciaian, M. Rajcaniova, and d. Kancs, "The economics of BitCoin price formation," *Applied Economics*, vol. 48, no. 19, pp. 1799–1815, 2016.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[5] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten, "Have a snack, pay with Bitcoins," in *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2013, pp. 1–5.

[6] R. Jamie, "The evolution of the bitcoin vending machine," https://news.bitcoin.com/evolution-bitcoin-vending-machine/, July 2016, (Accessed on May 29 2016).

[7] "Bitcoinj," http://bitcoinj.github.io/.

[8] M. Hearn and M. Corallo, "BIP37: Connection Bloom filtering," 2012, https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki.

[9] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," 1970.

[10] G. Bianchi, L. Bracciale, and P. Loreti, ""Better Than Nothing" Privacy with Bloom Filters: To What Extent?" in *Proc. of International Conference on Privacy in Statistical Databases*. Springer, 2012, pp. 348–363.

[11] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. of IEEE Symposium on Security and Privacy (SP)*. IEEE, 1980, pp. 122–122.

[12] A. Gervais, S. Capkun, G. O. Karame, and D. Gruber, "On the privacy provisions of Bloom filters in lightweight Bitcoin clients," in *Proc. of ACM Annual Computer Security Applications Conference (ACSAC)*. ACM, 2014, pp. 326–335.

[13] M. Spagnuolo, F. Maggi, and S. Zanero, "BitIodine: Extracting intelligence from the Bitcoin network," in *Proc. of Financial Cryptography and Data Security (FC)*. Springer, 2014, pp. 457–468.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on April 15,2023 at 06:49:36 UTC from IEEE Xplore. Restrictions apply.