On The Unforkability of Monero

Dimaz Ankaa Wijaya*
Monash University
Melbourne, Australia
dimaz.wijaya@monash.edu

Joseph K. Liu+
Monash University
Melbourne, Australia
joseph.liu@monash.edu

Ron Steinfeld
Monash University
Melbourne, Australia
ron.steinfeld@monash.edu

Dongxi Liu
Data61, CSIRO
Sydney, Australia
dongxi.liu@data61.csiro.au

Jiangshan Yu
Monash University
Melbourne, Australia
jiangshan.yu@monash.edu

ABSTRACT

Monero, ranked as one of the top privacy-preserving cryptocurrencies by market cap, introduced semi-annual hard fork in 2018. Although hard fork is not an uncommon event in the cryptocurrency industry, the two hard forks in 2018 caused an anonymity risk to Monero where transactions became traceable due to the problem of key reuse. This problem was triggered by the existence of multiple copies of the same coin on different Monero blockchain branches such that the users spent the coins multiple times without preemptive action. We investigate the Monero hard fork events by analysing the transaction data on three different branches of the Monero blockchain. Although we have discovered an insignificant portion of traceable inputs compared to the total available inputs in our dataset, our analyses show that the scalability of the event depends on external factors such as market price and market availability. We propose a cheap, easy to implement strategy to prevent the problem of key reuse, should in the future stronger Monero forks emerge in the market.

CCS CONCEPTS

• **Security and privacy** → *Pseudonymity, anonymity and untraceability*;

KEYWORDS

Monero, key reuse, hard fork, traceability, anonymity, ring signature, cryptocurrency

ACM Reference Format:
Dimaz Ankaa Wijaya, Joseph K. Liu+, Ron Steinfeld, Dongxi Liu, and Jiangshan Yu. 2019. On The Unforkability of Monero. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand*. ACM, Auckland, New Zealand, 12 pages. https://doi.org/10.1145/3321705.3329823

1 INTRODUCTION

Cryptocurrency has become a global phenomenon. The first of its kind, Bitcoin, was created by Satoshi Nakamoto [22]. Bitcoin was launched in 3 January 2009 as a peer-to-peer payment system employing a shared ledger called blockchain, where a consensus method is employed such that the system is not controlled by any central party. In the published whitepaper, Satoshi describes Bitcoin as an anonymous, where no information links the public keys to the real identity of the owners [22]. However, the anonymity of the public keys is insufficient to protect the privacy of the users. Researchers have developed analysis methods to reveal information about the users from external sources [18]. The transparent blockchain data was also utilised to reveal the transaction patterns such that the users' behaviours are identified [27].

Much research has been conducted in the blockchain privacy area, where privacy-preserving cryptocurrencies such as Monero try to address the problem. Monero is one of the most successful privacy-preserving cryptocurrencies and has been one of the most valuable cryptocurrencies according to Coinmarketcap.com[1]. Monero is based on CryptoNote protocol [34], where the untraceability of the sender and the unlinkability of the receiver are protected by cryptographic methods such as Linkable Ring Signature (LRS) and one-time public key (OTPK) which makes transactions more anonymous compared to other cryptocurrencies such as Bitcoin [34]. Although anonymity features were implemented, research results show that the transactions in Monero can still be traced due to the problem of zero mixin transactions [14, 21].

In a blockchain ecosystem, a fork is an event where the protocol changes [40]. Zamyatin et al. [40] classified the changes into several possibilities: protocol expansion, reduction, conflicting (bilateral), and conditionally reduction (velvet). There are generally two types of blockchain fork, namely hard fork and soft fork. Hard fork is related to either protocol expansion or bilateral which results in a blockchain split or chain split. Soft fork, including protocol reduction and velvet, does not produce any chain split [40].

Blockchain fork in the cryptocurrency setting usually benefits the users financially [26]. For example, a user had one Bitcoin before the Bitcoin Cash fork. After the forking event happened, the same user will double his/her money: one Bitcoin in the original blockchain and one Bitcoin Cash in the newly created blockchain

---

*Also with Data61, CSIRO, Melbourne, Australia. +Corresponding author..

[1] The total market cap for Monero is US$1.7B on 26 October 2018

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329823

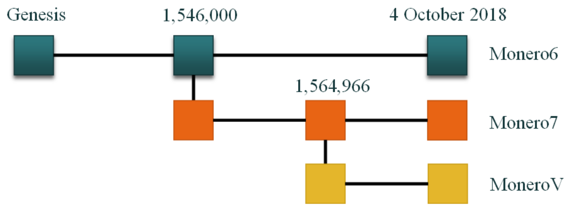

Figure 1: The Monero hard fork timeline which shows two hard forks resulting in three different chains by October 2018.

branch. The price of Bitcoin was US\$2,808 (open price)[2] while Bitcoin Cash was traded at the rate US\$555.89 (open price) each[3], so the user received a coin worth US\$555.89 for every bitcoin he/she held by doing nothing.

As a part of the attempt to keep its system updated, Monero has a regular hard fork which is scheduled semi-annually. The hard fork of Monero mostly aims to improve the privacy solutions. However, Monero also aims to become an ASIC-proof cryptocurrency, where the software developers discourage the development of ASIC machines for participating in Monero mining through the existing Proof-of-Work (PoW) mechanism [6, 34]. For this purpose, on 6 April 2018 Monero updated its mining algorithm to render existing ASIC machines' efficient computing void. The event created a new Monero fork as Monero upgraded from protocol version six to version seven.

The hard fork happened due to the incompatibility between two protocol versions. The protocol version seven started from block number 1,546,000[4] which was on 6 April 2018. There were several independent parties claiming that they still wanted to run version six protocol. These independent parties rebranded the blockchain system running Monero protocol version six into three different names: Monero Original (XMO), Monero 0 (Monero Zero or ZMR), and Monero Classic (XMC)[19]. Although there are three different names, in reality there is only one blockchain [2]. After the Monero hardfork, another project was also forked from the Monero version seven, called MoneroV (XMV), starting from block number 1,564,966[5] which was on 3 May 2018. MoneroV runs a modified version of protocol seven. One of the modifications on MoneroV's protocol from Monero version seven is the reduction of the decimal places from 12 to 11. This causes all coin values to be multiplied ten fold. The fork history is shown in Figure 1.

Similar to other blockchain fork events, the Monero users doubled their Monero coins on each event, where they receive the same amount or even more coins in the newly created cryptocurrencies. However, when the users decide to use the same coins without extra caution, the anonymity of the transactions is potentially reduced.

The traceability could be compromised in a Monero hard fork event. The hard fork event creates new coins which will be spendable by the pre-fork users. The problem occurs when the Monero users spend the newly created coins, such that the traceability of the transaction's sender can be deduced. We denote this as the problem of key reuse. The problem of key reuse reduces the anonymity of a transaction because the users reuse the same keys when spending the same coins. This traceability problem was known to Monero developers [6], where the existing strategies are not sufficient to completely mitigate the problem.

Contributions. We summarise our contributions as follows.

- We investigate the impact of two Monero hard forks which occurred in 2018. We collected the data of each blockchain branch and cross-reference the transaction data on all of the blockchain branches to determine traceable inputs. We discover over 55K distinct traceable inputs, where 90% of them are found on two blockchain branches running version six and seven. About 19% of all inputs in Monero version six of our dataset are traceable. However, the percentage of the traceable inputs is insignificant compared to the total inputs in Monero version seven (or the main branch). Unlike existing attacks [14, 21, 35, 37, 39], the analysis on the problem of key reuse is able to identify traceable inputs in Monero's RingCT. None of the traceable inputs of our findings can be discovered by existing attacks.
- We analyse the correlation between the identified traceable inputs and market prices by using statistical analyses, namely correlation coefficients and linear regression. The results on the correlation coefficients show that there is a strong correlation (0.553 to 0.761) between the two factors in Monero6, whereas a weak correlation (0.242 to 0.32) is found in MoneroV. Likewise, the linear regression results show a medium relationship in Monero6 but a small relationship in MoneroV. A bigger support from cryptocurrency exchanges to Monero6 compared to the support to MoneroV also explains the huge gap of traceable inputs on both blockchain branches. These findings also show that the scalability of the problem of key reuse depends on the identified external factors.
- We propose a mitigation strategy for the problem of key reuse caused by hard forks in Monero. `Scalable Bloom Filters` [1] provide an inexpensive checking mechanism of existing key images and mixins. By adding the checking mechanism prior submitting new transactions to the network, transactions spending the same coins will maintain the same level of anonymity. A new service node called `joint node` is proposed such that the mitigation strategy can be deployed in the current Monero protocol without any significant changes.

Organisation. The rest of the paper is organised as follows. Section 2 contains technical background about Linkable Ring Signature (LRS) and Monero protocol. In Section 3, related work about existing anonymity attacks to Monero, an alternative strategy to hard fork, and replay protection. A threat model is presented in Section 4. Section 5 describes our analysis methods. Section 6 covers current mitigation strategies to the identified problems, where our mitigation strategy proposal is also presented in this section. Section 7 provides discussions on the security and the performance of the proposed mitigation strategy, while Section 8 concludes the findings and describes future work.

---

[2] https://coinmarketcap.com/currencies/bitcoin/historical-data/?start=20170723&end=20170723

[3] https://coinmarketcap.com/currencies/bitcoin-cash/historical-data/?start=20170723&end=20170723

[4] https://github.com/monero-project/monero

[5] https://github.com/monerov/monerov

2 BACKGROUND

2.1 Linkable Ring Signature

Linkable Ring Signature (LRS) [15, 16] was developed based on Ring Signature (RS) technique. LRS enables the users to take advantage of RS. In RS, a signer cannot be identified over a set of potential signers even if the signer creates many signatures, but in LRS, the signature creator can only create one signature for every private key she holds if she wants to maintain the signature anonymity. A tagging system is employed in LRS. If the same signer reuses the same private key to create more than one signature, then a verifier will be able to link the signatures and determine that they are created by using the same private key, because the same tag will appear in both signatures. As the same tag is only known by the signer, this implies that the signatures are created by the same person. LRS limits the anonymity in RS. However, this limitation is useful in many scenarios such as e-voting [5, 38] and cryptocurrency system [31], where a double-vote or double-spending of a coin should be prevented.

2.2 Monero Transaction Structure and Protocol

The structure of a Monero transaction is shown in Figure 2. An input I contains a ring signature R which uses of an output set O of size r. The output set O contains an output to be spent $o_i \in O$ and $r - 1$ outputs as decoys. The decoys are usually chosen at random from the public blockchain B such that the probability of guessing the real output being spent is not more than $^1/_r$. A Monero transaction can produce zero or more new outputs which will then be added to the output database in the blockchain B. These new outputs will be available to be spent or can also be chosen as decoys in the next transactions.

Each output o in the blockchain B is associated with exactly one secret key called `key image`. A key image k_i needs to be published in order to spend an output o_i, however, by using a ring signature scheme which contains a set of r outputs $O = \{o_a, \ldots, o_i, \ldots\}$, the output o_i is indistinguishable from the decoys. An output cannot be double-spent in the same blockchain. To enforce the rule, the spent output's associated key image k_i is recorded in the blockchain. If a new transaction is reusing the same key image k_i, the system will be able to detect the double spending effort and reject the new transaction.

3 RELATED WORK

3.1 Monero Traceability Analyses

Monero implements anonymity technologies as its main features over other cryptocurrencies such as Bitcoin. Linkable Ring Signature (LRS) provides the untraceability of the sender. Ring Confidential Transaction (RingCT) [23] was implemented to support encryption of the amount of coins being transacted from the payer to the payee. One-time Public Key (OTPK) provides the unlinkability of the receiver by forcing the sender to create a new destination address for every new transaction on behalf of the receiver [34].

Despite the implementation of the mentioned features, researchers developed analysis techniques to determine traceable inputs. LRS uses decoys (the decoys are also called mixins) in the signature to avoid detection of the real signer. Analyses showed that zero mixin transactions remove the untraceability of the sender [14, 21]. Although the weakness has been patched by not allowing transactions without mixins, researchers found new ways to discover the traceability of the sender through crafted transactions [35, 37] or Monero's Payment ID [37]. Closed set attack was introduced to determine spent coins in Monero [39].

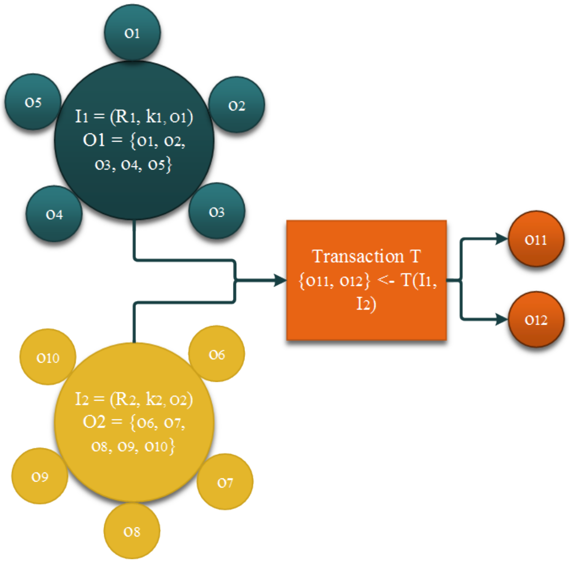

Figure 2: Input-output construction in a Monero transaction. In a transaction, a set of inputs containing multiple existing outputs produce a set of new outputs.

3.2 Velvet Fork

Velvet fork is a term coined by Kiayias et al. [12]. Rather than having a hard fork which is considered as a risky event, the velvet fork offers a new strategy where variables replace constant parameters when changing the protocol. By using velvet fork, the hard fork can be avoided. The past occurrence of velvet fork was further investigated [40].

Although velvet fork is useful to avoid a hard fork that changes parameters, the velvet fork is unusable when the change is in the protocol layer. If the velvet fork technique is applied to Monero, then increasing ring size would be extremely easy by modifying a prepared variable holding the ring size value. However, protocol layer changes such as modifying decoy selection method, adding new signature features [30], or changing consensus method [6] cannot be done by using velvet fork.

3.3 Replay Protection

Replay protection is a mechanism to avoid replay attack. The term replay attack in cryptocurrency refers to retransmission of a valid transaction data on a cryptocurrency to other compatible forks of the cryptocurrency [17]. The result of the replay attack is that the payee will get multiple payments in different cryptocurrencies such that the payer suffers loss. The paper by McCorry et al. [17] describes several examples of replay protections being implemented in different cryptocurrencies, such as `Chain ID`, `Transaction Version`, `Check Block At Height`, and `Sighash Enum`.

`Chain ID` has been implemented in Ethereum since `Spurious Dragon` hard fork [4], while `Transaction Version`, `Check Block At Height`, and `Sighash Enum` were proposed as alternative solutions for replay protection in Bitcoin [17]. A new proposal introduced new replay protection methods, namely `Migration Input` and `Hardfork Oracle` [17]. `Migration Input` was proposed to be implemented in Bitcoin protocol, where the input hash is modified from 32 bytes to 41 bytes to accommodate extra information. By having a different input hash scheme, the old protocol will not be able to validate the transaction, and therefore only the new compatible protocol validates the transaction [17]. `Hardfork Oracle` was proposed to be implemented as an Ethereum smart contract which is used for automatic detection of transactions from different forks.

There is no replay protection currently being implemented in Monero protocol [25]. One of the ways to create a transaction with a built-in replay protection is to include at least one output that can only be found in the intended chain as one of the decoys in the ring signature [32].

3.4 Attacks on Monero Protocol Update

The asynchronous protocol update (i.e hard fork) on Monero can also lead to attacks [36]. A research discovered that the nodes that have not yet updated their applications to the latest version (which run the latest protocol version) are prone to Denial of Service (DoS) attack, where a large number of transactions can be created to flood the nodes' temporary storage *txpool*.

Furthermore, the DoS attack can be utilised to "announce" the traceability of the inputs to public by submitting two different transactions that spend the same coins, one transaction submitted to the old nodes (which run the old protocol) and another to the new nodes (wich run the new protocol). If the required condition of the network holds, then the transactions that were sent to the old nodes will never be confirmed to the network, hence the double spending will never occur. However, since two coins appear twice in different nodes (which run different protocols), then the real inputs of the related transactions can be deduced.

4 THREAT MODEL

The untraceability in Monero requires that an observer cannot guess the real output being spent in a ring construction R with a probability of more than $\frac{1}{r}$, where r is the number of ring members. In this case, the anonymity of the real output depends on the size of r.

We define an input I_j traceable to an output o_j as follows. The output o_j is an output of a blockchain B_1 in a linkable ring signature R_1 with a set of output $O_1 = \{o_a, \ldots, o_j, \ldots, o_l\}$ as its ring members and key image k_j, such that R_1, k_j, and O_1 are parts of I_j. The same output o_j appears on another blockchain B_2 which is included in another linkable ring signature R_2 with a set of output $O_2 = \{o_m, \ldots, o_j, \ldots, o_z\}$ and key image k_j. From this occurrence, it can be concluded that the key image k_j is associated to the output o_j. Therefore, the input I_j is traceable to the output o_j, because the probability of guessing the real output is 1. This occurrence also fulfills the linkable condition as described in Linkable Ring Signature [16] as it can be concluded that the two ring signatures R_1 and R_2 are created by the same person, assuming that the secret key image k_j is only known to the owner.

Anonymity reduction occurs when q members of a ring R can be deduced since they are no longer fit as the candidate when guessing the real output, therefore the anonymity r is reduced by q. We define a reduced anonymity input I_i as an input of a blockchain B_1 in a linkable ring signature R_3 with r as the ring size using a set of output $O_3 = \{o_b, \ldots, o_e, \ldots, o_i, \ldots, o_m\}$ and key image k_h. The same key image k_h was found on a ring signature R_4 recorded on another blockchain B_2 using a set of output $O_4 = \{o_c, \ldots, o_e, \ldots, o_i, \ldots, o_n\}$ where at least two outputs $\{o_e, o_i\}$ in R_4 fulfill the following criteria: $\{o_e, o_i\} \in O_4$ and $\{o_e, o_i\} \in O_3$. In this scenario, it is inconclusive whether key image k_h is associated to o_e or o_i. The example of a traceable output and an anonymity reduction is shown in Figure 3.

We define passive attack and active attack in Monero. In the passive attack, an attacker collects information from the public blockchain(s) and conduct traceability analyses. In the active attack, the attacker controls dishonest nodes which return false information. By returning false responses of key image-related requests, the dishonest nodes expect that the client suffer anonymity reduction from the problem of key reuse, especially when a client spends the same coins in different blockchains.

We assume that all cryptocurrency backers, including software developers, community members, and users, desire the best privacy-preserving features for their systems. However, there also exist some users in the system who unknowingly spend identical coins in multiple blockchains such that the traceability of those transactions are revealed. The events cause "cascade effect" which make other transactions traceable or suffer anonymity reduction [21]. It is also assumed that the majority of the nodes in the system behave honestly by sending the correct responses or information from any requests.

It is assumed that the current system can only accept small modifications which do not greatly affect how the whole protocol is run. However, it is also assumed that hard forks can occur at any given time such that the same unspent coins before the fork can be spent multiple times on different blockchains after the fork. Conducting a hard fork is incentivized financially as the newly created coins can be sold in the markets.

5 ANALYSES

5.1 Analysis on Traceable Inputs

We collected all transaction data on three different blockchains, which we called Monero6, Monero7, and MoneroV. We use the term Monero6 to refer cryptocurrencies that are using the Monero protocol version six: Monero Original, Monero 0, and Monero Classic. Monero7 is used to refer the Monero main blockchain which runs protocol version seven (as of October 2018). MoneroV is a self-explanatory, referring to the blockchain system which runs MoneroV protocol.

A checking algorithm was constructed as follows.

(1) An array K_{res} is initialised. It will be used to store the identified key images as the final result.
(2) For each key image $k_i \in K$ do the following steps.

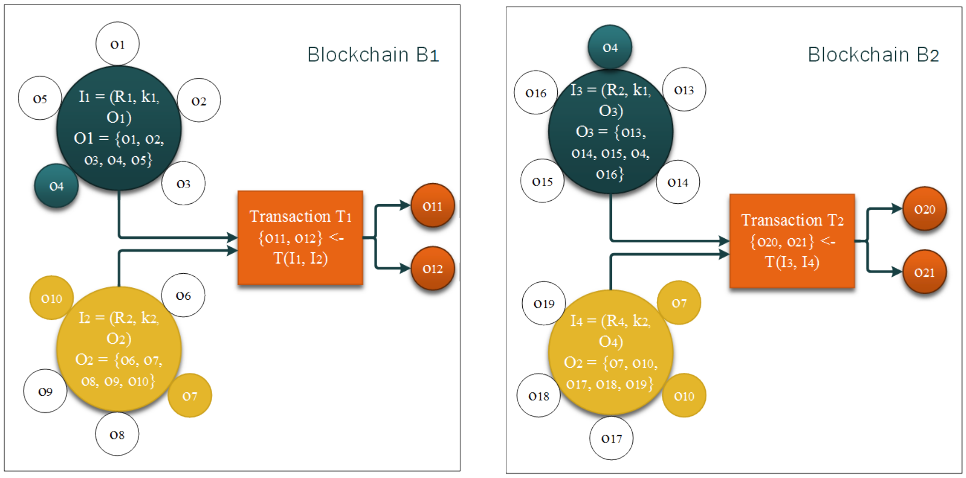

Figure 3: The inputs I_1 and I_3 have two common similarities: the key image k_1 and the output o_4. Both inputs I_1 and I_3 are traceable. The inputs I_2 and I_4 contains the same key image k_2, however there are two identical outputs in the ring, namely o_7 and $o_1 0$. Both inputs I_2 and I_4 suffer anonymity reduction by three.

   (a) Compute the total number of its occurrence in all three blockchains and store the result in a variable, *occ*.
   (b) Compute the number of unique transaction hash (*txhash*) and store the result in another variable, *unq*.
   (c) Execute a conditional statement as follows: $if\ (occ > 1)$ and $(unq > 1)$ then $K_{res} \leftarrow k_i$. The conditional statement is required to filter out key replay cases as they do not help identifying the real output to be spent.

(3) Return K_{res}.

We have examined three blockchains by using the algorithm above on a dataset we built by extracting non-coinbase transctions (transactions that are not block reward) confirmed on block number 1,546,000 to 1,675,606 in *Monero6* (181 days period), block number 1,546,000 to 1,675,303 in *Monero7* (181 days period), and block number 1,564,966 to 1,671,617 in *MoneroV* (152 days period). The cut-off period for the data extraction is 4 October 2018. We also have conducted a cascade effect analysis to determine how many traceable inputs as the impact of the problem of key reuse on the same dataset. The result is presented in the Table 1.

Based on the algorithm we developed, we discovered 52,924 traceable inputs on Monero6 (including the ones discovered using cascade effect method), 53,477 traceable inputs on Monero7, whereas there are only 7,542 traceable inputs found on MoneroV. Although there are only 29 days difference between MoneroV and both Monero6 and Monero7, the traceable inputs found on MoneroV is only around 14% of traceable inputs found on both Monero6 and Monero7. There is also an extreme difference on the number of non-coinbase transactions between Monero7 and the other two cryptocurrencies. Monero7 has 810,409 transactions, where Monero6 and MoneroV only contains 5% and 10% of Monero7's transactions respectively.

The traceable inputs in Monero6 are 19% of Monero6's total number of inputs in our dataset, whereas the traceable inputs in Monero7 and MoneroV are only 2% of their total inputs. This shows that the problem of key reuse has a more significant impact on Monero6 than on Monero7 and MoneroV. Also, about 90% of all traceable inputs are found among Monero6 and Monero7, whereas only 6% of the traceable inputs are found on all three blockchain branches. This shows that Monero6 is the greater source of the problem of key reuse to Monero7 than MoneroV to Monero7.

5.2 Analysis on Anonymity Reduction

We also examined reduced anonymity as the side effect of the traceable inputs and cascade effect. An input with a ring size of r is deemed to suffer anonymity reduction if there are at most $r-2$ ring members that have been identified to be spent on other transactions. By suffering reduced anonymity, the real output is still untraceable. However, the probability of guessing the real output increases from $\frac{1}{r}$ to $\frac{1}{r-u}$, where $1 \leq u \leq (r-2)$ and u is the number of outputs that have been known to be spent in other transactions.

We discovered 1,848 inputs in Monero6 that suffer anonymity reduction. Likewise, 2,819 inputs in Monero7 and 264 ring signatures in MoneroV suffer anonymity reduction while still being untraceable. Figure 4 shows the trend of the anonymity reduction on all three blockchain branches. About 95% of the anonymity reductions have a reduction size u between 1 to 5.

The result shows that a transaction having a ring size of 5 or less is riskier than one having a ring size of more than 5. We calculated the average ring size for the three blockchain branches starting from the first block of the fork to the cut-off period on 4 October 2018, where Monero6, Monero7, and MoneroV have an average ring size r of 5.07, 7.56, and 7.75 respectively. Having a bigger ring size provides a better protection on the anonymity of the input, although creating a transaction with a bigger ring size may result in a more expensive transaction fee to be paid by a user. Determining a bigger minimum ring size in the protocol level can also be helpful to ensure that the users have a sufficient anonymity in their transactions. While Monero6's minimum ring size was five, Monero7 and MoneroV has a minimum ring size of seven. Based on the result, it is advisable to have ring size $r > 5$.

Table 1: The summary of traceable inputs from the problem of key reuse and the cascade effect it caused.

Blockchain	Height	Key Reuse		Cascade Effect		Dataset	
		Traceable Input	Tx.Count	Traceable Input	Tx.Count	Input Count	Tx.Count
Monero6	1,675,606	52,646	3,148	278	276	274,131	44,467
Monero7	1,675,303	53,162	5,680	315	312	1,876,341	810,409
MoneroV	1,671,617	7,542	888	0	0	269,335	84,053

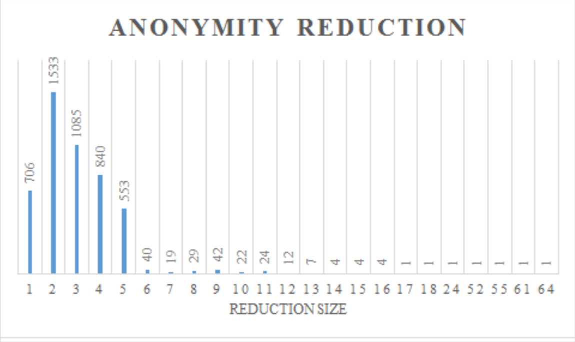

Figure 4: The summary of anonymity reduction as the result of key reuse problem on Monero6, Monero7, and MoneroV.

5.3 Analysis on Key Reuse and Cryptocurrency Market Price Correlation

We collected historical prices of Monero Classic and Monero Original from Coinmarketcap.com (20 April 2018 to 3 October 2018), while the prices of MoneroV were gathered from Coingecko.com (4 July 2018 to 3 October 2018). We calculated statistical data regarding the market prices for Monero Classic, Monero Original, and MoneroV. The summary of the market prices can be found on Table 2. We identified that while the prices of Monero Classic and Monero Original are in the average of US$4 (lowest at US$1 and highest at US$27), the average price of MoneroV is extremely low, which is at US$ 0.06 (lowest at US$0.02 and highest at US$0.26). If the price of MoneroV is multiplied by ten, then the MoneroV price only becomes US$0.6 in average, whereas the average price of Monero Classic and Monero Original is seven times more expensive.

Based on the price information of Monero Classic (price at high, low, and close), Monero Original (price at high, low, close), and MoneroV (price at open and close), we studied the correlation between market prices and traceable input count as well as transaction count by computing **Kendall's tau-b correlation coefficient** and **Spearman's rho correlation coefficient**. The result is presented in Table 3.

Cohen as cited by Sauro and Lewis [28] provided three interpretations of correlation coefficient r as follows: r is small when $0.1 \leq r \leq 0.3$, medium when $0.3 \leq r \leq 0.5$, and large when $r \geq 0.5$. Our results show that all coefficients for both Monero Classic and Monero Original are in the range of 0.557 and 0.754, which indicates a strong correlation between the price and key reuse in Monero Classic and Monero Original. On the other hand, a small correlation between the price and key reuse in MoneroV can be summarised according to the result, where the coefficients are 0.242 and 0.32. Based on this information, we deduct that the market price and the number of traceable inputs are correlated, although the causality between the two parameters cannot be determined from the statistical analyses performed.

The linear regression analysis of the dataset is shown in Figure 5. The R-squared values are 0.180, 0.146, and 0.003 for Monero Classic, Monero Original, and MoneroV respectively. The result shows a medium relationship between the number of traceable inputs and the market price on both Monero Classic and Monero Original, but a small relationship among the two variables on MoneroV.

5.4 Analysis on Key Reuse and Coin Availability

At the time of writing, cryptocurrency portal Coinmarketcap.com lists Monero Classic (XMC) and Monero Original (XMO) as two separate coins, although in reality these two cryptocurrencies are in the same blockchain branch. Each of these cryptocurrencies are being traded in different cryptocurrency markets. Monero Classic[6] is traded at:

- Gate.io (XMC/USDT pair and XMC/BTC pair)
- HitBTC (XMC/BTC pair, XMC/USDT pair, and XMC/ETH pair)
- TradeOgre (XMC/BTC pair)

Monero Original, on the other hand, is only available at HitBTC (XMO/BTC pair, XMO/ USDT pair, and XMO/ETH pair). The other Monero fork, MoneroV (XMV), is only available at TradeOgre (XMV/BTC pair). On 4 April 2018, two days before Monero6 fork, the top six most busy cryptocurrency exchanges serving Monero are HitBTC, Binance, Bitfinex, Poloniex, Kraken, and Livecoin, which made 84.8% of the total daily Monero trading volume [37]. HitBTC has the largest portion of 37.68%, while Livecoin had the lowest portion of 3.82% among the six exchangers [37]. Although the Monero pre-fork trading information history in TradeOgre is unavailable, it can still be deducted that the majority of the cryptocurrency markets did not support MoneroV, therefore the Monero users never received MoneroV from the cryptocurrency exchanges. On the other hand, as Monero6 was supported by big exchanges such as HitBTC, it is assumed that the number of Monero6 coins distributed to the users were larger than MoneroV coins.

The market availability is likely to be one of the factors that cause the number of traceable inputs on MoneroV is only 14.3% of Monero6's traceable inputs. However, it is not possible to determine how much coins were spent in the traceable inputs due to the

---

[6] Information about Monero Classic and Monero Original is taken from Coinmarketcap.com, while information about MoneroV is taken from Coingecko.com.

Table 2: The summary of market prices of Monero Classic, Monero Original, and MoneroV.

	Monero Classic				Monero Original				MoneroV	
	Open	High	Low	Close	Open	High	Low	Close	Open	Close
Max.Price	21.99	27.42	18.02	21.55	21.81	24.36	14.88	20.75	0.26	0.26
Min.Price	1.20	1.29	1.04	1.20	1.25	1.29	1.22	1.23	0.02	0.02
Avg.Price	4.38	4.84	4.04	4.34	4.24	4.64	3.92	4.21	0.06	0.06

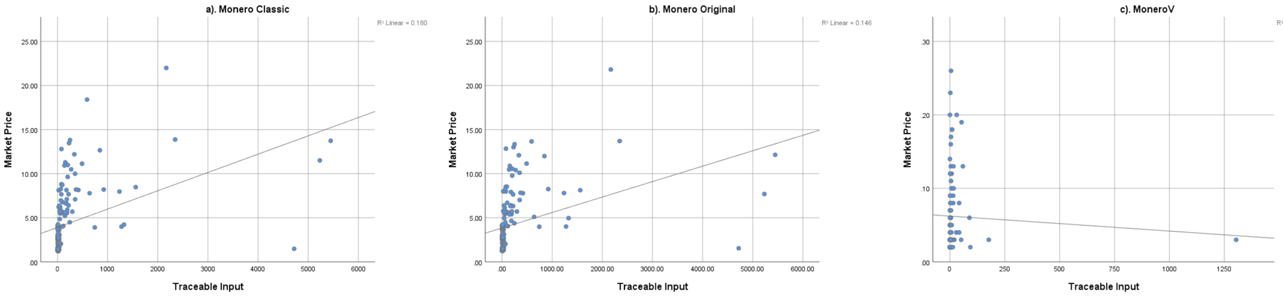

Figure 5: The linear regression of the number of traceable inputs and market price of Monero Classic (figure a), Monero Original (figure b), and MoneroV (figure c).

Table 3: Correlation between the number of traceable input and market price (open price) of Monero Classic, Monero Original, and MoneroV

	Correlation	
	Kendall's Tau-b	Spearman's Rho
Monero Classic	0.561	0.755
Monero Original	0.557	0.754
MoneroV	0.242	0.32

implementation of `RingCT` which obfuscates the amount of coins involved in the transactions.

We also made several assumptions as follows.

1. The users will be given free coins after a hard fork, as long as they hold an amount of coins on the original blockchain branch before the hard fork [9–11].
2. The users prefer to get a maximum benefit by selling the coins at a high price.
3. If the users are using private wallets (either by using a computer wallet, smartphone wallet, web wallet, or paper wallet), then the users need to setup new wallet applications for the new chains and import the information from their old wallet before they can create transactions.
4. If the users deposit their coins to cryptocurrency exchanges' wallets before the hard fork, then it depends on the exchanges whether they are supporting the new coins; only then the users will receive the new coins from the exchanges which will be credited to their accounts on those exchanges [9–11].

The assumption 1 and 2 motivate the users to redeem their new coins, hence, the problem of key reuse potentially occurs. However, assumption 3 becomes a barrier for the users, as setting up new wallets by importing private keys from old wallets is not trivial and requires a technical knowledge. In assumption 4, when the cryptocurrency exchanges do not support the forked coins, then it is not possible for the users to claim the newly created cryptocurrencies.

6 MITIGATION STRATEGIES

In this section, existing mitigation strategies for the problem of key reuse are discussed. A new mitigation strategy is also proposed.

6.1 Current Mitigation Strategy

6.1.1 Not claiming the coins. There is a suggestion for Monero users not to use the newly created coins they received for free [29]. The amount of coins received by the users vary depending on the new systems. Monero6 provides 1:1 coin distribution. This means each Monero holder receives the exact amount of coins on the new blockchain branch. MoneroV offers 1:10 coin distribution, in which the Monero holders receives ten times the amount of Monero they have before the fork occurrs.

There are two identified problems in this method. Firstly, the method is not preferable by the users, since they may lose potential profits by not claiming coins. The free coins can be traded for other cryptocurrencies or even local currencies. The suggestion is even more unlikely to be followed by the users when the amount of profit they can get by spending the coins is substantial. Secondly, the method can only be effective if and only if all Monero users do not redeem their new coins. If any of the Monero users redeem the new coins, the redeeming transactions can potentially reduce or even completely remove the anonymity of other users' transactions which make their transactions traceable.

6.1.2 Churning. Churning technique expands the output selection by sending the users' coin to their own addresses multiple times [13].

By churning the coins, the number of ring members or decoys will expand, which in turn makes it harder for an attacker to determine the real outputs being used from multiple transactions created during the churning process to trace the transactions. However, churning technique is still prone to known attacks such as timing attack and network attack which deduct real outputs based on the origins of the transactions [29]. The effectiveness of churning to mitigate the key reuse problem is also questionable [29].

6.1.3 Blackball tool. The Monero developers created a tool called `blackball`. `Blackball` is a term originally coming from a document that describes the first identified problem within Monero system, where an adversary tries to add his/her own outputs to the blockchain by creating as many transactions as possible [24]. The outputs controlled by the adversary is called `blackballs` or `black marbles`, while other outputs created by genuine users are `white balls` or `white marbles`. Whenever the blackballs are included in a transaction t as members of the mixins, then the adversary will be able to determine his/her `blackballs` as decoys and not the real outputs being spent. Hence the anonymity level of the affected transaction t is reduced.

The `blackball` application created by the Monero developers tries to blacklist known bad outputs; all outputs in the blacklist are discouraged to be used by the users as mixins. The application is not mandatory to be run by the users; the application was released as a standalone application and it was not integrated in the official Monero Node application nor Monero Wallet application as a part of Monero protocol.

The problem with the `blackball` tool is that the users need to have a full copy of each blockchain branch, which then the tool will compare and extract the information from the blockchain branches. However, assuming one blockchain branch requires $50GB$ of storage space, then three blockchain branches will require at least $150GB$ of storage space, just to keep their anonymity level intact. Clearly, the `blackball` cannot be run on light hardware such as smartphones, where space and computing power is limited. Hence it is unlikely that all users can run `blackball` tool to make their transactions safe. Other than the aforementioned `blackball` tool, there are features in the default/official `CLI`-based Monero wallet which can be used to reduce the problem of key reuse.

- To let users to manually set the mixins/decoys themselves. This feature is implemented in `set_ring` command [33].
- To only use mixins that exist before fork. This feature is implemented in `segregate-pre-fork-outputs` command [33].
- To combine mixins from before fork and after fork. This feature is implemented in `key-reuse-mitigation2` command [33].

Having identical mixins in transactions that are being published in multiple blockchain systems will prevent the passive attacker to trace the transactions, because the anonymity level is not compromised. However, there is no sufficient solution to help the users conducting the best practice in maintaining the anonymity of their transactions.

6.2 Our Proposed Solution

Our proposed solution consists of three parts, namely hard fork management, key image management, and joint nodes.

6.2.1 Hard Fork Management. We propose to add `Chain_ID` information in every transaction, which will be useful for several reasons. Firstly, the `Chain_ID` will be used for replay attack prevention, a feature that does not exist in Monero yet. Secondly, the `Chain_ID` will be an identifier when the users want to get information about outputs (when they want to create new transactions) or existing key images (which will be further described in the solution).

To complement the `Chain_ID`, a `Fork_Point` information also needs to be managed. `Fork Point` is the first block height of a new chain that has a different block hash compared to its parent, which is similar to Ethereum's `FORK_BLKNUM` [4]. Unlike `Chain_ID`, the `Fork_Point` does not need to be embedded to transaction data. The reason for not embedding the `Fork_Point` in the block or in the transaction is to save space from less useful information. For this requirement, a new database called `Chain_Info` will be created. The new database contains both `Chain_ID` (as the primary key) and `Fork_Point`.

Chain_Info. New chains produced from hard forks will need to be registered in the database in a `First-Come-First-Serve` basis. The `Chain_ID` can be used to query the `Fork Point` from the newly created `Chain_Info` database. The `Chain_Info` database will be stored in the node's blockchain database file. This approach is different compared to Ethereum's method which stores the `Chain_ID` information on a Github page [4], while Monero stored the history information of its own hard fork by *hardcoding* it to the source code[7]. However, this approach will be infeasible when dealing with external hard forks, where the occurrences might not be known to the Monero developers and Monero community.

6.2.2 Key Image Management. We have identified several issues that need be addressed in relation to managing key image information, including:

1. Multiple blockchain branches having different block interval.
2. Multiple transactions submission having identical key images in a short period of times
3. Updatability of the key image ring members (e.g. member addition).
4. New transactions with ring members that are not available in the parent chain where the key images were first recorded.
5. New chains that are designed to have a lower mandatory ring size compared to the parent chain.

Scalable Bloom Filter. Scalable Bloom Filters [1] is proposed to solve the identified issues. Scalable Bloom Filters (`SBF`) is an extended version of the original Bloom Filter [3] where scaling is enabled by utilising multiple Bloom Filters instead of a single filter as in the standard Bloom Filter (`BF`). Therefore, the capacity in `SBF` can be expanded after initialisation, contrary to BF which cannot exceed the predefined capacity. Similar to BF, an `SBF` can produce false positive results where the filter detects that a data is in a set where it should not. However, `SBF` also inherits the characteristics of BF where false negative is negligible. False negative is when the

---

[7] https://github.com/monero-project/monero/blob/master/src/cryptonote_core/blockchain.cpp#L120

BF returns `False` (that the data is not in the set) when it is supposed to be `True` (the data is actually in the set).

In our proposed solution, several SBFs are introduced. The first SBF is used to filter key images, namely SBF_k. Key images from related blockchains (parents, siblings, or child chains) will be included to compute SBF_k. By constructing SBF_k, new key images can be identified whether they have existed in any blockchains, such that when the checking algorithm result is true, then the protocol raises a flag to avoid the problem of key reuse. The second SBF is used to filter hash values of key image-mixin tuples, namely SBF_m. Similar to SBF_k, SBF_m is constructed by collecting key image-mixin tuples from all related blockchains. The purpose of SBF_m is to help the system to identify whether an incoming key image-mixin tuple has existed in one or more blockchains.

Despite its scalability feature, SBF does not support data deletion. Therefore, to mitigate different block intervals and block reorganisation where immature blocks can still be replaced by other stronger blocks, temporary *SBFs* are introduced. These temporary *SBFs*, namely $tSBF_k$ and $tSBF_m$, are associated to key images and key image-mixin tuples respectively. By using temporary *SBFs*, it means new information in immature blocks and memory pools will not go directly to the main *SBFs* but to *tSBFs*. After the information is confirmed in mature blocks, the data can be stored to the main *SBFs*. SBF_k and SBF_m can complement each other by the following mechanism.

1. The system checks a key image value in SBF_k. If it does not exist, then checking process complete, otherwise continue. In this step, it is not known whether the checking result is a false positive result or a genuine result.
2. We define t as the threshold to be satisfied by new transactions.
3. For each ring signature R with a ring size r, there will be r key image-mixin tuples. The system checks every key image-mixin tuple if they exist in SBF_m and count the positive results p. If $p = r$, then there is a possibility (due to SBF's false positive characteristic) that the input has the exact same ring members as the existing input, and this is a desirable occurrence. However, that might not always be the case. It is possible that $p < r$, but as long as p can satisfy the threshold t where $t > 1$, then $t <= p <= r$ must be satisfied. Otherwise:
   (a) If $t = 0$ then the transaction that contains the ring signature R can be accepted as it is a false positive caused by SBF_k.
   (b) If $t = 1$ then the transaction that contains the ring signature R can be blacklisted as it can cause traceable output.
4. To increase the probability of the new transactions using an identical set of the existing ring members, the threshold t can be set to $t = r$ such that $t = p = r$.

Blacklisting can be used as an option instead of rejecting the transaction, as described in the step 3b, because transaction rejection might motivate users to recreate the transaction which will make the new transaction traceable [20].

False Positive. The false positive rate is the trade-off for not using the real transaction data in our solution, where SBFs are used for a cost-efficient solution. The error rate in the original design of SBF is expected to be between 0.0001% to 0.1% [1]. The use of two different *SBFs*, namely SBF_k and SBF_m is expected to greatly reduce the false positive rate in the case of a new key image that has never been spent, such that the false positive result indicates otherwise.

We utilise a simple equation of probability of two independent events $P = p_1 \times p_2$ where p_1 and p_2 are the probabilities of the first and the second event, respectively. By using the equation and taking the largest error rate of SBF, the probability of a key image that has never been spent to be detected as false positive in both checks is 0.0001%.

SBF for Multiple Blockchain Branches. An SBF consists of multiple Bloom Filters (BF) [1] where $SBF = \{BF_1||BF_2||\dots\}$, where the symbol || is a concatenation operation. An SBF can also be constructed by concatenating multiple SBFs such that $SBF_{result} = \{SBF_a||SBF_b||\dots\}$. We denote `Local SBFs` (LSBFs) as a set of SBFs which are created by using information from a single blockchain branch. We also denote `Global SBFs` (GSBFs) as a set of SBFs which are created by concatenating all `Local SBFs`. The GSBFs are used to check the existence of a related information regardless in which blockchain the information resides, while the LSBFs are used to check information on a specific blockchain.

SBFChain. For accountability purposes of the created GSBFs, we introduce `SBFChain`. `SBFChain` is a blockchain-like data structure which maintains metadata about the GSBFs and tracks changes to the GSBFs. Each entry in `SBFChain` is numbered. An entry e_n in the `SBFChain` connects to the entry e_{n-1} by adding the hash value $h_{e_{n-1}} = H(e_{n-1})$ to the entry e_n. An entry is created on every period of time, i.e. 4 minutes to show a gradual process of creating the GSBFs. The structure of `SBFChain` is shown in Figure 6.

An entry e_n contains the following information:

- The hash value $h_{e_{n-1}}$.
- The block number n.
- A timestamp ts_n.
- The hash values of the most recent GSBFs.
  - $h_{GSBF_k} = H(GSBF_k)$.
  - $h_{GSBF_m} = H(GSBF_m)$.
- The metadata of all blockchain branches in which the information is added to the most recent SBFs.
  - `Chain_ID`.
  - Block Height.
  - Data Count.

By referring to the most recent entry e, one can determine which information has been added to the recent GSBFs. The entry e will also help reconstructing the GSBFs at any given time by referring to information stored in the nearest entry e.

6.2.3 *Joint Node.*

We coin the term `joint node` to describe a new type of node, which stores and manages GSBFs and `SBFChain`. The `joint node` will be operated under a collaboration between maintainers of multiple blockchain branches. The idea of the `joint node` originally came from `blackball` databases, where the information is collected from multiple parties [7]. At the same time, the joint node also behaves similar to hardfork oracle [17], in which information about multiple chain forks can still be managed in one place. The `joint node` collects all related information from different blockchain branches and constructs SBFs and `SBFChain`.

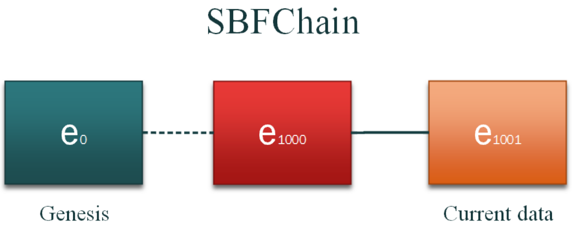

Figure 6: The structure of SBFChain.

The `SBFChain` can be used to synchronise `SBF`s maintained by different joint nodes. It is assumed that there exists a simple consensus method among the joint nodes to add new entries in the `SBFChain`, where every information update in the `SBFChain` is followed by all joint nodes as the members of the system.

The `joint` node will not cause any scalability issue to the main application of each blockchain, especially related to providing necessary storage and computing power to process requests and responses. `Joint` nodes form a new Monero subsystem which is separated from the main system consisting of normal nodes running Monero protocols. Although `joint` nodes and normal nodes are in different systems, it is assumed that there exists a mechanism such that the nodes are able to exchange information.

RPC can be used as a communication scheme between normal nodes and `joint` nodes as well as between `joint` nodes and the SPV wallets on the client side. P2P communication scheme is necessary for the `joint` nodes to update new information from the network of multiple blockchains. The relationship between joint nodes, normal nodes, and SPV wallets is shown in Figure 7.

The users' wallets can also actively seek advice from the `joint` nodes regarding the raw transactions the wallet create such that the problems of key reuse can be prevented on early stage. However, the normal nodes can utilise the `SBF`s maintained by the `joint` nodes to perform a simple checking algorithm before processing the transaction.

Although the `joint` nodes store the `GSBF`s, they are not authenticated to extend any blockchains nor modify the information that has been stored inside the blockchains. All updates on the blockchains and memory pools will be inserted into the respective `LSBF`s and `GSBF`s. We use the term service subsystem to refer a network of joint nodes.

7 DISCUSSION

In this section, security analysis and performance analysis of our proposed solution are discussed.

7.1 Security Analysis

7.1.1 Active Attack. We assume there exist dishonest `joint` nodes in the service subsystem where the majority of the node members are behaving honestly by following the protocol correctly. When the dishonest `joint` nodes receive requests from the a client (either a wallet or a normal node) to verify whether key images or key image-mixin tuples are in the current `SBF`s, the dishonest `joint` nodes will produce incorrect responses. To mitigate the problem, the client can send the requests to multiple `joint` nodes in the subsystem selected at random. Assuming that the majority of the `joint` nodes in the subsystems are behaving honestly, the client will find inconsistencies of the responses. The client then regards the results as votes to distinguish the correct responses form the incorrect ones, where the correct responses are likely become the majority as honest `joint` nodes always return correct responses.

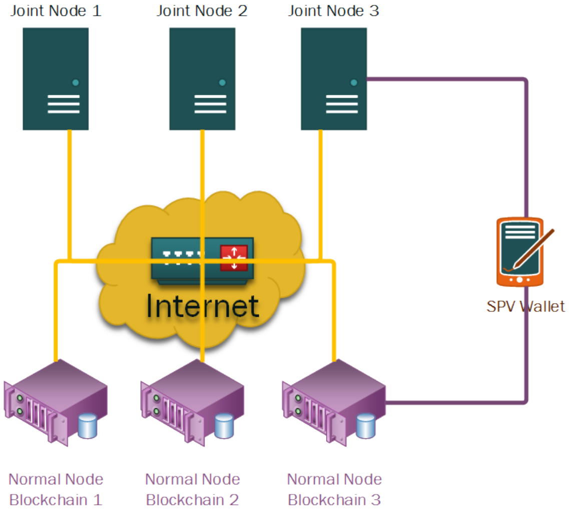

Figure 7: Joint nodes can assist SPV wallets as well as normal nodes of different blockchains.

A dishonest `joint` node can also be detected by its peers. The `joint` nodes validate each other's `SBF`s files by confirming the hash values of the `SBF`s and the hash values stored in the `SBFChain`. If the information does not match, any nodes returning incorrect information can be blacklisted. The blacklist information will be published to all clients. Random requests can also be utilised for checking mechanism to detect any dishonest `joint` nodes.

The normal nodes of different blockchain branches can also cooperate to verify the correctness of the `SBF`s maintained by the `joint` nodes. However, this requires extra computing resources by the normal nodes. The verification of the correctness of `GSBF`s can be done in a two-stage reconstruction.

1. **Stage one: intrachain reconstruction**. In this stage, the normal nodes of each blockchain branch compute `Local SBF`s (`LSBF`s) by using their own blockchain data according to an agreed entry on the `SBFChain`. The reconstruction of the `Local SBF`s can start from the `Fork Point` of that blockchain branch instead from the genesis block (block number zero). The correctness of the `Local SBF`s (`LSBF`s) depends on the honesty of the normal nodes of the blockchain. Assuming that the majority of the nodes behave honestly, then the correct `LSBF`s can always be generated.
2. **Stage two: interchain reconstruction**. The nodes of different blockchain branches cooperate to generate a set of `Global SBF`s (`GSBF`s). These `GSBF`s are created by concatenating all `LSBF`s. Assuming that all `LSBF`s are correct, then the produced `GSBF`s are also correct.

A dishonest normal node can also try to confirm transactions that have problems of key reuse into new blocks it produces by collaborating with miners that have a sufficient computing power. In this case, other normal nodes can re-validate these transactions with the help of `joint nodes`. When these transactions are proven to be malicious, then the blocks containing these malicious transactions can be ignored. Assuming that the majority of the nodes behave honestly, then there will be a temporary fork which will resolve after several blocks according to the current Monero protocol. Since the miners will suffer financial loss if the produced blocks are removed, they are less motivated to behave dishonestly.

7.1.2 Passive Attack. In the passive attack, it is assumed that the attacker has access to the public blockchains. The attacker develop analytic tools to determine traceable transactions. The success of the attack is determined by the number of traceable transactions and the portion of the traceable transactions compared to the total number of transactions in the system.

Since that the active attack can be prevented by using our proposed solution, the passive attack can also be prevented. Passive attacks analyse existing valid transactions that have been confirmed in the blocks. With no malicious transaction being added to the blocks, then the passive attack will not produce any expected outcome, assuming no extra information is given to the attacker.

7.2 Performance Analysis

7.2.1 Hard Fork Management. We conducted experiments to calculate the extra computing resource in managing the extra information for hard fork management. The experiments used a Ubuntu 18.04 LTS virtual machine equipped with 8GB RAM and maximum 2 CPU cores. A new table called `Chain_Info` was created using `LMDB` database system, which is the same database product that is being used to store and manage the blockchain data of the current version of Monero.

Two million `Chain_ID` - `Fork_Point` tuples were written to the database and then read. The processes were then repeated 10,000 times. About 1.2MB storage was required to store the two million records, while writing average time was 28.37 milliseconds and the reading average time was 28.19 milliseconds. The detailed result is shown in Figure 8. The experiment shows that the computing resource for the required operations is small such that today's regular computers can afford it.

7.2.2 Joint Node Affordability. In our proposal, the `GSBF` will be maintained by a special type of node called `joint node`. A `joint node` maintains a set of `GSBF`s which is relevant to all existing or future Monero blockchain branches.

An experiment was conducted to calculate the time and storage needed to create an `SBF`. The experiment utilises Jay Baird's Scalable Bloom Filter Python library, `pybloom`[8]. The experiment used a `LARGE_SET_GROWTH` setting to anticipate a large dataset growth. In this setting, a surge jump in storage size will happen every time the system hits its maximum capacity. The results as in Figure 9 show that the time required to create an `SBF` is linear to the number of the data inserted in the `SBF` with the average of 17.308 data per second. The file size, however, increased significantly every time the capacity is full, according to the `LARGE_SET_GROWTH` algorithm. Our experiment also showed that creating an `SBF` with 100 million data produced 372.1MB of `SBF` file within around 96 minutes. Due to the low resource requirement when creating the `SBF`, recalculating the `SBF` will not be a problem.

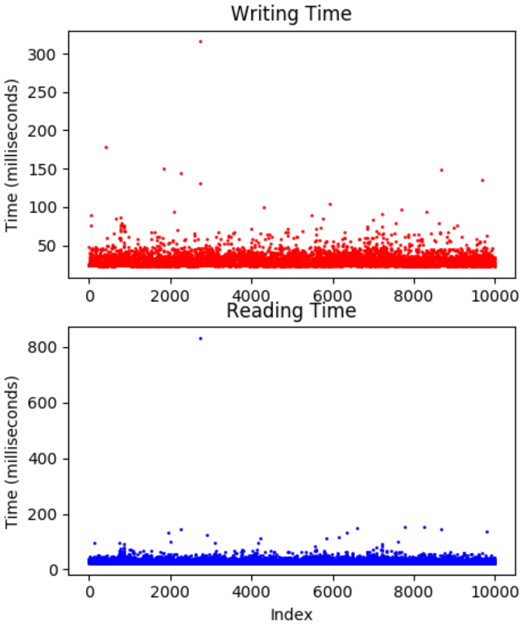

Figure 8: The read-write processing time for `Chain_Info` database using `LMDB`.

7.3 Limitation

By mitigating the problem of key reuse, our solution is able to mitigate a passive attack which utilises analyses on public blockchains. Our solution cannot prevent a passive attack on network level which is still considered as one of the biggest privacy issues in cryptocurrency [8]. Our solution is also prone to a passive attack conducted by an honest-but-curious `joint node`, where the joint node can potentially trace users' transaction given enough information. This problem, however, is not exclusive to our proposed system, but also applies to all Monero nodes.

8 CONCLUSION AND FUTURE WORK

We investigate the problem of key reuse as an unwanted impact of Monero hard forks. We build a dataset from three different blockchain branches and determine the traceable inputs as the result of the key reuse problem. We also identify the cascade effect and the reduced anonymity as the side effects of the main problem. Our analyses discover that the scalability of the problem of key reuse is correlated to the market price of the respected coins and the supports from cryptocurrency markets to the newly created cryptocurrencies. We also propose a mitigation strategy in the form

---

[8] https://github.com/jaybaird/python-bloomfilter

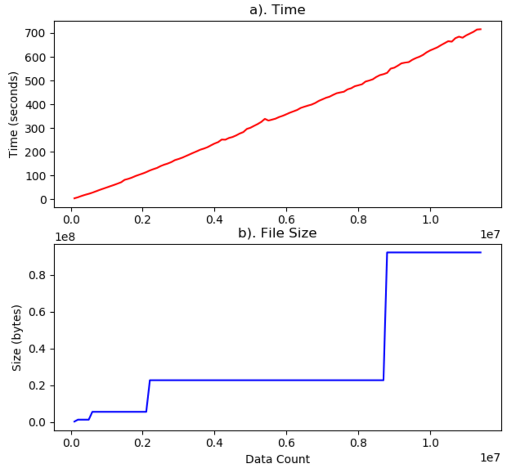

Figure 9: Part a) shows the creation time of SBF which is a positive linear to the data size. Part b) shows the result's file size where the file size will be increased when the capacity of the SBF is full.

of hard fork management and key image management, where joint nodes play an important role in the proposed strategy.

For future work, we will investigate how our solution can be implemented in different types of cryptocurrency. We will also investigate different options in handling protocol-level changes to avoid hard fork. The new method should be able to support fundamental changes in the system without creating a new blockchain branch. This type of solution will be useful to be implemented in systems with active development such as Monero. It is also interesting to further investigate the correlation between cryptocrrency market price and the number of transactions recorded in Monero blockchain to uncover the actual behavior of Monero users and how Monero is used in the real world.

ACKNOWLEDGMENT

The work of Ron Steinfeld and Joseph K. Liu was supported in part by ARC Discovery Project grant DP180102199.

REFERENCES

[1] Paulo Sérgio Almeida, Carlos Baquero, Nuno Preguiça, and David Hutchison. 2007. Scalable bloom filters. *Inform. Process. Lett.* 101, 6 (2007), 255–261.
[2] BatmanLovesCrypto. 2018. Monero Classic and Monero Original on the same blockchain? Help me understand. https://www.reddit.com/r/Monero/comments/8eovv5/monero_classic_and_monero_original_on_the_same/
[3] Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* 13, 7 (1970), 422–426.
[4] Vitalik Buterin. 2016. Simple Replay Attack Protection. https://github.com/ethereum/EIPs/blob/master/EIPS/eip-155.md
[5] Sherman S. M. Chow, Joseph K. Liu, and Duncan S. Wong. 2008. Robust Receipt-Free Election System with Ballot Secrecy and Verifiability. In *NDSS*.
[6] dEBRYUNE. 2018. PoW change and key reuse. https://ww.getmonero.org/2018/02/11/PoW-change-and-key-reuse.html
[7] Justin Ehrenhofer. 2018. Monero Blackball Site. https://monero-blackball.github.io/monero-blackball-site/
[8] Ryan Henry, Amir Herzberg, and Aniket Kate. 2018. Blockchain access privacy: challenges and directions. *IEEE Security & Privacy* 16, 4 (2018), 38–45.
[9] HitBTC. 2018. The Monero Original Fork has happened. https://blog.hitbtc.com/the-monero-original-fork-had-happened/
[10] HitBTC. 2018. Statement on MoneroV fork. https://blog.hitbtc.com/statement-on-monerov-fork/
[11] HitBTC. 2018. Statement on XMO Monero fork. https://blog.hitbtc.com/statement-on-xmo-monero-fork/
[12] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. 2017. *Non-interactive proofs of proof-of-work.* Technical Report. Cryptology ePrint Archive, Report 2017/963, 2017. Accessed: 2017-10-03.
[13] knaccc. 2017. Description of a potential privacy leak and recommendation to mitigate. https://github.com/monero-project/monero/issues/1673#issuecomment-278509986
[14] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. 2017. A traceability analysis of Monero's blockchain. In *European Symposium on Research in Computer Security.* Springer, 153–173.
[15] Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. 2014. Linkable Ring Signature with Unconditional Anonymity. *IEEE Trans. Knowl. Data Eng.* 26, 1 (2014), 157–165.
[16] Joseph K Liu, Victor K Wei, and Duncan S Wong. 2004. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy.* Springer, 325–335.
[17] Patrick McCorry, Ethan Heilman, and Andrew Miller. 2017. Atomically trading with roger: Gambling on the success of a hardfork. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology.* Springer, 334–353.
[18] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2013. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. *USENIX ;login:* (2013).
[19] Monero. [n. d.]. Monero XMR Forks & Hard Forks. https://monero.org/forks/
[20] monero hax123. 2018. Corrupt RPC responses from remote daemon nodes can lead to transaction tracing. https://hackerone.com/reports/304770
[21] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. 2018. An Empirical Analysis of Traceability in the Monero Blockchain. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 143–163.
[22] Satoshi Nakamoto. 2008. *Bitcoin: A peer-to-peer electronic cash system.* Report. http://bitcoin.org/bitcoin.pdf
[23] Shen Noether, Adam Mackenzie, et al. 2016. Ring confidential transactions. *Ledger* 1 (2016), 1–18.
[24] Surae Noether, Sarang Noether, and Adam Mackenzie. 2014. MRL-0001: A note on chain reactions in traceability in CryptoNote 2.0. *Technical report2014* (2014).
[25] propercoil. 2018. Replay protection? https://www.reddit.com/r/Monero/comments/8agjfd/replay_protection/dx0lun4/
[26] Bailey Reutzel. 2017. Logical or Not, Bitcoin's Coming Fork Is Boosting Its Price. https://www.coindesk.com/logical-not-bitcoins-coming-fork-boosting-price/
[27] Dorit Ron and Adi Shamir. 2013. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Security.* Springer, 6–24.
[28] Jeff Sauro and James R Lewis. 2016. *Quantifying the user experience: Practical statistics for user research.* Morgan Kaufmann.
[29] sgp. 2018. How can individuals safeguard themselves and the community against a key reusing fork? https://monero.stackexchange.com/a/7847
[30] Riccardo Spagni. 2018. Monero 0.13.0 "Beryllium Bullet" Release. https://www.getmonero.org/2018/10/11/monero-0.13.0-released.html
[31] Shifeng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. 2017. RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero. In *ESORICS II (LNCS)*, Vol. 10493. Springer, 456–474.
[32] user36303. 2017. Replay attack and Cryptonotes. https://monero.stackexchange.com/a/5718
[33] user36303. 2018. How can individuals safeguard themselves and the community against a key reusing fork? https://monero.stackexchange.com/a/7844
[34] Nicolas van Saberhagen. 2018. Cryptonote v 2.0, 2013. *URL: https://cryptonote.org/whitepaper.pdf. White Paper. Accessed* (2018), 04–13.
[35] Dimaz A. Wijaya, Joseph Liu, Ron Steinfeld, and Dongxi Liu. 2018. Monero Ring Attack: Recreating Zero Mixin Transaction Effect. In *TrustCom.* IEEE, 1196–1201.
[36] Dimaz Ankaa Wijaya, Joseph Liu, Ron Steinfeld, and Dongxi Liu. 2019. Risk of Asynchronous Protocol Update: Attacks to Monero Protocols. (2019). to appear.
[37] Dimaz Ankaa Wijaya, Joseph Liu, Ron Steinfeld, Dongxi Liu, and Tsz Hon Yuen. 2018. Anonymity Reduction Attacks To Monero. In *The 14th International Conference on Information Security and Cryptology.* Springer.
[38] Bin Yu, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. 2018. Platform-Independent Secure Blockchain-Based Voting System. In *ISC (LNCS)*, Vol. 11060. Springer, 369–386.
[39] Zuoxia Yu, Man Ho Au, Jiangshan Yu, Rupeng Yang, Qiuliang Xu, and Wang Fat Lau. 2019. New Empirical Traceability Analysis of CryptoNote-Style Blockchains. In *Financial Cryptography and Data Security.*
[40] Alexei Zamyatin, Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, Edgar Weippl, and WJ Knottebelt. 2018. A wild velvet fork appears! Inclusive blockchain protocol changes in practice. In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security*, Vol. 18.