# A Practical De-mixing Algorithm for Bitcoin Mixing Services

Younggee Hong
Department of Computer Science and Engineering
Korea University
Seoul, Republic of Korea
gee308@korea.ac.kr

Hyunsoo Kwon
Department of Computer Science and Engineering
Korea University
Seoul, Republic of Korea
hs_kwon@korea.ac.kr

Jihwan Lee
Department of Computer Science and Engineering
Korea University
Seoul, Republic of Korea
jhlee01s@korea.ac.kr

Junbeom Hur
Department of Computer Science and Engineering
Korea University
Seoul, Republic of Korea
jbhur@korea.ac.kr

## ABSTRACT

Bitcoin mixing services improve anonymity by breaking the connection between Bitcoin addresses. In the darkweb environment, many illegal trades, such as in drugs or child pornography, avoid their transactions being traced by exploiting mixing services. Therefore, de-mixing algorithms are needed to identify illegal financial flows and to reduce criminal activity. Unfortunately, to the best of our knowledge, few studies on analyzing mixing services and de-anonymizing transactions have been proposed.

In this paper, we conduct an in-depth analysis of real-world mixing services, and propose a de-mixing algorithm for Helix, one of the most widely used Bitcoin mixing services. The proposed algorithm de-anonymizes the relationship between the input and output addresses of mixing services by exploiting the static and dynamic parameters of mixing services. Our experiment showed that, we could identify the relationships between the input and output addresses of the Helix mixing service with a 99.14% accuracy rate.

## CCS CONCEPTS

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

## KEYWORDS

Bitcoin; Mixing service; Helix; Blockchain analysis

**ACM Reference Format:**
Younggee Hong, Hyunsoo Kwon, Jihwan Lee, and Junbeom Hur. 2018. A Practical De-mixing Algorithm for Bitcoin Mixing Services. In *BCC'18: The 2nd ACM Workshop on Blockchains, Cryptocurrencies and Contracts, June 4, 2018, Incheon, Republic of Korea.* ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/10.1145/3205230.3205234

## 1 INTRODUCTION

Bitcoin is a cryptocurrency created by Satoshi Nakamoto in 2008 [19]. Since a Bitcoin address is recommended to be used only once, a new address is generated for each transaction. Since Bitcoin addresses are randomly generated from their owners' private keys and act as pseudonyms for their owners, identifying specific users by the addresses themselves is difficult.

However, Bitcoin cannot completely guarantee user anonymity for the following reasons. First, since the blockchain is public, anyone can see the transaction flow transparently [1, 2], which in turn can increase probability of the deanonymization attack by clustering related addresses into a single wallet [3, 13, 14, 17, 20, 21]. Second, most exchanges have a Know Your Customer (KYC) policy. That is, before purchasing Bitcoins, customers must go through the process of authentication, such as by account or mobile phone. Finally, it is possible to deanonymize users using information publicly known in the blockchain networks [5, 6, 16]. For example, Biryukov et al.[5] proposed a deanonymisation method using IP address that generated the transaction. Koshy et al. [16] has mapped Bitcoin addresses to IPs using transactions collected over five months. Biryukov et al. [6] showed that even with Tor, it is possible to fingerprint users and deanonymize them to some extent.

In order to overcome such limitations and improve anonymity in practice, the Bitcoin ecosystem has adopted mixing services. To achieve this, a third-party called a mixer mixes multiple Bitcoin transactions in such a way that the relationships between the transactions are hidden from the outsider's point of view. In addition, most mixing services are accessible only through Tor[1] in order to break the connection between IP address and Bitcoin address. As a result, they enhance anonymity by disconnecting input and output addresses.

Unfortunately, several studies show that mixing services are being frequently abused for criminal activities such as the trading of illegal goods [4, 22]. For example, `silk road` [9] traded various drugs, malicious code, hacking technologies, credit card information, and stolen accounts. In addition, recent ransomware attackers have enforced victims to pay them in Bitcoin by a certain deadline to restore encrypted files [24]. Such cases, also include the use of a mixing service to avoid tracking by investigative agencies.

[1]https://www.torproject.org/

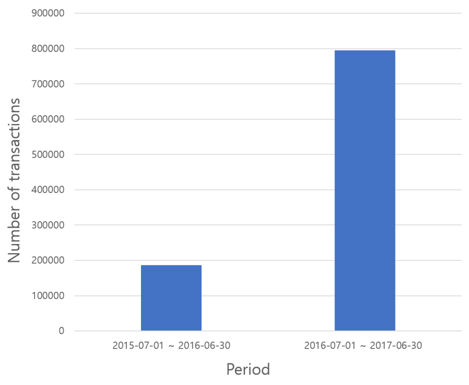

**Figure 1: Transaction increment**

We looked into the transaction volumes of mixing services from 2015-07-01 to 2017-06-30 to figure out how many people are using them. As shown in Figure 1, the transaction volume from mid 2016 to mid 2017 increased 4.27 times from the previous year[2], revealing the rapid growth in mixing service usage. Even though several previous studies [11, 18] have been proposed analyzing mixing services, their results are specific to a few specific mixing services, which are now closed. Therefore, the design of a more generic and efficient de-mixing algorithm is still an open and challenging problem.

In this paper, we propose a de-mixing algorithm which connects the input and output transactions of a mixing service. The main contributions of our study are as follows:

- We measured the approximate anonymity levels based on usage of the mixing service.
- We investigated mixing service parameters and found traceable features for several real-world mixing services. Exploiting those features, we then proposed a generic de-mixing algorithm.
- On the basis of our experiment, we could identify the relationships between input and output addresses of the Helix mixing service, one of the most widely used mixing services in practice, with a 99.14% accuracy rate.

Although to show its effectiveness the proposed algorithm measures performance through the Helix mixing service, it is important to note that our algorithm is not limited to Helix, but can be applied to the other mixing services.

## 2 RELATED WORK

Moser et al. [18], analyzed three mixing services (*Bitcoin Fog*[12], *Blockchain.info*[1], *BitLaundry*) using the taint analysis function of *Blockchain.info*. When it came to *Bitcoin Fog* and *Blockchain.info*, they could not find any meaningful relationship between mixed transactions. However, for *BitLaundry*, they could find it successfully because there are just a few transactions using the mixing services. However, some limitations apply when using this method to analyze mixing services these days. First, the taint analysis function of *Blockchain.info* is currently removed. Second, transaction volumes have significantly increased so such manual analyses of a few mixing transactions has become infeasible. Third, *BitLaundry* disappeared from the web and *Blockchain.info* has stopped its mixing service. Therefore, such an analysis method specific to certain mixing services cannot be used as a generic tool for analyzing mixing services.

Balthasar et al. [11], analyzed three mixing services (*DarkLaunder*, *Helix* [15], *Alphabay*) using Chainalysis [8], which is a commercial Bitcoin transaction analysis and clustering tool. They found that *DarkLaunder* was vulnerable because it collects almost all input addresses into a single central address, making it easy to find the path from there to the customers' wallet addresses. They also claimed that *Helix* and *Alphabay* cannot guarantee complete anonymity through manual analyses. However, this analysis has similar limitations to Moser et al.'s study [18]. The mixing service environment changes rapidly—the mixing algorithm can change at any time and mixing services may become inaccessible. *DarkLaunder*, for instance, cannot be accessed these days. In addition, *Alphabay* was terminated by the US Government [23]. Furthemore, their de-mixing algorithm is not generic, and thus limited to specific mixing services.

Hence, designing a generic mixing service analysis algorithm remains an open and challenging problem. In this paper, we conduct an in-depth analysis of mixing services and propose a generic de-mixing algorithm. Our algorithm works by exploiting only the input and output transactions of mixing services, so can be applied to all mixing services.

[2]We used commercial tools to investigate the volume for Helix.

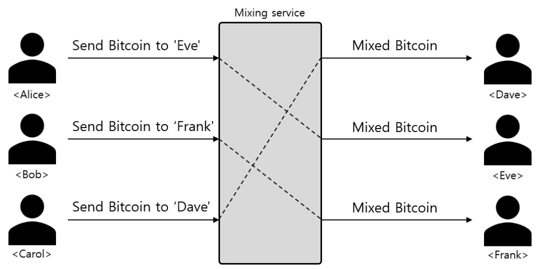

**Figure 2: Mixing service**

## 3 MIXING SERVICE ANALYSIS

### 3.1 Mixing service

Bitcoin mixing services improve anonymity by breaking the connections between addresses. If there are multiple input-output transaction pairs, the mixer mixes them in such a way that associating input and output transactions from the outsider's point of view is impossible, as shown in Figure 2. Thus, the anonymity level can be determined from the number of transactions involved.

Since the anonymity level is proportional to the number of inputs, identifying and obtaining the transactions involved in the mixing service is the first step of the analysis. Thus, we gathered transaction data involved in the mixing service after Jan. 2017 using the Chanalysis tool. On the basis of our investigation, we found that most mixing services set a minimum deposit value for mixing. However, while collecting the inputs, we found that there were

**Table 1: Mixing service usage**

| Mixing service | Number of inputs | Period |
|---|---|---|
| *BitcoinBlender* | 2285 | 2017-01-01 ~ 2017-09-01 |
| *CryptoMixer* | 459 | 2017-01-01 ~ 2017-10-16 |
| *Bitcoin Fog* | 26798 | 2017-01-01 ~ 2017-10-26 |
| *BitMixer* | 41826 | 2017-01-01 ~ 2017-07-24 |
| *Helix* | 154890 | 2017-01-01 ~ 2017-10-26 |

Bitcoin values lower than the minimum deposit value. We classified these as abnormal values and removed them from the set of normal data used for analysis. Table 1 shows the transaction information we gathered for each different mixing service, that is, *BitcoinBlender* [7], *CryptoMixer* [10], *Bitcoin Fog* [12], BitMixer[3], and *Helix* [15].

The number of input transactions in the cases of *BitcoinBlender* and *CryptoMixer* is relatively small. Thus, de-mixing transactions for the services would be much easier even with a manual analysis. On the other hand, *Bitcoin Fog*, *BitMixer* and *Helix* have a large number of inputs. Fortunately, we can reduce the anonymity level by using other information such as time and Bitcoin (BTC) transaction values. For example, in Figure 2, if the input value of Alice and output value of Eve are both 2 BTC, and the rest values are all 1 BTC, we can connect Alice and Eve with high probability. However, mixing services utilize some other parameters to make transactions difficult to track. The next section shows the mixing parameters commonly used in practical mixing services.

## 3.2 Mixing service parameter analysis

We analyzed the capabilities and parameters provided by most of the currently available mixing services.

- **Mixing service fee**. Mixing services make a profit by receiving a mixing service fee. Most mixing services receive various fees rather than a fixed one. The amount of the mixing fee is set by the user or randomly determined by the mixing service. The randomness of the mixing fee is used to make it difficult to associate the input and output transactions.
- **Delay**. If the mixing service generates output transactions as soon as it mixes the input transactions, time-based attacks are easily possible. Therefore, mixing services avoid this by setting delays to some extent. As the range of delay widens, associating input and output transactions after mixing becomes harder. The delay is set by the user or randomly determined by the mixing service.
- **Max_output_address**. The mixing service can split an input into multiple outputs. Most mixing services can also set a different delay time for each output. Thus, splitting transactions with different delays makes the services much less vulnerable to de-mixing. Therefore, such combinations of different mixing parameters should be considered when analyzing mixing services.
- **Transaction Fee**. Bitcoin must impose a transaction fee on the miner each time a transaction is created. Therefore, if the customer divides the Bitcoin input into several pieces, the transaction fee increases and most mixing services charge a fee for each address.

[3]The BitMixer service is currently stopped.

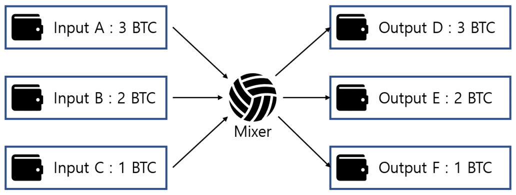

**Figure 3: Example transactions for a mixing-service**

Analyzing mixing services would be complex if there parameters were used in a diverse combinatorial way. However, after investigating real-world mixing services, we found there is a gap between the theoretically achievable and practically implemented anonymity. As shown in Table 2[4]. For *BitCloak*, the Max_out_address is 1. This means that we can easily calculate the number of matching cases by simply comparing inputs to each individual output. For *Bitcoin Mixer*, *Helix* and *Helix light*, the mixing service fee is fixed. If the fee is fixed, the output value can easily be calculated and used to find possible connections to given input addresses. In the next section, we propose a de-mixing algorithm by exploiting these observations for real-world mixing services.

## 3.3 De-mixing algorithm

As a simple example, we suppose there are three input transactions A, B and C, and three output transactions D, E and F for a mixing service as shown in Figure 3. We also assume that there is no mixing service fee in this example for simple analysis. In order to find relationships among the input and output transactions, the algorithm starts with the first input transaction. To find possible output transactions corresponding to the input transaction A, the following two cases may be considered as candidate input transactions: (1) transaction D, or (2) transactions E + F, since their Bitcoin values are the same as that of A, that is, 3. Since the related output has not yet been determined, the algorithm temporarily stores the state and checks the next input. For the next transaction B, there is only a single matching output transaction E. Thus, the algorithm removes the corresponding output E, and restarts from the previously stored state. In this case, D is determined as the output associated with A since it is now the only output candidate for A. Then, the last input transaction C, can similarly be easily associated with the output transaction F.

Thus, the proposed de-mixing algorithm progresses recursively to find relationships among the transactions involved in mixing services. Although the previous example shows a simple case for easy understanding, the practical algorithm should consider more complex scenarios with diverse mixing parameters, such as mixing delays, mixing service fees, and a number of output addresses.

[4]As of January 3, 2018.

**Table 2: Mixing service list**

| Mixing service | Url | Minimal deposit | Mixing service fee | Delay | Max_output_address |
|---|---|---|---|---|---|
| *CoinMixer* | coinmixibh45abn7.onion/ | 0.001 | 1 - 3% | ~120h | 5 |
| *BitcoinBlender* | bitblendervrfkzr.onion/ | 0.01 | 1 - 3% | ~99h | 10 |
| *CryptoMixer* | cryptomixns23scr.onion/ | 0.001 | 0.5 - 3% | ~48h | 10 |
| *BitMix* | bitmixbizymuphkc.onion/ | 0.01 | 0.4 - 4% | ~24h | 5 |
| *PrivCoin* | tr5ods7ncr6eznny.onion/ | 0.01 | 0.8 - 3.8% | ~24h | 10 |
| *Bitcoin Fog* | foggeddriztrcar2.onion/ | 0.005 | 1 - 3% | ~48h | 20 |
| *BitCloak* | bitcloak43blmhmn.onion/ | 0.015 | 1 - 3% | ~8h | 1 |
| *Bitcoin Mixer* | Bitcoin-mixer.org/ | 0.01 | 1.5% | ~24h | 10 |
| *Helix* | grams7eu3phkfrt3.onion/helix/ | 0.02 | 2.5% | ~24h | 5 |
| *Helix light* | grams7eu3phkfrt3.onion/helix/light/ | 0.02 | 2.5% | ~6h | 5 |

**Algorithm** De-mixing

```
For $i$ = 1 to $N$, repeat the following
1: $OUTPUT'$ ← Extract the output set satisfying the following conditions.
   (1) The time range is between $Input_i.time$ and $Input_i.time + Max\_delay$.
   (2) The Bitcoin value is less than $Input_i.value$.
2: $OUTPUT_{subset}$ ← Stores all combinations in $OUTPUT'$ that can be made up to $Max\_out$ or less.
3: Compare the Bitcoin values of the $input_i * (1 - fee)$ and each $OUTPUT_{subset}$. If the values are the same, add to $Result$ set.
4: If the size of the $Result$ set is 1 (determines output).
   (a) Remove $input_i$ from set INPUT.
   (b) Remove corresponding $OUTPUT_{subset}$ from set OUTPUT.
   (c) The algorithm is repeated from the first input.
5: If the size of the $Result$ set is not 1 for all inputs (cannot determine output).
   (a) Print the number of cases for each input.
```

The above algorithm 'De-mixing' describes the pseudo-code for the proposed de-mixing procedure. In the algorithm, $N$ denotes the number of inputs and $M$ denotes the number of outputs. $Max\_out$ denotes the maximum number of output addresses, where the range of $M$ is ($N \leq M \leq N * Max\_out$). Where $Max\_delay$ denotes the maximum delay provided by the mixing service. INPUT = $\{Input_1, Input_2, \ldots, Input_N\}$ and OUTPUT = $\{Output_1, Output_2, \ldots, Output_M\}$ denote the sets of input transactions and output transactions, respectively. $Input_i.time$ denotes the time when an $input_i$ transaction occurred. $Input_i.value$ represents the Bitcoin value that the $input_i$ transaction has.

In the first step, for every output, if the time at which the output transaction occurred is between $Input_i.time$ and $Input_i.time + Max\_delay$, the algorithm stores it in $OUTPUT'$. In the second step, the algorithm calculates all possible output combinations from $OUTPUT'$ and stores them to $OUTPUT_{subset}$. When $M'$ is the size of $OUTPUT'$, the size of $OUTPUT_{subset}$ is:

$$\sum_{j=1}^{\texttt{Max\_out}} \binom{M'}{j}.$$

In the third step, if the sum of each $OUTPUT_{subset}$ and $input_i * (1 - fee)$ are the same, the algorithm adds it to the $Result$ set. If the size of the $Result$ set is 1, the algorithm determines that the associated output has been found correctly. The matched output is then removed from OUTPUT so that it is not included in the next procedures. Then the algorithm repeats from the previously stored state again in a recursive way. If the size of the $Result$ set is not 1 for all inputs, the algorithm cannot determine the unique output transactions associated with a given input transaction. Nevertheless, according to our experiment, we found that the number of possible outputs could be significantly reduced at this moment. It implies the algorithm can determine a reduced candidate set of outputs with higher probability than that of selecting candidate outputs from uniformly distributed set. Therefore, in this case, the algorithm prints the $Result$ set for each input.

In terms of time complexity, the worst case occurs when no matching is found. In this case, step 2 in the algorithm needs to calculate all possible combinations of $OUTPUT'$ because the size of the OUTPUT has not been reduced. For one input, the combination operation should be performed $|OUTPUT_{subset}|$ times, so the total time complexity is $O(n * \sum_{j=1}^{\texttt{Max\_out}} \binom{M'}{j})$. On the other hand, the best case occurs when each matched output address is found immediately. At this time, the size of $M'$ converges to 1.

## 4 EVALUATION

### 4.1 Test data generation

We implement[5] the proposed algorithm to analyze Helix, which is one of the most widely used mixing services in practice—as shown in Table 1. We have collected a total of 700 Helix input transactions that occurred in June 2017 using Chainalysis. We then generated the corresponding Helix outputs for the test data set by randomly setting Helix mixing parameters, that is the mixing service fee, the mixing delay, and the number of output addresses in accordance with Table 2.

### 4.2 Experimental result

We applied the proposed de-mixing algorithm to the Helix mixing service using the test data. We implemented it in C++ on a PC equipped with 3.40 GHz CPU, 16 GB RAM and Windows 10 OS.

[5]https://github.com/HongYoungGee/De-mixing

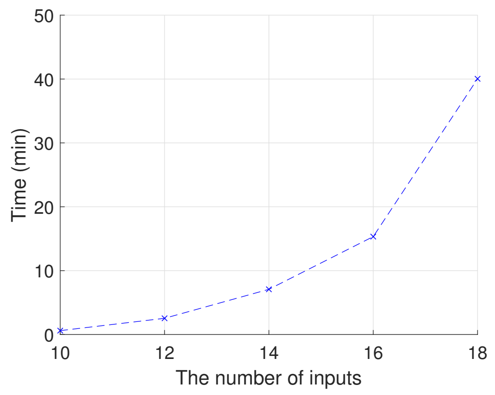

**Figure 4: Computation cost**

Figure 4 shows the computation cost with different number of input transactions. We repeated the algorithm 10 times for each set of input transactions, and calculated the average processing time for the inputs.

In terms of accuracy, 694 out of the total 700 inputs were successfully associated with unique output transactions. Thus, we could identify the relationships between Helix input and output addresses with a 99.14% accuracy rate. For the remaining 6 input transactions whose corresponding outputs could not be determined, we analyzed what circumstances could cause such mismatches, as discussed in the next section.

## 4.3 Mismatch analysis

In this section, we analyze why an 100% accuracy rate is not achieved. We defined it as a mismatch when an input transaction cannot be associated with a unique set of output transaction. We consider two cases of mismatch.

In the first case, a mismatch could occur when having the same input value. For example, in Table 3, $t_{A_2}$, $t_{A_3}$ and $t_{A_4}$ are the outputs of input $t_{A_1}$; $t_{B_2}$ and $t_{B_3}$ are the outputs of input $t_{B_1}$. Since $t_{A_1}$ and $t_{B_1}$ have the same input value, the sum of the output value would be the same even if the number of output addresses is different because the mixing service fee is fixed in Helix as follows.

$$\begin{aligned} t_{A_1} &= t_{A_2} + t_{A_3} + t_{A_4}, \\ &= t_{B_2} + t_{B_3}, \\ t_{B_1} &= t_{A_2} + t_{A_3} + t_{A_4}, \\ &= t_{B_2} + t_{B_3}. \end{aligned}$$

In the second case, a mismatch could occur even if the input values are not the same. As shown in Table 4, $t_{C_2}$, $t_{C_3}$ and $t_{C_4}$ are the outputs of input $t_{C_1}$; $t_{D_2}$, $t_{D_3}$, $t_{D_4}$ and $t_{D_5}$ are the outputs of input $t_{D_1}$. Since it holds that

$$t_{C_2} + t_{C_4} = t_{D_2} + t_{D_3} + t_{D_4} + t_{D_5},$$

**Table 3: Mismatch case 1**

| | Type | Time | Value |
|---|---|---|---|
| $t_{A_1}$ | In | 2017-6-4 9:42 | 0.1 |
| $t_{A_2}$ | Out | 2017-6-4 14:20 | 0.05703751 |
| $t_{A_3}$ | Out | 2017-6-5 2:36 | 0.024375 |
| $t_{A_4}$ | Out | 2017-6-5 4:14 | 0.01608749 |
| $t_{B_1}$ | In | 2017-6-4 9:42 | 0.1 |
| $t_{B_2}$ | Out | 2017-6-4 23:43 | 0.07605 |
| $t_{B_3}$ | Out | 2017-6-5 0:40 | 0.02145 |

**Table 4: Mismatch case 2**

| | Type | Time | Value |
|---|---|---|---|
| $t_{C_1}$ | In | 2017-6-4 12:5 | 0.1 |
| $t_{C_2}$ | Out | 2017-6-4 14:38 | 0.03217499 |
| $t_{C_3}$ | Out | 2017-6-4 19:14 | 0.024375 |
| $t_{C_4}$ | Out | 2017-6-5 6:36 | 0.04095001 |
| $t_{D_1}$ | In | 2017-6-4 12:5 | 0.075 |
| $t_{D_2}$ | Out | 2017-6-4 14:16 | 0.03598892 |
| $t_{D_3}$ | Out | 2017-6-4 15:59 | 0.00501637 |
| $t_{D_4}$ | Out | 2017-6-4 22:58 | 0.00146249 |
| $t_{D_5}$ | Out | 2017-6-4 23:25 | 0.03065722 |

the following two cases could be possible candidates

$$\begin{aligned} t_{C_1} &= t_{C_2} + t_{C_3} + t_{C_4}, \\ &= t_{D_2} + t_{D_3} + t_{C_3} + t_{D_4} + t_{D_5}, \\ t_{D_1} &= t_{C_2} + t_{C_4}, \\ &= t_{D_2} + t_{D_3} + t_{D_4} + t_{D_5}. \end{aligned}$$

By summarizing the above cases, we can conclude that given output sets $T_X$ and $T_Y$, if the sum of the subset $T_{X_i}$ that can be created from $T_X$ and the sum of the subset $T_{Y_j}$ that can be created from $T_Y$ are the same, a mismatch could occur. However, as we analyzed in Section 4.2, such cases occur rarely in practice and we believe this is because, in a Bitcoin environment, the value of a unit of BTC is very large (1 BTC = 17,190 USD[6]).

## 5 CONCLUSION AND FUTURE WORK

In this study, we investigated several real-world mixing services, and their mixing policies. As a result, we found some traceable features which can be practically exploited to analyze the mixing service. On the basis of these observations, we designed a de-mixing algorithm to find input-output relationships among the mixed transactions. The proposed algorithm could identify the relationship between the mixed input and output addresses to 99.14% accuracy. We also discussed the cases where any mismatch could occur. To the best of our knowledge, this is the first generic de-mixing algorithm proposed so far.

[6]As of January 6, 2018.

However, the proposed scheme has limitations in terms of computation overhead. Specifically, it takes a long time to perform combination operations due to the recursive procedures in the algorithm. If we can effectively reduce redundant operations, the time complexity would be reduced significantly. Thus, as future work, we aim to improve the performance of the de-mixing algorithm.

## ACKNOWLEDGMENTS

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00395, Information management system for dark web scanning). This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No.2016R1A2A2A05005402).

## REFERENCES

[1] *Blockchain.info*. https://blockchain.info/. [Online; accessed 9-January-2018].
[2] *Walletexplorer*. https://www.walletexplorer.com/. [Online; accessed 9-January-2018].
[3] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 2013. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*. Springer, 34–51.
[4] V Bhaskar, Robin Linacre, and Stephen Machin. 2017. The economic functioning of online drugs markets. *Journal of Economic Behavior & Organization* (2017).
[5] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. 2014. Deanonymisation of clients in Bitcoin P2P network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 15–29.
[6] Alex Biryukov and Ivan Pustogarov. 2015. Bitcoin over Tor isn't a good idea. In *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 122–134.
[7] BitcoinBlender. . Onion URL: bitblendervrfkzr.onion/. [Online; accessed 2-January-2018].
[8] Chainalysis. . https://www.chainalysis.com/. [Online; accessed 9-January-2018].
[9] Nicolas Christin. 2013. Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 213–224.
[10] CryptoMixer. . Onion URL: cryptomixns23scr.onion/. [Online; accessed 2-January-2018].
[11] Thibault de Balthasar and Julio Hernandez-Castro. 2017. An Analysis of Bitcoin Laundry Services. In *Nordic Conference on Secure IT Systems*. Springer, 297–312.
[12] Bitcoin Fog. . Onion URL: foggeddriztrcar2.onion. [Online; accessed 15-January-2018].
[13] Diksha Gupta, Jared Saia, and Maxwell Young. 2017. Proof of Work Without All the Work. *arXiv preprint arXiv:1708.01285* (2017).
[14] Martin Harrigan and Christoph Fretter. 2016. The Unreasonable Effectiveness of Address Clustering. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*. IEEE, 368–373.
[15] Helix. . Onion URL: grams7eu3phkfrt3.onion/helix. [Online; accessed 15-January-2018].
[16] Philip Koshy, Diana Koshy, and Patrick McDaniel. 2014. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*. Springer, 469–485.
[17] 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. In *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 127–140.
[18] Malte Moser, Rainer Bohme, and Dominic Breuker. 2013. An inquiry into money laundering tools in the Bitcoin ecosystem. In *eCrime Researchers Summit (eCRS), 2013*. IEEE, 1–14.
[19] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).
[20] Jonas David Nick. 2015. *Data-Driven De-Anonymization in Bitcoin*. Ph.D. Dissertation.
[21] Fergal Reid and Martin Harrigan. 2013. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*. Springer, 197–223.
[22] Kyle Soska and Nicolas Christin. 2015. Measuring the Longitudinal Evolution of the Online Anonymous Marketplace Ecosystem.. In *USENIX Security Symposium*. 33–48.
[23] Telegraph. *AlphaBay: World's largest dark web site is shut down*. http://www.telegraph.co.uk/technology/2017/07/20/alphabay-us-government-shuts-worlds-largest-dark-web-market/. [Online; accessed 15-January-2018].
[24] Wikipedia. *Wannacry ransomware attack*. https://en.wikipedia.org/wiki/WannaCry_ransomware_attack. [Online; accessed 9-January-2018].