CoinMingle: A Decentralized Coin Mixing Scheme with a Mutual Recognition Delegation Strategy

1st Mixue Xu
State Key Laboratory of Mathematical Engineering and Advanced Computing
Zhengzhou, China
mixue_xu@163.com

2nd Chao Yuan
State Key Laboratory of Mathematical Engineering and Advanced Computing
Zhengzhou, China
yc_xxgcdx@163.com

3rd Xueming Si
Shanghai Key Laboratory of Data Science
Shanghai, China
sxm@fudan.edu.cn

4th Gang Yu
State Key Laboratory of Mathematical Engineering and Advanced Computing
City, Country
gyu1010@126.com

5th Jinhua Fu
State Key Laboratory of Mathematical Engineering and Advanced Computing
Zhengzhou, China
304518289@qq.com

6th Feng Gao
Shanghai Key Laboratory of Data Science
Shanghai, China
gaofeng@163.com

***Abstract*—Bitcoin is the first decentralized cryptocurrency proposed by Satoshi Nakamoto. Although transactions are conducted between pseudonyms, Bitcoin cannot offer strong privacy guarantees. Mixing coins, to some extent, can address this drawback through different technologies, but also meets flexibility, storage and anonymity problems. This paper proposes a decentralized coin mixing scheme CoinMingle, based on ring signature, one-time output address, CoinJoin and our mutual recognition delegation strategy. Our mutual recognition delegation strategy is simple but effective: when a user prepares to submit messages, he will delegate it to other users for anonymity. Users in CoinMingle don't need pre-compute. And CoinMingle can tolerate different input amounts and plug the leak of personal information in shuffling phase. In addition, CoinMingle owns parallel structure which is more efficient than other coin mixing schemes.**

***Index Terms*—coin mixing, one-time output address, CoinJoin, mutual recognition delegation**

I. Introduction

With the development of P2P technology, more and more decentralised systems are in use. Sometimes the participants need self-determination, which means they can get access to the system freely. So they prefer to generate public-private key pair by themselves to unlink the online behavior with true identity. In accordance with the demands, the implementation on P2P network must attach importance to anonymity.

A. Development of Cryptocurrency

Bitcoin [1] is a successful implementation of P2P technology. People are attracted to the anonymity provided, however, actually pseudo-anonymity. An increasing body of research shows that anyone in P2P network can de-anonymize Bitcoin by using information in the blockchain [2]–[5]. Recently, more creative schemes were proposed such as ZCash [6], Dash [7], [8],CoinShuffle [9], CoinParty [10], Monero [11] and other schemes [12] to implement a true anonymous cryptocurrency.

Zcash uses zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs) [14] to guarantee anonymity. However, the proving key pk and verification key vk used in the process of zero-knowledge proof are generated by a trusted third party. Although the paper [15] has given a solution by hardcoded, everybody in the scheme can't check them for its "high threshold". At the same time, every transaction in system will cost the sender a few minutes producing a proof, so that the transaction efficiency is greatly affected.

Dash and other CoinJoin proposals, such as Mixcoin [16], SharedCoins [17], CoinShuffle and JoinMarket [18] meet the same dilemma: the attacker may disturb the transaction by inserting invalid output before users take turns signing a CoinJoin transaction. As for anonymity, although the CoinParty announced that not even the mixing peers can link input addresses to the corresponding output addresses, its shuffle phase is even weaker than CoinShuffle for everyone should submit their output addressed in the first step of shuffling. It means among the technologies the consistency of transaction message can't be guaranteed or the participant can know all the exchange graph. And the level of anonymity offered by CoinJoin can also be diminished by "CoinJoin Sodoku" [19] if the protocol is implemented incorrectly.

Monero based on CryptoNote v 2.0 mainly uses the technology of linkable ring signature [20] and one-time address to satisfy the two properties untraceability and unlinkability, by which users can trade securely. Unfortunately, Monero suffers from storage problems. Every linkable ring signature script size is linearity correlation to the cardinal of ring. Once the user needs high anonymity, the storage cost is rather expensive. A straightforward way to improve the system is constructing a constant-size linkable ring signature which can decrease script size to a constant. However, a constant-size ring signature requires a trusted party to initiate its accumulator which is contrary to decentralization. And one more question: "Temporal Analysis" [21], shows that predicting the right output in a ring signature is easier than previously thought.

978-1-5386-4870-4/18/$31.00 ©2018 IEEE

B. Our Contributions

We are totally committed to a practical decentralized anonymous cryptocurrency scheme. We note that, in the former schemes, the advantages of different privacy-protecting technologies are not fully utilized. Firstly, our CoinMingle based on ring signature one-time output address, CoinJoin and mutual recognition delegation strategy (MRDS) can cut the storage costs, improve the anonymity and support users' self-organized transaction. Then, "CoinJoin Sudoku" and "Temporal Analysis" in Monero will not go into effect in CoinMingle.

C. Paper Organization

The remainder of this paper is organized as follows. We present background information on one-time address and CoinShuffle in Section 2. Section 3 formalizes the problem and requirements of coin mixing scheme. Section 4 describes MRDS. Then, we outline our CoinMingle scheme in Section 5. Section 6 give a comprehensive analysis of mixing correctness, anonymity, flexibility, performance and further a comparison with related works. We summarize our contributions in Section 7.

II. Preliminaries

A. Ring signature

Ring signatures were first suggested by Rivest et al, who introduced the RST scheme in 2001 [22]. Ring signatures offer honestly participating users unconditional anonymity without any group manager of trusted third center. They simply require users to be part of an existing public key infrastructure [23]. A user who signed message m with a group of public key gpk and his own private key sk and outputted ring signature σ, the other people can only verify if the signature σ belongs to some participant in the group but do not know who is he. In the following paper we denote ring signature by $\sigma = R\_sign(m, sk, gpk)$ and verification by $b = Verify(\sigma, m, gpk)$. b is equal to 1 or 0 meaning the verification is successful or not.

B. One-time Output Address Using Model

In Monero, if Alice wants to transact with Bob, who has published his two standard ECDSA public keys (A, B) , then Alice will produce a new one-time output address for Bob. Following is the process of address generation and validation:

1) Alice generates a random $r \in [1, l-1]$ and computes a one-time output address $add_B = H(rA)G + B$, with hash function and G the generator of the elliptic curve.
2) Alice uses add_B as a destination address for the output and also packs $R = rG$ (as a part of the Diffie-Hellman exchange) somewhere into the transaction.
3) Bob checks the transaction with his private key (a, b), and computes $add_B{}' = H(aR)G + B$. So it is clear that $add_B{}'$ is equal to add_B.

One-time output address using model in Monero can be seen in Fig. 1.

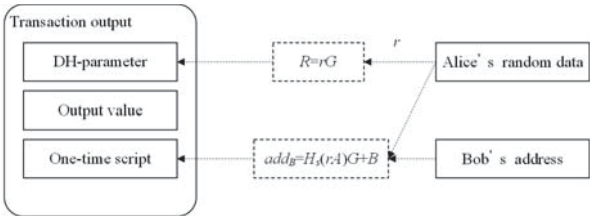

Fig. 1. One-time Output Address Using Model

Note that Alice can certainly generate a public address of a corresponding private key, but only the one who owns the private key can pay this output amount. If another user calculates the public key by this DH parameter, the result will be different from the specific one. So the owner of the output address is unique.

C. CoinShuffle

Users in CoinShuffle implement the untraceable property by shuffling phases. Suppose that there are m participants $\{M_i | i = 0, 1, \ldots, m\}$.

1) Each mixing peer M_i encrypts his output address O_i using the public keys $K_{i+1}, \ldots, K_m$ of the mixing peers in a layered encryption $[O_i]_{K_{i+1}:K_m} := E_{K_{i+1}}(E_{K_2}(\ldots E_{K_m}(O_i)))$.
2) If $i = 1$, the first peer node will send her encrypted output address $[O_1]_{K_2:K_m}$ to the peer M_2. Otherwise, M_i removes the outermost decryption E_{K_i} of the elements in received set S^{i-1}, then sends $S^i = \pi_j \circ \{D_{K_i}(S^{i-1}) \cup [O_i]_{K_{i+1}:K_m}\}$, with a private permutation π_j, to the next mixing peer M_{i+1}.
3) M_m removes the last layer of encryption E_{K_m}. and sorts the output addresses using permutation π_m and broadcasts the resulting S^m.

We can outline the process in Fig. 2.

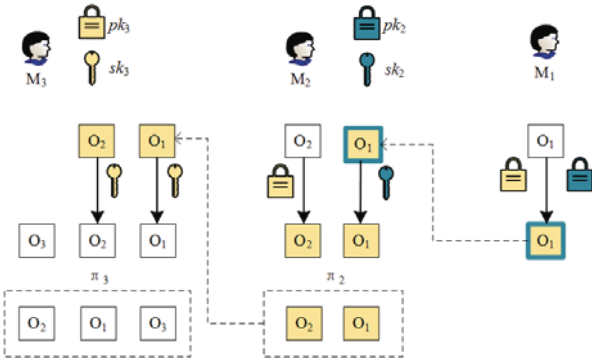

Fig. 2. Shuffling phases

Note that there is a weakness that can't be addressed: each participant produces a signature after shuffling which allows imitative behaviour of external attackers.

III. Problem Statement

Motivated by recent works on mixing coins [6]–[11], we consider the problems of how to give users the utmost assurance of their privacy and how to make it as convenient and safe as possible for the participants. Formally, m input peers which each have flexible amounts of coins available at input addresses $I_1, \ldots, I_m$ want to mix them into a set of output addresses $O_1, \ldots, O_n$ (A brilliant mechanism may tolerate various kinds of transaction). Such a mixing service should meet the following requirements:

- **Correctness:** Honest input peers can accomplish mixing operation in a timely manner. Malicious adversary can't break the mixing without any cost.
- **Anonymity:** The mixing must be anonymous. Malicious adversary can not link the input address with corresponding output address.
- **Flexibility** The mixing must be flexible. Any kind of transaction can be mixed with close attention.
- **Consistency:** The mixing must be consistent. The participants can sign on the same mixing result, even there are some external attackers.

Clearly, the centralized mixing services like Bitcoin Fog, BitLaundry and CoinSplitter do not provide correctness for the risk of steals funds. Then, works on decentralized mixes provide anonymity but not consistency in the presence of an external attacker corrupting the mixing service. Further, the inner participants can be easy to link input identification with corresponding output address of others.

IV. MRDS

Our mutual recognition delegation strategy has four steps *Gen*, *Left_share*,*Right_share* and *Verify*. Formally they are described as follows:

- **Gen.** In this phase, users publish their public key and get a *m*-vectors gpk ($m > 2$). Suppose the key pair (pk_l, sk_l) belongs to the message submitter while (pk_r, sk_r) belongs to a specific participant who is assigned to verify (it would be okay if $l = r$).
- **Left_share.** σ to be submitted should include the encrypted message mes and ring signature $R\_sign(mes, sk_l, gpk)$ using the key pk_r.
- **Right_share.** The designed confirmer using his public key pk_r decrypting σ announced in the previous phase to seek out the $left\_share$ message. Once it was found, the confirmer using the verifying key gpk to a further verification. If he runs verify successfully, the confirmer will publish σ' consisting of the message and his signature on it.

The users can run *Left_share* phase in the same time, and *Right_share* phase as well, so the beauty is time saving.

- **Verify.** The verifier is the submitter in the second phase who verifies whether his message is in σ'. He can make sure whether his submitted message is verified by the assigned participant. If succeed, the verifier will sign on the outputs of *Right_share* phase and some additional messages.

The detailed is shown in Fig. 3.

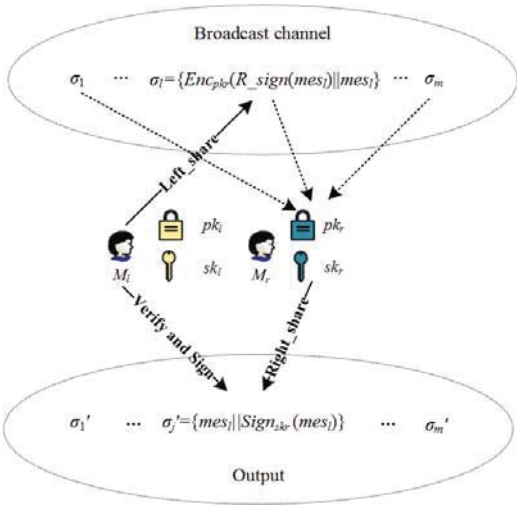

Fig. 3. Mutual Recognition Delegation Strategy

V. CoinMingle Scheme

In this section, we present a new decentralized anonymous cryptocurrency scheme based on the one-time output address, CoinJoin and our mutual recognition delegation strategy. The consensus of CoinMingle is mainly PoW and stipulates partial transaction fee as penalties to defend the attacks of malicious insiders. Proceeding to present the details, we first give an intuition of our scheme. We introduce CoinMingle scheme with three phases.(1) commitment, (2) mutual recognition delegation, (3)and a punish phase which is invoked when an error or malicious behaviour is detected in the two previous phases. Then, we give an overview of CoinMingle including some consensus-based strategies and the structure of package.

A. Commitment

Transaction package is the foundation of CoinMingle scheme, a package, the basic unit for miner to be noted on the block, can be seen as a container of mixing transaction. In the commitment phase, a package header will be established. Suppose Bob wants to make trades, with the public key $pk = H_s(rA)G - B$ and the public parameter $R = rG$. A normal commitment includes three situations:

1) **Creat a package.** If Bob wants to be the first one preparing the transaction, then Bob will submit not only his pre-output serial number sn, his public key pk, the transaction fee Tx_fee and the failed return script Re_script but also the locked time $lock\_time$, the timestamp TS, the lower boundary of anonymity $A\_bound = m$ and an input script i'_{script} of commitment which includes his signature on the above messages. So Bob submits the tuple(sn, pk, Tx_fee, Re_script, $lock\_time$, TS, A_bound, i'_{script}).

2) **Fill the header.** If Bob just wants to be a participant of a package which has been created, then he is supposed to submit the tuple $(sn,\ pk,\ Tx\_fee,\ Re\_script,\ i'_{script})$.
3) **Seal the header.** If Bob find that the number of the participants are up to anonymity lower bound after his submitting, Bob should seal the package with a timestamp and an input script (including the signature for all the messages in header). That is, the tuple $(sn,\ pk,\ Tx\_fee,\ Re\_script,\ TS,\ i'_{script})$.

B. Mutual Recognition Delegation

We give some details in the following. Suppose Bob wants to send v coins to Carol (the output script corresponding to Carol is o_{script}), and Bob assigns Smith to be right sharer (Smith is i-th participant in the mixing). The group gpk is composed by all the public key of the participants.

- **Left_share.**

Bob broadcasts the following message.

$$\sigma = Enc_{pk_i}(R\_sign(v \parallel o_{script}, sk, gpk) \parallel v \parallel o_{script})$$

We express $R\_sign(v \parallel o_{script}, sk, gpk)$ in term of c, and $v \parallel o_{script}$ in term of mes.

- **Right_share.**

Once Smith receives a message in the left share, he will run based on the following pseudo codes.

Algorithm 1 Right Share

Input: message σ, group gpk, private key sk_i
Output: right sharing message σ'_i

```
Decrypt σ with sk_i, then get mes' and c'
if mes' is meaningful then
    b = Verify(c, mes', gpk).
    if b == 1 then
        sign_i = Enc_{sk_i}(mes')
        σ'_i = mes' ∥ sign_i
    else
        return 0
    end if
else
    return 0
end if
return σ'_i
```

After the step 2, Smith will submit the output messages to the mixing package. v' and o'_{script} of the algorithm above will be defined as O_i in an output set O which will be seen by Bob.

- **Verify.**

Once all the participants have submitted their results (maybe a zero vector) in last step, it comes into the verify step. For Bob, he will run based on the following pseudo codes.

Algorithm 2 Verify

Input: output set O, Smith index i, header message h, output message $\{v \parallel o_{script}\}$ and private key sk
Output: an input script i_{script}

```
flag = 0
for all O_j ∈ O do
    if {v ∥ o_script} == O_j and j == i then
        i_script ← Enc_sk(h ∥ O)
        flag = 1
    end if
end for
if flag == 1 then
    return i_script
else
    return 0
end if
```

C. Punish

Note that there is a recall script to deal with the timeout. If the attacker has joined the package but doesn't submit a valid transaction, the header of package can be noted onto the block. In other word, the attacker's penalties (may be ten percents of transaction fee) will also be punished by miner with no need for a complete transaction. Once a miner find a package in transaction pool, he will check the following situations.

1) Check the completeness of header. If the number of participants in header is less than anonymity bound, miners will drop it.
2) Check the completeness of package. In this case, the header is completed by default. Miners check the state of the package, and if it is unlocked, they will check if the number of the input scripts is equal to the cardinal of the group. If not they will note the header only on the block.
3) Check the validity of package. In this case, the package is completed by default and the package is unlocked. Miners will verify the output amounts and input scripts. If the output amounts is equal to input amounts (including transaction fee) and input scripts is correct, they will note all the package on the block, or they will note the header only on the block.

D. Overview of CoinMingle

As the above phases show, it is allowed for miner to note the header only, so how to resist the miner noting header only to solve the hash puzzle more quickly. The first tip is the transaction fee of a header only package is less than a full package, so rational miner will not abandon a full package. Further, there is a branch selection function $f(t, \mu)$ in CoinMingle consensus, positively correlated with completeness μ, instead of the longest chain selection strategy in Satoshi. In other words, the honest miner tend to append their block after the one which has more complete packages over the same period of time.

Finally, we give an overview of the structure of CoinMingle package at Fig.4.

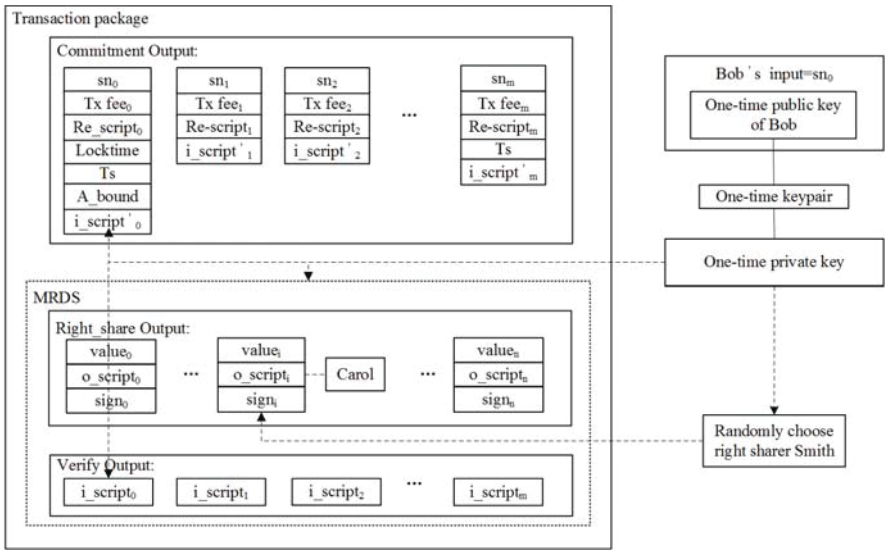

Fig. 4. CoinMingle Package

VI. Discussion of System Properties

In this section, we show that CoinMingle fulfils the requirements presented in Section 3 (consistency is general). We explain with the one-third malicious peers assumption, why CoinMingle can achieve a correct mixing in Section 5.1. In Section 5.2 we evaluate the anonymity quantitatively. This is also the basis for our analysis of flexibility in Section 5.3. Finally, we provide a comprehensive performance comparison between CoinMingle and other schemes in Section 5.4.

A. Correctness

Malicious Input Peers. In the commitment phase, the input peers broadcast their input UTXO and wait for mixing. The correctness of this phase depends on the Bitcoin script scheme. So malicious input peers can not interrupt a submittal in commitment phase. In the MRDS phase, a common method of attack is refusing to submit the left sharing message. However, the left sharer must be the participant in the commitment phase, once the left sharer does not meet corresponding requirement in time, the penalties will be confiscated by miner. So malicious input peers cannot interrupt the mixing without any costs.

Malicious Mixing Peers. In the MRDS phase, malicious mixing peers usually refuse to verify the left sharing message and output a right sharing message. For one thing, the right sharer meets the same dilemma with left. For another, the left one has a chance to change the right sharer. So the transaction will not be stalled by malicious mixing peers.

B. Anonymity

Before we describe our anonymity analysis, we are supposed to find out the anonymity problem of "CoinShuffle class". In this section, we will focus on CoinParty. Each mixing peer M_i in CoinParty encrypts his output address O_i using the public keys $K_1, \ldots, K_m$ and gets $[O_i]_{K_1:K_m} := E_{K_1}(E_{K_2}(\ldots E_{K_m}(O_i)))$. Then he broadcasts this result. So participants in the mixing can catch this message, and they even know the order of the layered encryption. After the mixing, they can get the plaintext O_i and encrypt it with stored order again to find out the link of O_i and IP address. For inner attackers, the layered encryption seems to be unnecessary. So the CoinParty pins its hope on network Tor to unlink the IP address and true identity.

CoinMingle scheme will provide users anonymity with no need for network Tor in unlinkability level. Express the mixing times and cardinal of group in terms of n and m. With the one-third malicious peers assumption, the probability of being linked is:

$$Pr = \sum_{i=0}^{n} (\frac{1}{3})^{n-i} \cdot C_n^i \cdot (\frac{3}{2m})^i \tag{1}$$

The following is the graphics of the anonymity:

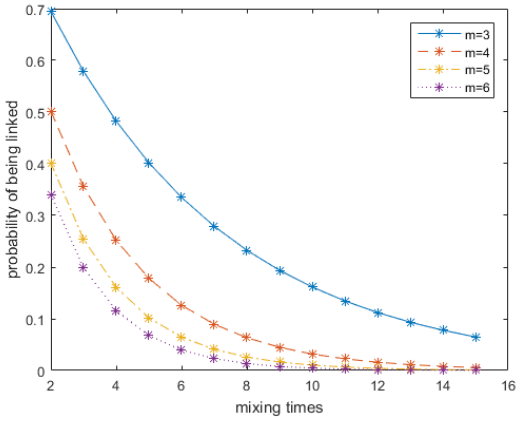

Fig. 5. Anonymity

As Fig. 5 shows that when the cardinal of the group is bigger than 4 or the mixing times is more than 4, the identity of the user is considered to be indistinguishable.

C. Flexibility

There is a visualization example as 6: Bob have two sums of coins 20 and 30, paying 40 coins to Carol. And David have 15 coins, paying 10 coins to Ella.

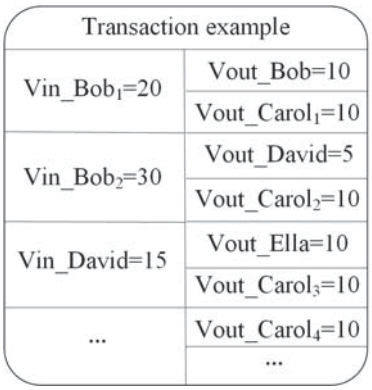

Fig. 6. An Example

In this way, our anonymity model is applicable, so we strongly advise that users in mixing can imitate the minimum spending as far as possible, And it can be allowed to follow some optimisation strategies [25].

D. Efficiency Analysis

In this section, we give a brief comparison of the efficiency of CoinMingle with some other schemes. Details can be found in Table I.

TABLE I
EFFICIENCY

Ref.	Input peer	Mix Peer	Verifier	Comm
CoinJoin	$m \cdot PM$	$PM + \pi$	PM	$4m \cdot \|G\|$
CoinParty	$m \cdot PM$	$PM + \pi$	PM	$4m \cdot \|G\| + m \cdot \|H_s\|$
Ours	$3PM$	PM	PM	$4\|G\|$

"m": the cardinal of group of input accounts; "PM": an dot multiplication operation in group G_1; "$|Hs|$": the output size of hash function; "$|G|$": the size of point in elliptic curve.

Just as table I shows, CoinMingle has advantages over other CoinJoin schemes in being storable and computing.

VII. Conclusion

In this work, we formalize a new efficient decentralized anonymous cryptocurrency schemes CoinMingle. Compared with ZCash, the technology is more easily understandable and dont need the privileged as well; compared with CoinJoin, our CoinMingle scheme is more flexible and credible; compared with Monero, our CoinMingle scheme needs very less storage space which is independent to the cardinal of group of input accounts. In addition, our CoinMingle scheme is friendly to users, for no need for a Tor network.

References

[1] S. Nakamoto, " Bitcoin: a peer-to-peer electronic cash system," bitcoin.org, 2008,unpublished.

[2] F. Reid and M. Harrigan, "An analysis of anonymity in the Bitcoin system," IEEE Third International Conference on Privacy, Security, Risk and Trust, 2012, pp. 1318–1326.

[3] Cazabet Remy, Baccour Rym and Latapy Matthieu, "Tracking Bitcoin users activity using community detection on a network of weak signals," International Workshop on Complex Networks and their Applications, 2018, pp. 1318–1326.

[4] S. Barber, X. Boyen, E. Shi and E. Uzun, "Bitter to better - how to make Bitcoin a better currency," Financial Cryptography and Data Security, 2012, pp. 399–414.

[5] D. Ron and A. Shamir, " Quantitative analysis of the full Bitcoin transaction graph," Financial Cryptography and Data Security, 2013, pp. 6–24.

[6] E. B. Sasson, A. Chiesay, C. Garman, M. Green, I. Miers, et al, "Zerocash: decentralized anonymous payments from Bitcoin," IEEE Symposium on Security and Privacy (SP), S. Jose, California, pp. 459–474.

[7] E. Duffield and D. Diaz, "Dash: a privacy-centric crypto-currency," unpublished.

[8] A. Greenberg, "Dark Wallet: is about to make bitcoin money laundering easier than ever," 2014, https://www.wired.com/2014/04/dark-wallet/,unpublished.

[9] T. Ruffing and P. M. Sanchez and A. Kate, "CoinShuffle: practical decentralized coin mixing for Bitcoin," Computer Security - ESORICS 2014, 2014, pp. 345–364.

[10] J. H. Ziegeldorf, F. Grossmann, M. Henze and N. Inden and K. Wehrle, "CoinParty: secure mMulti-party mixing of Bitcoins," ACM Conference on Data and Application Security and Privacy, S. Antonio, TX, USA, 2015, pp. 75–86.

[11] Nicolas van Saberhagen, "CryptoNote v 2.0," https: //cryptonote. org/ whitepaper. pdf, unpublished.

[12] C. Yuan and M. X. Xu and X. M. Si, "A new aggregate signature scheme in cryptographic currency," International Journal of Performability Engineering, vol. 13(5), 2017, pp. 754–762.

[13] J. Bonneau, A. Narayanan, M. Andrew, C. Jeremy, A. K. Joshua and W. F. Edward, "Mixcoin: anonymity for Bitcoin with accountable mixes," Financial Cryptography and Data Security, 2014, pp. 486–504.

[14] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer and M. Virza, " SNARKs for C: verifying program executions succinctly and in zero knowledge," Advances in Cryptology C CRYPTO 2013, 2013, pp. 90–108.

[15] E.Sasson, A. Chiesa, E. Tromer and M. Virza, " Succinct non-interactive zero knowledge for a von Neumann architecture," SEC'14 Proceedings of the 23rd USENIX conference on Security Symposium, 2014, pp. 781–796.

[16] J. Bonneau, A. Narayanan, M. Andrew, C. Jeremy, A. K. Joshua and W. F. Edward, "Mixcoin: anonymity for Bitcoin with accountable mixes," Financial Cryptography and Data Security, 2014, pp. 486–504.

[17] BobAlison, "Shared Coin - free trustless private bitcoin transactions," 2014, https://sharedcoin.com/,unpublished.

[18] AdamISZ and belcher, "Joinmarket - CoinJoin that people will actually use," https://bitcointalk.org/index.php?topic=919116.msg10096563,unpublished.

[19] K. Atlas, "Weak privacy guarantees for SharedCoin mixing service," http://www.coinjoinsudoku.com/advisory/,unpublished.

[20] J. K. Liu, V. K. Wei and D. S. Wong, " Linkable spontaneous anonymous group signature for ad hoc groups," Information Security and Privacy 11th Australasian Conference - ACISP, 2004. Sydney, Australia, vol. 2004, pp.325-335.

[21] A. Kumar, C. Fischer and S. Tople and P. Saxena, "A traceability analysis of Moneros blockchain," Security and Communicatin Networks, vol. 2017(2), 2017, pp. 153–173.

[22] R. L Rivest, A. Shamir and Y. Tauman,"How to leak a secret: Theory and applications of ring signatures", In Theoretical Computer Science, 2006, pp. 164–186.

[23] R. L. Rivest, A. Shamir and L. M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,"Communications of the ACM, 21(2),1978, pp.120C-126.

[24] Ivan Damgrd, "On the existence of bit commitment schemes and zero-knowledge proofs," Advances in cryptology - CRYPTO 89, 1989, pp. 17–27.

[25] F. K. Maurer and T. Neudecker, "Anonymous CoinJoin transactions with arbitrary values," International Conference on Electronics and Software Science, Takamatsu, Japan, 2017, pp. 486–504.