A Traffic-Analysis Proof Solution to Allow K-Anonymous Payments in Pseudonymous Blockchains

Francesco Buccafurri
bucca@unirc.it
Università Mediterranea
Reggio Calabria, Italy

Vincenzo De Angelis
vincenzo.deangelis@unirc.it
Università Mediterranea
Reggio Calabria, Italy

Sara Lazzaro
sara.lazzaro@unirc.it
Università Mediterranea
Reggio Calabria, Italy

ABSTRACT

Pseudonimity in blockchain often misses the goal of effectively hiding the actual identity of users. Also anonymous blockchains such as Monero and ZCash can be de-anonymized through network traffic analysis. In this work, we present a solution to achieve k-anonymity guarantees (resisting traffic analysis attacks) in pseudonymous blockchains. The idea underlying our solution is to organize users in rings of cover transactions, through which users indistinguishably exchange actual data or random noise and the initiator is hidden within the ring. Importantly, this mechanism does not require off-chain communication.

KEYWORDS

Pseudonymity, Traffic analysis attacks, k-anonymity

1 INTRODUCTION AND MOTIVATION

Blockchain is a distributed ledger that keeps track of the occurrence of events. An entity can generate a transaction toward another entity to exchange a value. This transaction is validated by peers participating in the network, and thus does not require any third-trusted party to be validated.

A relevant feature offered by the most known blockchains (e.g., Ethereum and Bitcoin) is pseudonymity. Each user is associated with an address (not directly linked to the real identity of the user) that allows them to send and receive cryptocurrency. Nevertheless, all the transactions a user makes with the same address are linked among them. In the literature, several works were proposed concerning the de-anonymization of blockchain addresses [4] also in the case a single user leverages multiple addresses [3]. In this case, the goal is to create a cluster of the addresses belonging to the same user and possibly associate it with external information (such as the IP address), thus de-anonymizing the blockchain address.

Different blockchains such as Monero and ZCash offer full anonymity [1] in place of pseudonymity, by making the transactions made by the same user unlinkable to each other. However, as shown in [2], effective de-anonymization attacks based on network analysis can be performed even against anonymous blockchains.

In this preliminary work, we propose a solution to achieve anonymity guarantees in pseudonymous blockchains resisting traffic analysis attacks. Specifically, we aim to hide the sender activity (i.e., the fact that a user generates a transaction) in an anonymity set of k users also against a global adversary observing the entire traffic (at the network layer) exchanged in the network. Importantly, our approach does not require off-chain communication channels.

2 THE PROPOSED APPROACH

The aim of our approach is the provision of a mechanism allowing users to send money in a k-anonymous way.

The idea at the basis of this solution consists of organizing the users in groups of k users, called *rings*. A ring represents an anonymity set of size k, in the sense that each user in a ring who generates a payment in favor of any (even external) user cannot be distinguished among the other $k-1$ users of the ring. From another point of view, a ring represents also a way to implement a private distributed ledger among the k users composing the ring. We will see below the exact meaning of this point. In the ring an information hiding mechanism is enabled, consisting of continuous execution of cover transactions moving in a circular fashion. This way, the users indistinguishably exchange actual data or random noise and the initiator is hidden within the ring. This mechanism is better explained below. Our solution leverages the typical built-in features of any pseudonymous blockchain supporting smart contracts. In this work, we refer to the Ethereum blockchain.

Now we see how payments are implemented. Each user deposits, in a dedicated smart contract, a given amount of cryptocurrency, which can be only spent for k-anonymous payments. All the k users are aware of this special account balance of any user in the ring. Actually, the account balances are the result of the agreement of users in the ring. In this sense, the ring implements a sort of internal ledger, as anticipated earlier. Each payment must be authorized by a threshold of t other users of the ring, by following a classical BFT approach for consensus (typically, t/k can be set to $2/3$). Specifically, we require that t confirmations, among the k users forming the ring, are needed to authorize a transaction from the smart contract to a given recipient. To achieve anonymity, the transaction authorizing the smart contract to spend cryptocurrency is not provided directly by the actual sender, but all the users of the ring are involved in the process. As outcome of the solution, we obtain k-anonymity against any external observer different from the k users also with the capability to monitor the entire network traffic (global passive adversary).

2.1 Ring Construction

In this section we describe how rings are built.

The first requirement is that ring construction should happen in a fully decentralized way, such that no off-chain interaction is performed. To reach this goal, we propose a Distributed Hash Table implemented via the smart contract that allows the users to find the ring to which they belong, through their Ethereum address. However, if this computation depended only on the Ethereum address, another problem would arise. Indeed, an adversary could generate a lot of Ethereum addresses in order to find at least t addresses that would belong to the same ring. This way, they would be able to control the ring and spend the cryptocurrencies of the other users.

We recall that, for each user belonging to a ring, an initial deposit is paid. This also works as a disincentive for an adversary. Moreover, if users were not able to precompute the ring in which they would fall, the adversary would have to generate (and pay for) a greater number of addresses to increase their chance to have at least t address in the same ring. Then, the economical effort required from the adversary would be high. Thus, we have to prevent an adversary from having control of at least t addresses in the same ring.

To achieve this, we propose the following mechanism. Suppose our system supports n users to split into rings of size k. As above mentioned, each user deposits a given amount of cryptocurrency in the smart contract to become part of a ring. To do this, the users simply invoke a function of the smart contract that collects their deposits and stores their Ethereum addresses. However, the rings are not formed until the n^{th} user asks for joining a ring. When the n^{th} user joins the ring, the invoked function performs in a slightly different way. After collecting the deposit and storing the Ethereum address of the n^{th} user, the function retrieves and stores the number of the last block included in the blockchain. The idea is to use the hash of the next block as an unpredictable value to implement the distributed Hash Table and prevent any adversary to precompute the rings to which the users belong.

Specifically, when the block including the last transaction is mined and added to the blockchain, any entity can invoke another function that retrieves the digest of such a block, say D. Then, for each user u with Ethereum address Eth_u, a value $r_u = H(Eth_u||D)$ is computed and associated with Eth_u, where H denotes a cryptographic hash function (e.g., Keccak256). Then, these values are ordered in an increasing way to form the rings. Specifically, the first k values form the first ring, the next k values form the second ring, and so on. Observe that, since D is not known before the last user joins the system, the users cannot precompute in advance the rings they fall.

2.2 The Anonymous Protocol

In our protocol, each user communicates through the blockchain by generating a transaction toward the next user in the ring. The next of the user u is the user associated with the smallest value greater than $H(Eth_u||D)$ or the smallest value of the ring (in the case $H(Eth_u||D)$ is the maximum value).

A transaction without any amount of cryptocurrency is sent by a user toward the next user in the ring. Such a transaction contains some data encrypted with the public key of the next user. After receiving this transaction and processing it, the receiving user forwards it to the next user (possibly after waiting an amount of time). Initially, the data exchanged in the transaction are random bytes. However, when needed, these data can be replaced with actual information encrypted in such a way that an external observer cannot distinguish between actual data and random ones, thus not identifying the originator of the information.

Consider a user u who wants to generate an anonymous payment of c Ethers toward a recipient r. The user waits for a transaction from the previous user in the ring containing random bytes. Then, it replaces these bytes with $Eth_u, H(Eth_u||s), s, r, c, Eth_{u'}$, where $H(Eth_u||s)$ represents the digest of the Ethereum address of u concatenated with a salt s, while $Eth_{u'}$ represents the Ethereum address of another randomly selected user u', called *exit-user*. Moreover, u adds to the transaction a signature σ_u, verifiable through the same public key associated with Eth_u, of all the data included in it. Then, the above information is encrypted with the public key of the user succeeding u in the ring and sent to them via a transaction. At this point, the transaction makes a complete loop of the ring and all the users crossed in the ring locally store the data of this transaction. Once the transaction reaches u, they send the same transaction to their next in the ring. Then, the transaction continues to turn in the ring from each user to the next.

Each user placed in the ring beyond u, after receiving the transaction, decrypts it and sees that it contains a transaction request from u. Then, they verify the signature σ_u. Finally, they check their address against that of the chosen *exit-user* u'. Now, two cases might occur: (1) their address precedes that of u' in the ring (i.e., u' can be reached, with a lesser number of users, from the current user by moving forward), (2) their address coincides with that of u' or their address follows that of u' in the ring (i.e., u' can be reached, with the lesser number of users, from the current user by moving backward).

In case (1), the user checks if the current transaction was seen in the ring more than one time. If it is not the case, the received transaction is encrypted (as it is) with the public key of the next user in the ring and sent to them. Otherwise, if the transaction was already seen in the ring more than one time, then the user performs as in case (2), explained below. In case (2), the user performs as follows. First, the user invokes a function of the smart contract, passing as input $H(u||s), r, c$. Observe that, the first user generating such a transaction is the *exit-user*. Then the smart contract locally stores $H(u||s), r, c$ along with the information that u' approved the requested transaction. Recall that, to perform the required transaction, the smart contract must receive at least t authorizations. At the same time, the transaction in the ring is sent to the following users thus making a complete loop in the ring. This way, all the users in the ring approving the requested transaction send to the smart contract the above-described authorization. Once the smart contract receives the t^{th} authorization, it performs the required transaction, i.e., it transfers the amount c toward r. Finally, all the users in the ring update the balance of u.

REFERENCES

[1] Nitish Andola, Raghav, Vijay Kumar Yadav, S. Venkatesan, and Shekhar Verma. 2021. Anonymity on blockchain based e-cash protocols—A survey. *Computer Science Review* 40 (2021), 100394. https://doi.org/10.1016/j.cosrev.2021.100394

[2] Alex Biryukov and Sergei Tikhomirov. 2019. Deanonymization and linkability of cryptocurrency transactions based on network analysis. In *2019 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 172–184.

[3] Dmitry Ermilov, Maxim Panov, and Yury Yanovich. 2017. Automatic bitcoin address clustering. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 461–466.

[4] Hao Hua Sun Yin, Klaus Langenheldt, Mikkel Harlev, Raghava Rao Mukkamala, and Ravi Vatrapu. 2019. Regulating cryptocurrencies: a supervised machine learning approach to de-anonymizing the bitcoin blockchain. *Journal of Management Information Systems* 36, 1 (2019), 37–73.