Re-thinking untraceability in the CryptoNote-style blockchain — The Sun Tzu survival problem

Jiangshan Yu*, Man Ho Allen Au†, and Paulo Esteves-Verissimo‡

*Monash University, Australia
†Hong Kong Polytechnic University, China
‡University of Luxembourg, Luxembourg

***Abstract*—We develop new foundations on transaction untraceability for CryptoNote-style blockchain systems. In particular, we observe new attacks; develop theoretical foundations to model transaction untraceability; provide the least upper bound of transaction untraceability guarantee; provide ways to *efficiently* and *automatically* verify whether a given ledger achieves optimal transaction untraceability; and provide a general solution that achieves provably optimal transaction untraceability.**

Unlike previous cascade effect attacks (ESORICS' 17 and PETS' 18) on CryptoNote-style transaction untraceability, we consider not only a passive attacker but also an active adaptive attacker. Our observed attacks allow both types of attacker to trace blockchain transactions that cannot be traced by using the existing attacks. We develop a series of new games, which we call "The Sun-Tzu Survival Problem", to model CryptoNote-style blockchain transaction untraceability and our identified attacks. In addition, we obtain seven novel results, where three of them are negative and the rest are positive. In particular, thanks to our abstract game, we are able to build bipartite graphs to model transaction untraceability, and provide reductions to formally relate the hardness of calculating untraceability to the hardness of calculating the number of perfect matchings in all possible bipartite graphs. We prove that calculating transaction untraceability is a $\#P-$complete problem, which is believed to be even more difficult to solve than NP problems. In addition, we provide the first result on the least upper bound of transaction untraceability. Moreover, through our theoretical results, we are able to provide ways to efficiently and automatically verify whether a given ledger achieves optimal transaction untraceability. Furthermore, we propose a simple strategy for CryptoNote-style blockchain systems to achieve optimal untraceability. We take Monero as a concrete example to demonstrate how to apply this strategy to optimise the untraceability guarantee provided by Monero.

1. Introduction

Crypto-technique plays an important role in providing anonymity.

Due to the potential sensitivity of transactions on blockchains, CryptoNote [17] was proposed to protect transaction untraceability in blockchain-based cryptocurrencies. As of November 2018, 18 blockchain systems have adapted the CryptoNote protocol [4], and 11 of them are cryptocurrencies with a large market capitalization. For example, Monero, one of the most popular CryptoNote-style cryptocurrencies with a market capitalization of USD 1.8B, is ranked top 10 on the CoinMarketCap [1].

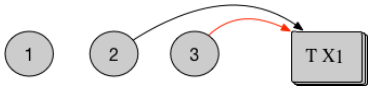
Unlike in Bitcoin where an observer of the blockchain can learn the trace of every coin in the ledger, CryptoNote hides the trace of coins. In particular, the input of a CryptoNote transaction not only contains the to-be-spent coin, but also contains several other coins of the ledger, called mix-ins, as noise to confuse the observer (Figure 1). This reduces the certainty of guessing the actual to-be-spent coin of a transaction. We use "real input" to refer the to-be-spent coin of a transaction, use "decoy input" to refer a mix-in, and use "anonymity set" to refer the entire input including both real input and decoy input. As illustrated in Figure 1b, transaction TX_1 has two inputs, namely a real input '3', and a decoy input '2'. They form the anonymity set of TX_1. An observer only knows that one of the two inputs is real, but cannot learn which coin is the real input.

Given an anonymity set of size ℓ, the traceable ring signature used in CryptoNote [17] guarantees that no one can make a correct guess on the real input with probability greater than $\frac{1}{\ell} + \epsilon$, where ϵ is negligible. This idea has been adopted by many other cryptocurrencies, of which Monero [12] using one-time linkable ring signature rather than traceable signature, is a leading example.

Crypto-technique alone is not enough. Recent identified attacks [6], [9] show that the use of cryptographic techniques is not sufficient to protect transaction untraceability in the CryptoNote-style blockchains. For example, in Figure 1b, crypto-technique guarantees that the probability of making a correct guess on the real input of transaction TX_1 is $\frac{1}{2}$ when no extra information is available. However, if auxiliary information allows the attacker to rule out the decoy input, then the observer can confirm that the other input is the real one. This is outside of the cryptographic model of the above mentioned schemes.

1. https://coinmarketcap.com. Data fetched on 10.Nov.2018.

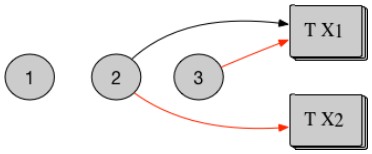
(a) Bitcoin



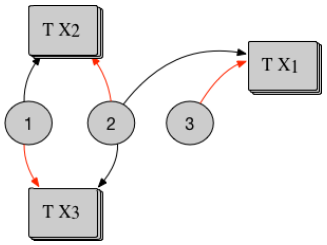
(b) CryptoNote

Figure 1: A view on coin trace in different cryptocurrencies. Each circle is a coin, and a rectangle is a transaction. A red (or black) arrow shows the real (resp. decoy) input of a transaction. In Bitcoin, each transaction clearly states its input, and does not provide any untraceability. With CryptoNote, decoy inputs are added into a transaction to hide the real input.

Auxiliary information can be obtained directly from the blockchain. For example, in Monero, 65.9% of all real inputs have zero mix-ins [6]. In other words, these real inputs were spent in plain. As shown in the below example (Figure 2a), zero mix-ins not only break the untraceability of zero mix-in transactions, but also affect other transactions that have included decoy inputs.



(a) An example of zero mix-in transaction.



(b) An example of inference attack.

Figure 2: Examples of auxiliary information that breaks transaction untraceability.

Consider transaction TX_1 in Figure 2a. This transaction was created with real input '3' and decoy input '2'. Ideally an observer should not be able to learn that coin '3' is the real input. However, if later transaction TX_2 has been created by using '2' as real input with zero mix-ins, i.e. no decoy input is used, then any observer can conclude that '3' is the real input of TX_2 (since coin '2' can be ruled out). This breaks transaction untraceability.

Enforcing a minimum mix-in size does not work either. Later versions of Monero enforce a minimum mix-in size to avoid attacks based on zero mix-ins. Unfortunately, existing work shows that coin age could reveal extra information — the youngest coin is very likely to be the real input of an anonymity set [6], [9]. This breaks the transaction untraceability even when a minimum mix-in size is enforced.

Inference attack – decoy input selection is vital to untraceability. Suggestions [6], [9] on how to reduce the information leaked by coin age have been proposed, however, we identify a new class of attacks that break the transaction untraceability even if the coin age does not leak anything. We call our newly identified attacks "inference attacks", which break transaction untraceability by only observing how decoy inputs (i.e., mix-ins) are selected. We demonstrate this attack by using the following example.

As shown in Figure 2b, TX_1 is the same as before. After the creation of TX_1, two new transactions, namely TX_2 and TX_3, are created. In particular, TX_2 is created by using coin '2' as real input and coin '1' as decoy input, and TX_3 is created by using coin '1' as real input and coin '2' as decoy input. Since each coin can only be spent once, and the anonymity sets $\{1,2\}$ of both TX_2 and TX_3 are the same, we know that coin '2' (as well as coin '1') must be the real input of either TX_2 or TX_3. Thus, an observer can conclude that only '3' can be the real input of TX_1[2]. However, at the time when transaction TX_1 was created, the payer of TX_1 has no clue about the involvement of coin '2' and '3' in later transactions.

This example shows that anonymity sets of some transactions may leak information about other transactions, and this is out of payers' control. So, coordination and system-wide rules on decoy input selection are necessary to provide better and predictable transaction untraceability.

What is missing in existing security models? Existing models try to capture the privacy requirement such that no additional information beyond the publicly available information shall be leaked. (For example, no information should be leaked from the cryptographic protocol.) However, they did not consider how the publicly available information would affect the transaction untraceability. For example, CryptoNote models [17], [14] require that given a transaction transcript from two possible payers, it is impossible for an attacker to guess correctly the actual payer with probability significantly over 50%. However, it did not consider the inference from other transactions. Ledger indistinguishability [1] requires that given the view of two ledgers with publicly consistent information, it is impossible for an attacker to match correctly a view with the ledger. However, it did not consider the information leaked within the same ledger based on the relations among anonymity sets of different transactions.

How to analyse transaction untraceability? At this point, we revisit the design goal of cryptocurrencies using decoy inputs as the main tool to protect transaction untraceability. In particular, we define two metrics, namely individ-

2. Since a coin can only be spent once, if coin '2' is the real input of TX_1, then it cannot be used again for TX_2 or TX_3. So, TX_2 and TX_3 must share the same real input, i.e., coin '1'. This forms a contradiction. Thus, only coin '3' can be the real input of TX_1.

ual untraceability and global untraceability, to evaluate the transaction untraceability against inference attacks.

From a payer's point of view, the optimal transaction untraceability would be the case that the de-anonymization of his transaction implies the compromise of all other decoy inputs in the anonymity set. We consider this kind of transaction untraceability individual untraceability. We define individual untraceability of a transaction as the metric to measure the difficulty of de-anonymising this transaction. The difficulty is defined as the maximum number of coins an attacker needs to corrupt in order to identify the real input with certainty. If the anonymity set of a transaction is of size ℓ, then the optimal individual untraceability is $\ell - 1$. For example, for transaction TX_1 in Figure 1b, the size of its anonymity set is 2, and its optimal individual untraceability is 1. We say that *a ledger achieves optimal individual untraceability if all transactions in the ledger achieve optimal individual untraceability.*

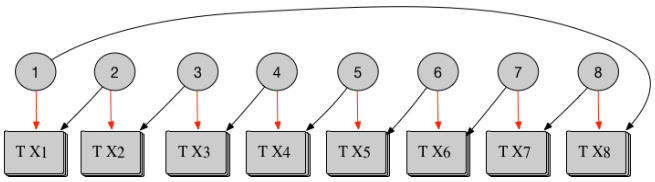
In contrast, global untraceability of a ledger measures the difficulty of de-anonymising all transactions in the ledger. Specifically, it is the number of coins required to be corrupted to de-anonymise all transactions in the ledger. For example, the global untraceability of Figure 2b is 1 — to identify the real input of all transactions, one only needs to know to which transaction the coin '1' (or coin '2') is the real input.

For a user, the global untraceability as a property may not be as interesting as the individual untraceability. However, the global untraceability is very useful as a helper property to assist us to analyse the blockchain transaction untraceability. In fact, we show in Section 5.2 that the notion of global untraceability allows us to obtain solutions achieving optimal individual untraceability, and prove that if a system achieves optimal global untraceability against a passive adversary, then it achieves optimal individual untraceability against active adaptive adversary. (An intuitively reasonable but flawed straw man definition of the global untraceability is discussed in Section 7, to assist readers further to understand our modeling considerations.)

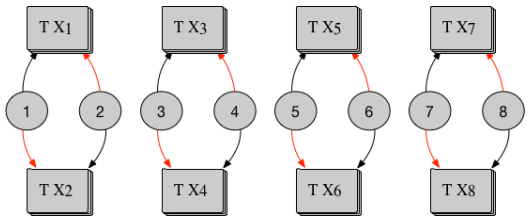
Transaction untraceability against an active adaptive attacker. Existing studies [6], [9] on the transaction untraceability only consider a passive attacker who can only observe the blockchain public information. However, in the real world, an attacker can be an 'insider', i.e., an attacker can own coins and can adaptively create transactions. So, the attacker will be able to learn even more information than learning merely through passive blockchain analysis. This makes the attacker much more powerful and makes the analysis even more difficult. We use the following example to illustrate the analysis of transaction untraceability.

Given two ledgers (A and B as shown in Figure 3) that provide optimal individual untraceability against a passive observer, we show that they could behave very differently when the adversary becomes an insider (e.g. a payer). At the extreme case, knowing the real input of one transaction could result in the de-anonymisation of *all transactions* in ledger A but *only one single transaction* in ledger B.

Consider the example ledger A and B as presented in



(a) Ledger A



(b) Ledger B

Figure 3: Example ledgers to demonstrate inference attacks from an active attacker.

Figure 3a and Figure 3b, respectively. Both A and B have 8 transactions, and each transaction has exactly one real input and one decoy input. In A, for all $i \in [1, 8]$ the anonymity set of the transaction TX_i is $(i, i + 1 \mod 8)$, where i is the real input and $i + 1 \mod 8$ is the decoy input. In B, for all $i \in [1, 8]$ the anonymity set of transaction TX_i is $(i, i + 1)$ when i is odd, otherwise it is $(i - 1, i)$; where in both cases the first element is the real input and the second element is the decoy input. Note that this example is only used to simplify the presentation. In practice, an attacker cannot identify the real input by checking if it is the first element of the anonymity set (for example, elements in the set may be ordered randomly).

It is obvious that the individual untraceability (against a passive observer) is 1 for all transactions in both system A and B. If the attacker is an insider, then it is easy to see that corrupting any coin in system A would allow the attacker to de-anonymise all these 8 transactions; whereas corrupting any coin in system B only allows the attacker to de-anonymise one additional transaction. For example, in system A if an attacker is certain that coin '1' is the real input of TX_1, then the attacker can deduce that coin '8' is the real input of TX_8, coin '7' is the real input of TX_7, and all the way back until de-anonymising all transactions. In contrast, in system B the attacker with the same knowledge can only conclude that coin '2' is the real input of TX_2, but has no clue about other transactions.

1.1. Our Contribution.

This work provides theoretical foundation on transaction untraceability for CryptoNote-style blockchains. We have four main contributions. First, we observe a new attack, called inference attack, on variants of CryptoNote protocol. In particular, we consider not only an observer but also an insider to launch such attacks. This new attack demonstrates

the importance of decoy input selection, and is stronger than the previous cascade attacks. Second, we propose a novel model to formalize the inference attack and untraceability properties, and we separate individual untraceability and global untraceability. In particular, we design a series of games, which we call "The Sun Tzu Survival Problem", to model the transaction untraceability and the inference attacks. Third, we provide ways to *efficiently* and *automatically* verify whether a given ledger achieves optimal untraceability, both for each individual transaction and for the entire ledger, in the light of inference attacks. The verification of whether a given system achieves a certain property is very challenging. For example, with crypto algorithms, it is difficult to determine whether a given system achieves IND-CCA2 security — one has to either formally prove that the system achieves it, or to find an attack to demonstrate the insecurity of the system. The former, i.e., manual proof, is error prone and requires a lot of time, effort, and expertise. Similarly, the latter is also very challenging as the search space is too big. Our last main contribution is providing a general solution that achieves provably optimal individual untraceability and global untraceability, supported by The Bregman's Theorem [2] in graph theory.

More specifically, we obtain a sequence of seven positive and negative results regarding individual untraceability, global untraceability and their relationship.

- Our **first two results** are negative. First, we show that optimal individual untraceability cannot be achieved without system-wide mix-in selection rules. This has been explained in Figure 2. Second, we show in Section 4 that it is very difficult to analyse the individual untraceability of a given transaction in a system.
- Our **third result** is also negative. We show in Theorem 1 that for a given mix-in selection strategy, calculating its global untraceability is extremely hard. In particular, we show that this calculation is a $\#P$-complete problem, which is believed to be even more difficult than solving NP problems [16].
- Our **fourth result** is positive. We, for the first time, provide the least upper bound of global untraceability (in Theorem 3 and Theorem 4). In particular, we reduce finding optimal global untraceability to computing the upper bound for the number of perfect matchings in all possible bipartite graphs. We show that under mild restriction, it is possible to obtain optimal global untraceability.
- Our **fifth result** is also positive. We show in Lemma 2 that if a system achieves optimal global untraceability against a passive observer, then this system also achieves optimal individual untraceability against even a strong active attacker that can adaptively choose its target (as detailed in Section 4.2). In other words, optimal global untraceability (with a passive attacker) implies optimal individual untraceability (with an active adaptive attacker).
- Thanks to Theorem 2 and our fifth result, our **sixth result** is positive — we provide ways to efficiently and automatically verify whether a given ledger achieves optimal global untraceability and optimal individual untraceability.
- Our **seventh result** is also positive. According to Theorem 2 and our fourth result, we propose a mix-in selection strategy which achieves the optimal global untraceability. Again, the same restriction applies. Surprisingly, our strategy is very simple but effective, however, no previous work has discovered this solution before.

1.2. Experimental analysis and disclosure.

We have disclosed our research findings to the corresponding team of CryptoNote, Monero, Bytecoin, and DigitalNote. Due to the space limitation, we published the experimental analysis on our identified passive attacks in a seperate paper [18]. In particular, we designed an efficient algorithm to implement our inference attack, and applied it (in combination with the existing attacks [6], [9]) on several CryptoNote-style systems. As a result, we have successfully identified the real input of 70.516% - 91.56% transactions of these systems to date. Out of the identified transactions, for example, our attack has uniquely identified the real inputs of about 80,000 transactions in Bytecoin[3], and 5,000 transactions in Monero[4]. These transactions cannot be identified through previous attacks.

2. Background

Bitcoin [11] has established itself as the most intriguing and successful decentralized cryptocurrency to date. A Bitcoin transaction contains a list of inputs and outputs. An input is a reference to an output from a previous transaction, which can be seen as the address of a to-be-spent coin. An output states how many coins to be transferred to which account (i.e. public key). To prove the ownership of an input coin, the user needs to issue a signature on the transaction, by using the signing key associated to the public key of the referenced output. BitCoin is able to provide transaction unlinkability by requiring a fresh account (i.e. public key) for each output.

Monero is a privacy-centric cryptocurrency launched in 2014. It has established itself as an intriguing privacy-focused decentralized cryptocurrency. As of July 2017, Monero is one of the top 10 most traded cryptocurrencies [3]. The initial Monero daemon employs the original CryptoNote protocol [17], which leverages traceable signature to provide anonymity. In particular, it aims to guarantee that it is impossible to link any two transactions to the same user (unlinkability), or to trace any coin back to another transaction (untraceability). However, the size of

3. The market cap of Bytecoin is about $238 Million USD, and is the 36th in the cryptocurrency coinmarketcap, as of 10th Nov 2018.

4. The market cap of Monero is about $1.8 Billion USD, and is the 10th in the cryptocurrency coinmarketcap, as of 10th Nov 2018.

a cryptonote transaction is big and the decoy outputs (a.k.a. mix-ins) have to have the same denomination as the to-be-spent coin. These weaknesses lead to several practical issues and security problems [7], [9], [6]. In 2015, Monero proposed Ring Confidential Transactions (Ring-CT) based on linkable ring signature to improve the CryptoNote protocol. Ring-CT eliminates the need of matching the candidate mix-ins' denomination with the denomination of the to-be-spent coins, and reduces the size of ring signatures to half.

3. Threat model

System setting. We assume a role called payer, that spends coins and creates transactions; and another one called payee, that receives coins from transactions. Users can perform one or both of those roles. Transactions are included in the blockchain by miners through the mining process. The mining process, e.g. proof-of-work, is specific to the design choice of different systems.

Adversary model. We consider two types of adversary, namely passive adversary and active adaptive adversary. A *passive adversary* has read access to the blockchain, namely it can read all the transactions between payers and payees contained in the blockchain, but it cannot create any transaction. An *active adaptive adversary* can adaptively create transactions, compromise a coin holder, and learn the real payer of a chosen transaction.

Assumptions. In this work we only put our focus on studying how the mixin selection strategy would affect privacy guarantees, when the used crypto techniques are secure. So, our model does not consider the timing-related issue. For both type of adversaries, in our *Sun Tzu* model we assume that the adversary does not have the notion of *time*. In particular, the adversary does not have access to the time related information, such as coin age. (Later in Section 6 we will show how to address the timing-related attacks in practice.) In addition, we consider the blockchain as a snapshot of the globally agreed transaction state of the system, and we define and evaluate its privacy properties.

4. The Sun Tzu Survival Problem

As demonstrated previously, crypto-techniques are necessary but not sufficient to guarantee transaction untraceability against inference attacks. To study the inference attacks, we propose to use *The Sun Tzu Survival Problem* to model and express transaction untraceability, when the crypto-techniques can be assumed secure.

This section introduces the Sun Tzu Survival Problem with two games. Both games have a focus on individual untraceability. The first game considers a simple passive adversary; whereas the second game considers an active adaptive adversary who can compromise a coin holder, and can learn the real payer of a chosen transaction. These two games allow us to show the different concerns of individual untraceability.

For the simplicity, we assume that public information of coins will not reveal extra information (e.g., we assume that the coin age will not leak useful information), each transaction only contains one real input and one output, and all anonymity sets are of the same size. Later, in the next section, we will show the result (Theorem 4) when the anonymity sets are of different sizes; and in Section 6 we take the public information of coins and the possibility of having multiple inputs and outputs into consideration.

4.1. Individual untraceability (Static model)

We imagine that Sun Tzu and his army comprising N soldiers have been captured in a war. The enemy general respects Sun Tzu as a wise man [15], and is willing to free him but not his soldiers. However, Sun Tzu refuses to leave without his men. At the end, Sun Tzu and the enemy general both agree to play a game to decide whether or not to free Sun Tzu's soldiers, as follows.

(N, ℓ)-Sun-Tzu Game-1 The general prepares N identical wooden tags, and each tag is engraved with a unique number from 1 to N. The N tags are randomly distributed to the N soldiers with the number covered. So, no one knows which tag number is given to which soldier, apart from the soldier itself. Soldiers are not allowed to communicate with each other, at any time during the game.

A paper is given to the soldiers. Each soldier, say the $i-$th soldier, needs to write down a set S_i of ℓ distinct numbers ranging from 1 to N on this paper, such that the set contains the number he got on the tag along with other $\ell-1$ decoy numbers. All soldiers write down their sets according to a strategy premade by Sun Tzu. In particular, Sun Tzu needs to create a strategy, and distributes the strategy to every soldier before s/he receives a wooden tag, and Sun Tzu cannot make any communication with any soldier after the strategy is distributed to the soldiers.

At the end, the paper is presented to the general. For each soldier (say the $i-$th soldier), the general makes a unique guess on which number in the set S_i is engraved on the soldier's tag. After all guesses are made, the general verifies its guesses against the distributed tags. If a guess is correct, then the according solider will be executed. Otherwise, this solider will be freed.

Insights: In the above game, the N wooden tags present the available coins in the blockchain. Each written set is considered an anonymity set, a real tag number is a real input coin address, and decoy numbers are the added noises. *This game simulates a passive adversary that sees the entire blockchain transactions, and tries to de-anonymise transactions.* The only available information to the adversary is the list of transactions on the blockchain. It is clear that each number can be a real input only once in our game. This simulates the fact that a coin in a blockchain payment system can only be spent once. In addition, in this game, a soldier models a payer, and the Sun Tzu's distributed strategy models the system's predefined mix-in selection rule, such as *selecting all decoy numbers at random* or *do whatever your wish*, that is hard coded in the software client in practice. A payer spending different coins

is modelled by having multiple soldiers in the system. We now show insights on how different strategies would affect the individual transaction untraceability.

We define the guess *difficulty* as the maximum number of guesses the general needs to try before it can make a correct guess. We call the difficulty that the general makes a correct guess on the chosen solider the *individual untraceability*. Ideally, for any soldier, its individual untraceability is $\ell - 1$, i.e. it is executed only with probability no more than $\frac{1}{\ell}$.

We use the following example to show why the strategy matters to Sun Tzu, and how it can affect the individual untraceability.

Example 1. *Let $N = 4$ and $\ell = 2$. Three strategies are proposed to Sun Tzu, where the four soldiers write down their sets $S_1, S_2, S_3,$ and S_4 respectively as follows:*

Strategy A: {1,2}, {2,3}, {2,3}, {1,4}
Strategy B: {1,2}, {3,4}, {1,3}, {2,4}
Strategy C: {1,2}, {1,2}, {3,4}, {3,4}

Let's first consider Strategy A. Since each coin can only be spent once, and both S_2 and S_3 use the same anonymity set {2,3}, the general will be certain that "2" and "3" must be the real tag of S_2 and S_3, but the general will not know whether "2" is the real tag of S_2 or of S_3. So, both S_2 and S_3 achieve optimal individual untraceability in this game, as the difficulty of making a single correct guess on them is 1.

However, since "2"can only be used as a real tag once, and it must be a real tag of either S_2 or S_3, the general will conclude that the real tag of S_1 must be "1". Similarly, "4" must be the real tag of S_4 since "1" is the real tag of S_1. So, the difficulty of making correct guess on S_1 and S_4 is 0.

It is not difficult to see that the difficulty of making a correct guess on any given set in Strategy B and in Strategy C is 1, so all their sets achieve optimal individual untraceability in this game, whereas Strategy A does not. Thus, in this example, Sun Tzu should choose either Strategy B or Strategy C.

4.2. Individual untraceability (Adaptive model)

Game-1 considers an adversary with no extra information except the list of masked transactions recorded in the public blockchain. In practice, however, an adversary may have access to other private information. For example, the adversary may be the payer or payee of a transaction, and this gives extra knowledge to the adversary. We propose Game-2 to simulate a strong adaptive adversary for individual untraceability. In particular, the adversary is able to compromise a coin holder, and to learn the real payer of a chosen transaction.

(N, ℓ)**-Sun-Tzu Game-2** We consider a setting similar to Game-1 with the following two changes.

First, some soldiers may be traitors, and a traitor will secretly tell the general which number tag he has obtained.

Second, the general is allowed to practice before choosing a target written set S_t to make his guess. More precisely, the general is allowed to ask Sun Tzu to reveal the real number of any written set. The general can practice as many times as he wants. However, the chosen target S_t must have not be revealed before and is not created by a traitor.

At the end, the general makes a guess as in Game-1. This game simulates a strong adversary that not only sees the entire blockchain transactions, but also with the following two oracles empowered respectively by the two changes. The first oracle is a *corruption oracle* that allows an adversary to learn the corresponding secret of a public identity. In practice, this can be any coin owner. The second oracle is a *de-anonymization oracle*, which allows an adversary to learn the real input of a target transaction. In practice, this may be done through side channel information.

Insights: Let k_1 be the number of traitors that is contained in the target written set S_t, and k_2 the number of soldiers that is contained in S_t and has been revealed by Sun Tzu during the general's practice. Ideally, for any soldier, its individual untraceability is $\ell - k_1 - k_2 - 1$.

We use the example below to show why the strategy matters to Sun Tzu, and how it can affect the individual untraceability.

Example 2. *Considering the last two strategies in the Example 1:*

Strategy B: {1,2}, {3,4}, {1,3}, {2,4}
Strategy C: {1,2}, {1,2}, {3,4}, {3,4}

Although both strategies provide optimal individual untraceability in the Game-1, the guarantees provided by them in the Game-2 are different.

With Strategy B, the reveal or corruption of any single solider will identify the real input of all other sets, whereas it only exposes the real input of another one set in Strategy C. For example, if the general learns that the real tag of S_1 is "1" through the de-anonymization oracle, then with Strategy B the general can derive the identity of all other written sets: the real tag of S_3 is "3", of S_2 is "4", and of S_4 is "2". Thus, the hardness for the general is now 0 for any of the rest written sets, rather than 0 for S_2, and 1 for S_3 and S_4 as in the ideal case. In other words, for any target S_t, the general can always win with certainty. With Strategy C, the general can only derive the real tag of S_2, but has no clue about S_3 and S_4. So, it achieves the desired hardness for optimal individual untraceability, i.e. 0,1,1 for S_2, S_3, S_4, respectively. In other words, the attacker has no advantage on S_3 and S_4.

The analysis on individual untraceability can be very difficult. Considering the following example, where the size of anonymity set is 3, and the total number of soldiers is 15.

Example 3. *Let $N = 15$ and $\ell = 3$. Two strategies are proposed to Sun Tzu, as presented in table 1.*

It is easy to see that Strategy A does not offer optimal individual untraceability even in the Game-1, as anyone can observe that "1" must be the real input of S_1 (due to the fact that "2", "3", and "4" must be the inputs of S_2, S_3,

TABLE 1: Strategies of Example 3

Strategy A.

S_1	S_2	S_3	S_4	S_5
{1,3,4}	{2,3,4}	{2,3,4}	{2,3,4}	{4,5,6}
S_6	S_7	S_8	S_9	S_{10}
{4,5,6}	{7,9,10}	{8,9,10}	{8,9,10}	{8,9,10}
S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
{10,11,12}	{10,11,12}	{13,14,15}	{1,7,15}	{1,7,14}

Strategy B.

S_1	S_2	S_3	S_4	S_5
{1,2,3}	{1,4,5}	{1,4,6}	{1,4,7}	{5,6,7}
S_6	S_7	S_8	S_9	S_{10}
{2,8,9}	{2,8,10}	{2,8,11}	{9,10,11}	{10,11,13}
S_{11}	S_{12}	S_{13}	S_{14}	S_{15}
{3,12,13}	{3,12,14}	{3,12,15}	{13,14,15}	{5,14,15}

and S_4). Similarly, the general can also conclude that "7" must be the input of S_7, and "15", "14" and "13" must be the input of S_{14}, S_{15}, and S_{13}, respectively.

For Strategy B, from only observing all written sets does not give any certainty of transaction real inputs to the general in Game-1. In Game-2, since the general is able to adaptively choose both its queries during the practice and the target written set after practice, the guarantee is much more difficult to analyse. For example, the general may start by asking Sun Tzu to reveal the real input of S_1. If "1" is the real input, then the general could choose to compromise number "4" through the corruption oracle, i.e. the solider with tag number "4" is a traitor. Since "4", "5", "6", and "7" must be inputs of S_2, S_3, S_4, and S_5 (in some order), we known that "4" must be an input of either S_2, S_3, or S_4. So, it will allow the general to make a correct guess on S_5 with 100% probability, even though none of the numbers in S_5 has been compromised through the corruption oracle or de-anonymization oracle. For example, if "4" is the real input of S_2, then we know that "6" and "7" must be the real input of S_3 and S_4, respectively. Thus, "5" must be the real input of $S_5 = \{5, 6, 7\}$, even though only "1" and "4" are revealed through the two oracles. In this special case, the individual untraceability of "5" is 0 rather than the desired optimal untraceability, which is 2. However, if "1" is not the real input of S_1, then the individual untraceability of "5" may not be 0. It is very difficult to analyse all cases in order to obtain the correct individual untraceability of "5", as it typically requires considering all possible cases. In blockchains, this might be impossible to analyse as there are millions (or even billions) of transactions.

In the above example, the general first makes a query about S_1 to the de-anonymization oracle, then a query about tag with number "4" to the corruption oracle, and finally choose S_5 as the target. Now, to show that the attacker can adaptively choose the queries and the target, considering the following example.

The general still starts by asking Sun Tzu to reveal the real input of S_1. However, this time "2" is the real input. Similar to the above example, the general is able to observe that "8", "9", "10", and "11" must be the inputs of S_6, S_7, S_8, and S_9, in some order; and "8" must be the real input of either S_6, S_7, or S_8. Now, the general requests to compromise number "8" through the corruption oracle. Similar to the above example, no matter the real input of which of the three sets is "8", the general can always make a correct guess on S_9 and S_{10}. For example, if "8" is the real input of S_7, then the general can conclude that "9" and "11" are the real inputs of S_6 and S_8. Thus, "10" and "13" must be the real input of S_9 and S_{10}, even though no number in S_9 and S_{10} have been directly revealed by using the two oracles. In this example, the attacker adaptively chooses its second query and its target based on the answer of the first query.

In fact, we show in the next section that analysing the hardness of untraceability of a given strategy is very difficult – it is a $\#P$-complete problem. Fortunately, we are able to obtain some meaningful results through our new model. In particular, we show that it is possible to calculate the least upper bound of the hardness, and we provide ways to achieve optimal individual untraceability.

5. Towards Global Untraceability

To solve the challenges in the Sun Tzu survival problem, this section considers a new untraceability property, which we call global untraceability. This property serves as helper to assist us analysing individual untraceability.

We first provide a game for global untraceability, and then prove that given a strategy that is not ideal, it is very hard ($\#P$-complete) to compute the global untraceability (or the guess difficulty) of this system. Thus, it it difficult to compare the global untraceability of two different unideal systems.

Fortunately, we are able to provide meaningful results — we provide least upper bound of the global untraceability in this game. Moreover, we provide an ideal strategy, i.e. it achieves the least upper bound of the global untraceability.

Furthermore, we show that the ideal strategy in this game also achieves optimal individual untraceability in the active adaptive model (Game-2), even though the game only considers a passive adversary. Thus, the global untraceability property provides a stronger notion and is a useful helper property for analysing and understanding transaction untraceability in the blockchain.

5.1. Sun-Tzu survival problem with global untraceability

(N, ℓ)-Sun-Tzu Game-3 We consider a setting similar to Game-1. Now, when the general is given the paper, he makes a guess on each of the written set, and marks all the guesses on the paper. If all guesses are correct, then all the soldiers will be executed. Otherwise, they will be freed.

To maximise the possibility of saving his soldiers, Sun Tzu needs to create a smart strategy for the soldiers to choose decoy numbers, such that the damage caused by each correct guess is reduced to minimal.

TABLE 2: Possible combinations of real tags.

Strategy A.

Written set	S1	S2	S3	S4
Selected tags	{1,2}	{2,3}	{2,3}	{1,4}
Combination 1	1	2	3	4
Combination 2	1	3	2	4

Strategy B.

Written set	S1	S2	S3	S4
Selected tags	{1,2}	{3,4}	{1,3}	{2,4}
Combination 1	1	4	3	2
Combination 2	2	3	1	4

Strategy C.

Written set	S1	S2	S3	S4
Selected tags	{1,2}	{1,2}	{3,4}	{3,4}
Combination 1	1	2	3	4
Combination 2	1	2	4	3
Combination 3	2	1	3	4
Combination 4	2	1	4	3

Understanding the Sun Tzu survival problem and global untraceability.

We call the difficulty that the general makes correct guesses on all written sets the **global untraceability**. Given a sequence S of the written sets $\{S_1, \ldots, S_N\}$, let N_{combo} be the number of all possible combinations of real tag guessing. So, for a given S, the probability that all the general's guesses are correct is $\frac{1}{N_{combo}}$. In other words, the larger the N_{combo} is, the more difficult for the general to make correct guesses on all written sets. So, to have an ideal strategy, Sun Tzu needs to solve the following challenges:

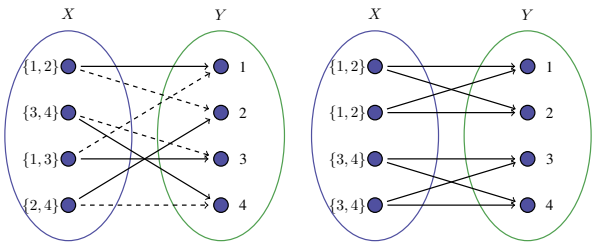
- **Challenge 1:** Since soldiers may propose strategies for Sun Tzu to consider, Sun Tzu needs to compute the N_{combo} of the proposed strategies to select the best one among all the proposals;
- **Challenge 2**: Finding the least upper bound of N_{combo} for all possible strategies. We use N_{combo}^{max} to denote this upper bound;
- **Challenge 3**: In the case that no proposed strategy is optimal, Sun Tzu needs to find a strategy to achieve the maximum number N_{combo}^{max} of possible real tag combinations.

It is easy to see that $\ell - 1$ and $N_{combo}^{max} - 1$ represent the upper bounds of individual untraceability and global untraceability of all possible strategies. In other words, they are the guess difficulties in the ideal case. Loosely speaking, *Challenge 1* aims at allowing Sun Tzu to compare all possible strategies, *Challenge 2* seeks the upper bound of the global untraceability, and *Challenge 3* is looking for a strategy that achieves the best global untraceability.

We use the example below to demonstrate the idea of global untraceability.

Example 4. *We consider the three strategies in Example 1 again:*

Strategy A: {1,2}, {2,3}, {2,3}, {1,4}
Strategy B: {1,2}, {3,4}, {1,3}, {2,4}
Strategy C: {1,2}, {1,2}, {3,4}, {3,4}



(a) Graph of Strategy B. (b) Graph of Strategy C.

Figure 4: Graphs representing the selections in the defined game. All two possible perfect matcchings of Strategy B's graph are presented by using solid lines (as one perfect matching) and dashed lines (as the other).

As demonstrated previously, the Strategy C offers a better individual untraceability in Game-2. We now consider the global untraceability. As shown in Table 2, N_{combo} of Strategy A, B, and C is 2, 2, and 4, respectively. That is, there are 2 different possible combinations of real tags in Strategy A and B, and 4 possible combinations in Strategy C. So, the global untraceability is 1 for both Strategy A and B, and 3 for Strategy C. In other words, the general can execute all soldiers with at most 2 guesses in the first two strategies, and at most 4 guesses in the last strategy. This shows that different strategies may give very different guarantees.

As we can see, even though the global untraceability of both Strategy A and B is the same, their individual untraceability is different. This seems to suggest that for a given strategy, both individual untraceability and global untraceability should be considered. However, as we will show later (in Lemma 2), optimal global untraceability implies optimal individual untraceability.

5.2. Linking the problem to graph theory

To solve the challenges in the Sun Tzu Survival Problem, we first present the relations between all written sets (each of size ℓ) and the N tags into a bipartite graph $G = (X, Y, E)$, such that $|X| = |Y| = N$, $|E| = N \cdot \ell$, and every vertex in X has the same degree ℓ. In particular, the left partitioned set X represents all written sets $\{S_1, \ldots, S_N\}$, the right partitioned set Y represents the N wooden tags, and the edge from a vertex in X to a vertex in Y represents a possible real tag guess. Since each vertex in X is a selection of size ℓ, each vertex in X has a degree ℓ, which is the number of possible real tag guesses in the correspond written set. We call such graph G the graph of (N, ℓ)-Sun-Tzu game, defined as follows.

Definition 1. *$G = (X, Y, E)$ is a graph of (N, ℓ)-Sun-Tzu game if $X = \{S_1, S_2, \ldots, S_N\}$, $Y = \{1, 2, \ldots, N\}$, $|E| = \ell \cdot N$, and the degree of each $S_i \in X$ is ℓ.*

Figure 4 shows two example bipartite graph G by using the Strategy B and C. In particular, Figure 4a shows the

graph with selection sequence $[\{1,2\},\{3,4\},\{1,3\},\{2,4\}]$, and Figure 4b shows the graph with selection sequence $[\{1,2\},\{1,2\},\{3,4\},\{3,4\}]$.

A matching of G is a set of edges from X to Y such that no two edges in the set are adjacent in G. A matching of G is *perfect* if every vertex of G is incident to exactly one edge of the matching. So, each perfect matching represents a possible sequence of real tag guesses on all written sets; and the number of possible perfect matchings is equal to the number of possible sequences of the real tag guesses. Hence, the upper bound of the perfect matching represents the maximum number of possible sequences of real tag guesses.

Example 5. *As shown in Figure 4a, the two possible perfect matchings represent all two possible guessing combinations, as presented in the Table 2. In particular, there are only two possible perfect matchings, one is represented by the solid lines and the other is represented by the dashed lines. The former represents the guessing combination 1, and the latter represents the guessing combination 2.*

This leads us to the following axiom.

Axiom 1. *Finding N_{combo} as stated in the* Challenge 1*, is equivalent to calculating the number of perfect matchings in all possible bipartite graph G of (N,ℓ)-Sun-Tzu game.*

It has been proven that, for a given bipartite graph G, the number of perfect matchings in G is equal to the permanent of its incidence matrix [5]. More precisely, given a graph $G=(X,Y,E)$, we have that its *incidence matrix* is a (0,1)-matrix $A=(a_{xy})_{(x,y)\in X\times Y}$, such that $a_{xy}=1$ if xy is an edge, and $a_{xy}=0$ otherwise. The permanent of A is defined as

$$per(A)=\sum_{\sigma\in S_N}\prod_{i=1}^{N}a_{i\sigma(i)}$$

where S_N is the symmetric group, i.e. S_N denotes the set of all permutations of N elements.

Example 6. *Graph of Strategy B (Figure 4a)'s incidence matrix is*

$$A=\begin{bmatrix}1&1&0&0\\0&0&1&1\\1&0&1&0\\0&1&0&1\end{bmatrix}$$

The permanent of A is 2, as calculated in Appendix A. Thus, the number of all possible perfect matching is 2.

We have the following axiom.

Axiom 2. *The permanent of an incidence matrix of a bipartite graph G is the number of perfect matchings in G.*

The term 'permanent' was introduced by Sir Thomas Muir in 1882 [10]. As proven in Valiant's seminal paper, computing the permanent of a $N\times N$ (0,1)-matrices is $\#P$-complete, which is believed to be even more difficult than solving NP problems [16].

Theorem 1. *(Valiant's theorem) The complexity of computing the permanent of $N\times N$ (0,1)-matrices is $\#P$-complete.*

So, unfortunately, calculating the global untraceability is $\#P$-complete. Thus, **the Challenge 1 is infeasible to solve.**

However, in 1963 Minc [8] conjectured the upper bounds for permanents of an $N\times N$ (0,1)-matrix, and this conjecture was later proved by Lev Bregman in 1973 [2]. (A simpler proof was provided by Schrijver in 1978 [13].) We present the Bregman's theorem (a.k.a. Bregman-Minc inequality) [2], as follows.

Theorem 2. *(Bregman's theorem.) Let A be an $N\times N$ (0, 1) matrix, i.e. $A=[a_{ij}]_{i,j=1}^{N}\in\{0,1\}^{N\times N}$. If for all $i\in[1,N]$, A has non-zero row sums $r_i=\sum_{j=1}^{N}a_{ij}$, then the permanent of A satisfies the inequality*

$$per(A)\le\prod_{i=1}^{N}(r_i!)^{\frac{1}{r_i}}$$

where equality holds if and only if up to permutation of rows and columns A is a block diagonal matrix in which each block is a square all-1 matrix.

From Axiom 1 and Axiom 2, we have that the *Challenge 2* is equivalent to finding the upper bounds of the permanent of G's incidence matrix. In other words, given a graph $G=(X,Y,E)$, the *Challenge 2* is to calculate the upper bound of permanent of $A=(a_{xy})_{(x,y)\in X\times Y}$, such that $a_{xy}=1$ if xy is an edge, and $a_{xy}=0$ otherwise.

Lemma 1. *For all possible graph $G=(X,Y,E)$ of (N,ℓ)-Sun-Tzu game, given its incidence matrix $A=(a_{xy})_{(x,y)\in X\times Y}$, we have that*

$$\forall x\in X, r_x=\sum_{y=1}^{n}a_{xy}=\ell$$

Proof. This lemma can be trivially proved due to the fact that for all $x\in X$, its degree is ℓ. So, each vertex in X is an endpoint of ℓ edges. Thus, the row sum $r_x=\ell$ for all possible selection x. □

Theorem 3. *For any (N,ℓ)-Sun-Tzu game, the maximum number N_{combo}^{max} of sequences of the real tag guesses is*

$$N_{combo}=(\ell!)^{\frac{N}{\ell}}$$

Proof. Let $G=(X,Y,E)$ be a graph of (N,ℓ)-Sun-Tzu game, and A the incidence matrix of G. Due to Axiom 1 and Axiom 2, we have that $N_{combo}^{max}=per(A)$.

Due to Lemma 1, we have that for all possible G, the row sum of each row in G's incidence matrix is ℓ. Due to Theorem 2, we have

$$N_{combo}^{max}=per(A)=\prod_{i=1}^{N}(\ell!)^{\frac{1}{\ell}}=(\ell!)^{\frac{N}{\ell}}$$

□

Similarly, we have the following theorem when we allow each soldier to write down a set S_i with different size $\boldsymbol{\ell_i}$.

Theorem 4. *For any (N, ℓ)-Sun-Tzu game with written sets S_i of size ℓ_i, the maximum number N_{combo}^{max} of sequences of the real tag guesses is*

$$N_{combo}^{max} = \prod_{i=1}^{N} (\ell_i!)^{\frac{1}{\ell_i}}$$

Thus, we have that for an (N, ℓ)-Sun-Tzu game, such that $N/\ell = a$ for some integer a and all sets of the same size ℓ, **the solution to Challenge 2 is $(\ell!)^{\frac{N}{\ell}}$**. More generally, if each written set S_i is of size ℓ_i, then the least upper bound is $\prod_{i=1}^{N} (\ell_i!)^{\frac{1}{\ell_i}}$.

Lemma 2. *In (N, ℓ)-Sun-Tzu game, if a strategy achieves optimal global untraceability of N_{combo}^{max}, then it also achieves optimal individual untraceability $\ell - 1$.*

Proof. As stated in the Theorem 2, the upper bound N_{combo}^{max} of global untraceability is achieved if and only if up to permutation of rows and columns, A is a block diagonal matrix in which each block is a square all-1 matrix. If an element a_{xy} of A is "1", then we know that there is an edge from $x \in X$ to $y \in Y$ in the associated graph G, i.e, y is 1-out-of-ℓ tags selected in x. Since each vertex in X is of degree ℓ, each vertex in X is an end point of ℓ edges. So, the size of each square all-1 matrix (i.e. the size of each block in A) is ℓ. In addition, since A is a block diagonal matrix, for any two blocks, they don't share the same edge of the graph. So, the individual untraceability of each block representing a written set is $\ell - 1$, which is optimal. □

Lemma 3. *For all possible graph $G = (X, Y, E)$ of (N, ℓ)-Sun-Tzu game, given any incidence matrix $A = (a_{xy})_{(x,y)\in X\times Y}$, it is efficiently verifiable that whether the strategy represented by A achieves the least upper bound of global untraceability.*

Proof. This can be proved easily by using The Bregman's theorem, i.e., the strategy represented by A achieves the least upper bound of global untraceability, if and only if up to permutation of rows and columns A is a block diagonal matrix in which each block is a square all-1 matrix. □

Lemma 4. *For all possible graph $G = (X, Y, E)$ of (N, ℓ)-Sun-Tzu game, given any incidence matrix $A = (a_{xy})_{(x,y)\in X\times Y}$, it is efficiently verifiable that whether the strategy represented by A achieves the least upper bound of both global untraceability and individual untraceability.*

Proof. This is a trivial result based on Lemma 2 and Lemma 3. According to Lemma 2, a solution achieving optimal global untraceability (in a static model) also achieves optimal individual untraceability (in an adaptive model). Thus, if a strategy achieves the least upper bound of global untraceability, it also achieves individual untraceability. □

Remark 1. *Intuitively, the individual untraceability provides more capabilities to an attacker, namely the ability to compromise a transaction and the ability to practice and to adaptively choose a target. Thus, we conjecture that individual untraceability and global untraceability implies each other (i.e., relation B of Figure 5).*

If this conjecture is correct, then thanks to Theorem 2 and Lemma 4, we provide ways to efficiently and automatically verify whether a given ledger achieves optimal untraceability, both for each individual transaction and for the entire ledger.

If it turns out that our conjecture is incorrect, i.e., the relation A is correct, then our above statement still holds, and we even additionally provide global untraceability as a stronger model to the study in the transaction untraceability.

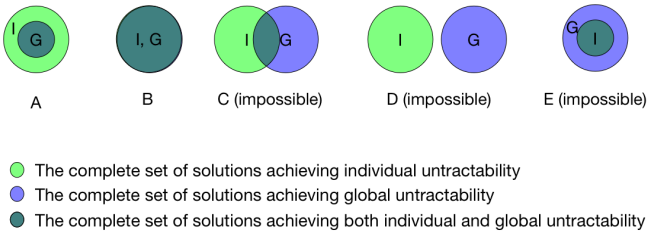


Figure 5: Possible relations between the set I and set G. Set I and G represent all systems achieving optimal individual untraceability and optimal global untraceability, respectively. Relation C,D,E are not possible due to our Lemma 2, which shows that a system achieving optimal global untraceability also achieves optimal individual untraceability.

The solution to Challenge 3, fortunately, can be obtained based on the results from Challenge 2. For an (N, ℓ)-Sun-Tzu game, if Sun Tzu picks N/ℓ distinct sets of ℓ tags as N/ℓ distinct selections, and repeat each of them ℓ times such that each time a different tag in the selection is marked as a real selection, then the incidence matrix of the resulting graph G is a block diagonal matrix in which each block is a square all-1 matrix. According to Theorem 2 and Theorem 3, this achieves N_{combo}^{max}.

From this, we know that for any anonymity set in a cryptocurrency system, all elements (i.e. the real input and decoy inputs) of this set should not be contained in other sets. In this way, the set of all anonymity sets would result in a matrix, such that up to permutation of its rows and columns, the matrix is a block diagonal matrix in which each block is a square all-1 matrix. Thus, it guarantees the upper bound of N_{combo}, and provides the optimal global untraceability. As stated previously in Lemma 2, this strategy, in fact, also achieves the optimal individual untraceability. Loosely speaking, in the above strategy, since all distinct anonymity sets do not contain the same element, any anonymity set will not leak information about a different anonymity set. So, the difficulty of making correct guesses on a anonymity set of size ℓ is always $\ell - 1$, regardless how many correct guesses one has made on other anonymity sets.

6. Application to Monero

This section applies our theoretical results to CryptoNote style protocols. In particular, we choose Monero as a con-

crete example to simplify our presentation. We first review existing attacks, then propose our suggestions on mix-in selection for Monero.

6.1. Review of existing attacks

In January 2015, a report [7] from Monero research lab shows that earlier version of Monero does not offer untraceability. In particular, three issues have been identified, namely zero-mix spending, temporal association, and common root spending.

Zero-mix spending refers to the transaction inputs that do not use any mix-in for the ring signature. When spending an unspent transaction output (UTXO) without mix-ins, anyone can trace the output of this transaction back to the input, so can break the untraceability. Recent research [9], [6] further explore the possible effects of zero-mix spending. In particular, their assessments show that zero-mix spending not only breaks the untraceability of the zero-mix transactions, but will also weaken the untraceability of other mixed spendings where the output of the zero-mix spending is used as a mix-in. The issue of zero-mix spending is fixed in the later versions ($>$ 0.9.0.) of Monero, by making at least 2 mix-ins per transaction mandatory.

Temporal association refers to the extra information leaked by the coin age. This has two effects. First, in an earlier version of Monero (version before 0.9.0), mix-ins are selected uniformly. So, older coins have been selected as mix-ins more often than younger coins. This makes older coins more damaging than the younger coins, when they are possessed by an attacker. Second, it is likely that an older (e.g. 1 year old) coin has already been spent compared to a younger (e.g. 5 days old) coin. So, younger coins in the mixed spending are most likely to be the genuine coins being spent. To mitigate the issue of temporal associations, later versions (after 0.9.0) of Monero select mix-ins according to the triangular distribution, and in the latest versions (after 0.10.1) to date, they additionally require that some of the mix-ins are selected from the "recent zone" (i.e., those coins created within the last 5 days).

Common root spending refers to transactions that spend coins obtained from a single previous transaction. For example, if two coins in the set of inputs and mix-ins of a transaction were obtained from a single previous transaction, then it is very likely that the two coins are the genuine inputs and owned by the same user. To address this issue, the report [7] recommends to generate a separate transaction for each output. However, if the outputs of this transaction are spent from a common tree of transactions, from a common block, or from within a short span of time, then they are also likely to be associated. So, the report also suggests that only one input from the payer and only one output to the payee should be used for every transaction. As a result, the payer needs to generate multiple transactions in order to complete one target transaction. However, the authors admit that this is very inefficient and expensive.

6.2. Mix-in selection for Monero

Based on our theoretical results, we present an optimal mix-in selection strategy for Monero, which guarantees optimal global untraceability and individual untraceability.

Additional considerations. We consider three main out-of-model factors in the mix-in selection, namely the information leakage from transactions, multiple inputs and outputs, and the advantage of malicious block creators.

In our model, we assume that public information of coins will not reveal extra information (e.g., we assume that the coin age will not leak useful information), and each transaction only contains one real input and one output. They lead to our first two considerations.

For information leakage, it cannot be prevented completely as demonstrated by the three identified attacks, namely zero-mix spending, temporal association, and common root spending. The zero-mix spending leaks information about a real coin input of a transaction when no mix-in is added, and this can be easily mitigated by making having mix-ins mandatory, as implemented in Monero since version 0.9.0. Temporal association refers to the extra information leaked by the coin age. Existing works suggest to either put more weight on the younger coins, or try to select mix-ins based on the real coin-spending distribution. However, the first proposal still leaks extra information to an attacker, and the second proposal is difficult to achieve as the distribution of transaction output ages varies over time, and, is depending on the economic performance of the currency [7]. We observe that, if a coin and its associated mix-ins have the same age, then the coin age will not provide any extra information to a third party. Thus, we propose to choose mix-ins from the coins with the same age, i.e. real inputs and decoy inputs are chosen from the same block. Common root spending attack refers to the extra information leaked when two to-be-spent coins are generated from the same transaction. As mentioned previously, the report [7] suggests that only one input from the payer and only one output to the payee should be used for every transaction. However, this is very inefficient and expensive in practice, as a payer needs to generate multiple transactions in order to complete one target transaction. So, to balance the efficiency and privacy, we will enforce that no two real inputs of a transaction can come from the same ring.

For multiple inputs and outputs, we choose mix-ins for each input separately and independently according to our strategy, then perform the required computation with all inputs.

The last consideration is the advantage of a malicious miner. More precisely, we concern the extra power a miner has, namely a miner is able to choose the transactions to be included in a block, and to decide their order in the block. If our strategy chooses mix-ins from the same block with a deterministic algorithm, then in order to trace a targeted transaction, a malicious miner may put his $\ell-1$ transactions around the targeted transaction in the way that they will be chosen as mix-ins of the targeted transaction according to the strategy. Although this attack is expensive, not scalable,

and its success rate depends on the mining power of the attacker, it still provides practical targeted attacks.

We aim at making the attackers job much more difficult and expensive. In particular, we choose mix-ins not only from the same block as the real input, we also choose mix-ins from neighbour blocks. We call these neighbour blocks ring neighbour blocks of the to-be-spent coin. In this way, a malicious miner has to successfully create multiple successive blocks without the other miners finding blocks between them. Assuming that the mining power rate of a malicious miner over the entire mining network is α, and the defined number of neighbour blocks is k, then the probability of successfully launching this targeted attack is α^k. In addition, the information leakage from the coin age is kept minimal as mix-ins are of a very similar age, i.e., the age difference of coins are about several minutes.

Putting these all together, we minimise the potential information leakage from transactions.

Algorithm 1 Mix-in selection

Input: The set $\mathcal{S}_\pi := \{\mathsf{coin}_{\pi,1}, \ldots, \mathsf{coin}_{\pi,m}\}$ of coins to be spent in a transaction.

Output: The set mix_k of mix-ins for each coin $\mathsf{coin}_{\pi,k}$, where $(k \in [1, m])$.

1: $k = 1$
2: $\mathsf{RingPos} = \{\}$
   # RingPos records the position of all rings.
3: **while** $k \leq m$ **do**
4: $(i, j) \leftarrow \mathsf{coin\_pos}(\mathsf{coin}_{\pi,k})$
5: $p_1 \leftarrow \lfloor j/q \rfloor$
6: $p_2 \leftarrow \lfloor i/\ell' \rfloor$
7: **if** $(p_1, p_2) \notin \mathsf{RingPos}$ **then**
8: $\mathsf{RingPos} = \mathsf{RingPos} \cup \{(p_1, p_2)\}$
   # (p_1, p_2) are used to mark the ring position.
9: **else**
10: Break with common root spending warning.
11: **end if**
12: **for all** $y \in [p_1 \cdot q + 1, (p_1 + 1) \cdot q]$ **do**
13: **for all** $x \in [p_2 \cdot \ell' + 1, (p_2 + 1) \cdot \ell']$ **do**
14: **if** x==i and y==j **then**
15: continue
16: **else**
17: $\mathsf{mix}_k = \mathsf{mix}_k \cup \{\mathsf{pos\_coin}(x, y)\}$
18: **end if**
19: **end for**
20: **end for**
21: k++
22: **end while**

Mix-in selection strategy. Let $\mathsf{coin\_pos}(\cdot)$ be the function that takes a coin as input, returns the position (i, j) of the coin indicating that the coin is the i-th coin of all coins in the j-block. Let $\mathsf{pos\_coin}(\cdot)$ be the function that takes a position (i, j) as input, outputs the coin of this position. We assume that ℓ' mix-ins are chosen from each of q successive blocks, such that $\ell = \ell' \cdot q$. In addition, we assume that each block in the blockchain contains the same number of outputs, and the number of outputs is divisible by ℓ'. In practice, this can be easily enforced on the miner side when they are creating and validating a block.

Given the set $\mathcal{S}_\pi := \{\mathsf{coin}_{\pi,1}, \ldots, \mathsf{coin}_{\pi,m}\}$ of coins to be spent, the process of selecting the set mix_k of mix-ins for each coin $\mathsf{coin}_{\pi,k}$ $(k \in [1, m])$ is presented in the Algorithm 1.

Loosely speaking, to ensure that all rings are independent to each other, instead of choosing them randomly, we choose them deterministically. We take the advantage of the transparency offered by the blockchain, and use it to coordinate the mix-in selection from all users (line 4-6, 12-20). In particular, we partition each block into distinct parts, such that the collection of ℓ' successive coins (a.k.a. transaction outputs) in every q blocks form a ring, and no coin is contained in different rings. This way of forming rings provides two advantages. First, it ensures that all rings are independent. So, this achieves the upper bound of global untraceability and individual untraceability, as proved in the previous section. Second, since all members of a ring are contained in the associated q successive blocks, they have the same or very similar age. This solves the challenges of how to match mix-in selection with coin-spending distribution, as the coin age leaks no extra information now.

To spend a coin where all ring neighbour blocks are already confirmed, the payee can make the transaction directly and does not need to wait for any extra time. However, if not all ring neighbour blocks are confirmed, then the payee may need to wait for extra time t in order to known which coin to be used as a mix-in. This only happens if the coin is located at the tail of the blockchain, i.e. the coin is contained in the latest confirmed blocks. Currently a Monero block is created every two minutes in average, so in the worst case — when the coin is in the last confirmed block of the blockchain and the coin is the first block amongst all successive ring neighbour blocks — the waiting time is 2 minutes for each ring neighbour block. Currently, in Monero 96% of transactions have at most 4 mix-ins [6], and the suggested confirmation time is 10 confirmations $\times$ 2 minutes = 20 minutes. If we assume that each real input has 4 mix-ins, and 2 of them are chosen from the same block and the other 2 are chosen from the ring neighbour blocks, then in the worst case a payee needs to wait for 4 extra minutes, which is considered acceptable.

Moreover, to prevent common-root spending attacks, as also recommended in [7], we enforce a user to spend coins only if they are not from the same ring (line 7-11).

7. Discussion

Randomised selection strategies. One may think that all mentioned strategies in our examples are fixed and do not account for randomised selection. However, in fact, the example strategies mentioned in the paper mainly serve as examples to demonstrate the possible outcome of any type of strategies. In particular, we first show that if randomized

strategies give a certain kind of outcome (such as the one in our examples), then the privacy cannot be guaranteed. Second, we show that according to our theorem, the randomized strategies will not be able to provide optimized privacy guarantee, as the overlapped sets will leak extra information.

Modeling consideration. Intuitively, it is reasonable to define the global untraceability in a similar way as defining the individual untraceability. For example, a straw man definition could be defining the global untraceability as the least number of coins required to be corrupted in order to de-anonymize any transaction. Let's call it "least-number model (LNM)" for simplicity. However, this model is actually weaker than our proposed model due to the cascade effect. Considering Example 1, where Strategy B is $\{1,2\}$, $\{3,4\}$, $\{1,3\}$, $\{2,4\}$, and Strategy C is $\{1,2\}$, $\{1,2\}$, $\{3,4\}$, $\{3,4\}$. In the LNM model, the global untraceability for both B and C is "1", i.e., the least number of coins required to be corrupted in order to de-anonymize any transaction is "1". However, the compromise of a single coin in B will de-anonymize all transactions, but this is not the case in C. In contrast, our model considers the difficulty of de-anonymising all transactions, which is stronger, i.e., our model considers the cascade effect, and the ideal global untraceability in our model implies the idea global untraceability in the LNM.

Model limitation. As the focus of our model is on the snapshots of a CryptoNote-style blockchain, we do not consider timing issues in our model. Including timing considerations certainly deserves more care, and extending our model to cover them is an interesting future work.

Global coordinator. We do not assume any global coordinator for choosing decoy transactions. In our model, Sun Tzu, which might be misunderstood as a global coordinator, models the community who decides the blockchain protocol rules, which will be hard-coded in the client software. This is the reason that our model only allows Sun Tzu to distribute the strategies to the soldiers before they receive a wooden tag, and does not allow Sun Tzu to have any communication with the soldiers after they received their strategies. For example, if considering the rule of the maximum size of a block in BitCoin as a strategy (for a different purpose of course), then the "global coordinator" is the community who decides the hard-coded maximum block size. In practice, these rules are hard-coded in the client. For example, similar to the current version of Monero client which already enforces the minimum number of mix-ins, our scheme can be integrated into the Monero client to choose mix-ins according to the blockchain data.

Binned mixin sampling. Binned mixin sampling [9] is a leading and advanced mixin strategy that aims at preventing attacks using the knowledge of spend-time distribution. It has a design principle similar to the strategy proposed in Section 6.2. In particular, binned mixin sampling groups all outputs in a CryptoNote-style blockchain into "bins", which are sets of a fixed size. All outputs in a bin are chosen from the same block or neighboring blocks. So, outputs in the same bin have a similar age. If any of the output in a bin is used as an input in a transaction, then all other outputs should be mixed in as well. To eliminate the the advantage of a malicious miner, the assignment of outputs to bins are determined by the block header, which prevents a malicious miner to predict the assignment of outputs to bins.

However, while being able to prevent timing-related attacks, it does not achieve optimal individual untraceability. In particular, with binned mixin sampling, transactions can have arbitrary number of bins that is calculated based on the number of used mixins. Thus, it is possible that rings in different transactions have partial overlapped bins. This allows an attacker to rule out some overlapped bins, to increase its success rate of guessing a real input in a transaction.

8. Conclusion

Blockchains have become popular in the last years, and blockchain-based cryptocurrencies have a capital market of billions of dollars. This motivates attackers to break user privacy in order to learn more about their transactions. We observed new attacks to show the importance of mix-in selection strategy. The observed attacks can by applied by a passive attacker and/or an active adaptive attacker. We also proposed new games to model the different transaction untraceability properties, and modeled them by using bipartite graphs. Our models allowed us to provide several important results, such as the least upper bound of global untraceability, and the relations between global untraceability and individual untraceability. We further provided ways to evaluate whether a given ledger achieves optimal global untraceability, and strategies to achieve optimal global untraceability and individual untraceability.

References

[1] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[2] L. M. Bregman, "Some properties of nonnegative matrices and their permanents," in *Soviet Math. Dokl*, vol. 14, no. 4, 1973, pp. 945–949.

[3] "Cryptocurrency market capitalizations," retrieved 2017-07-01. [Online]. Available: http://coinmarketcap.com/

[4] "Cryptonote currencies," 2018. [Online]. Available: https://cryptonote.org/coins/

[5] D. C. Kozen, *The Design and Analysis of Algorithms*. New York, NY, USA: Springer-Verlag New York, Inc., 1992.

[6] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of monero's blockchain," in *ESORICS*, 2017, pp. 153–173.

[7] A. Mackenzie, S. Noether, and M. C. Team, "Improving obfuscation in the cryptonote protocol," Monero research lab report MRL-0004, 2015. [Online]. Available: https://lab.getmonero.org/pubs/MRL-0004.pdf

[8] H. Minc, "Upper bounds for permanents of (0, 1)-matrices," *Bulletin of the American Mathematical Society*, vol. 69, no. 6, pp. 789–791, 1963.

[9] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, and N. Christin, "An empirical analysis of traceability in the monero blockchain," *PoPETs*, vol. 2018, no. 3, pp. 143–163, 2018.

[10] T. Muir, "On a class of permanent symmetric functions," *Proceedings of the Royal Society of Edinburgh*, vol. 11, pp. 409–418, 1882.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2009.

[12] S. Noether, "Ring signature confidential transactions for monero," Cryptology ePrint Archive, Report 2015/1098, 2015, http://eprint.iacr.org/2015/1098.

[13] A. Schrijver, "A short proof of minc's conjecture," *Journal of combinatorial theory, Series A*, vol. 25, no. 1, pp. 80–83, 1978.

[14] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero," ESORICS, 2017.

[15] Sun Tzu, *The Art of War*, 771 BC to 476 BC.

[16] L. G. Valiant, "The complexity of computing the permanent," *Theoretical computer science*, vol. 8, no. 2, pp. 189–201, 1979.

[17] N. van Saberhagen, "Cryptonote v 1.0," 2012. [Online]. Available: https://cryptonote.org/whitepaper_v1.pdf

[18] Z. Yu, M. H. Au, J. Yu, R. Yang, Q. Xu, and W. F. Lau, "New empirical traceability analysis of cryptonote-style blockchains," in *Financial Cryptography and Data Security (FC)*, 2019.

Appendix

A, as below, is the incidence matrix (given below) of Graph of Strategy B (Figure 4a). The permenent of A is calculated as follows.

$$A = (a_{xy})_{(x,y)\in 4\times 4} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} Per(A) = & \sum_{\sigma\in S_N}\prod_{i=1}^{N} a_{i\sigma(i)} \\ = & \; a_{11}\cdot a_{22}\cdot a_{33}\cdot a_{44} + a_{12}\cdot a_{23}\cdot a_{34}\cdot a_{41} \\ & + a_{13}\cdot a_{24}\cdot a_{31}\cdot a_{42} + a_{14}\cdot a_{21}\cdot a_{32}\cdot a_{43} \\ & + a_{11}\cdot a_{23}\cdot a_{32}\cdot a_{44} + a_{12}\cdot a_{24}\cdot a_{31}\cdot a_{43} \\ & + a_{13}\cdot a_{22}\cdot a_{34}\cdot a_{41} + a_{14}\cdot a_{21}\cdot a_{33}\cdot a_{42} \\ & + a_{11}\cdot a_{24}\cdot a_{33}\cdot a_{42} + a_{12}\cdot a_{21}\cdot a_{34}\cdot a_{43} \\ & + a_{13}\cdot a_{22}\cdot a_{31}\cdot a_{44} + a_{14}\cdot a_{23}\cdot a_{32}\cdot a_{41} \\ & + a_{11}\cdot a_{22}\cdot a_{34}\cdot a_{43} + a_{12}\cdot a_{24}\cdot a_{33}\cdot a_{41} \\ & + a_{13}\cdot a_{21}\cdot a_{32}\cdot a_{44} + a_{14}\cdot a_{23}\cdot a_{31}\cdot a_{42} \\ & + a_{11}\cdot a_{23}\cdot a_{34}\cdot a_{42} + a_{12}\cdot a_{21}\cdot a_{33}\cdot a_{44} \\ & + a_{13}\cdot a_{24}\cdot a_{32}\cdot a_{41} + a_{14}\cdot a_{22}\cdot a_{31}\cdot a_{43} \\ & + a_{11}\cdot a_{24}\cdot a_{32}\cdot a_{43} + a_{12}\cdot a_{23}\cdot a_{31}\cdot a_{44} \\ & + a_{13}\cdot a_{21}\cdot a_{34}\cdot a_{42} + a_{14}\cdot a_{22}\cdot a_{33}\cdot a_{41} \\ = & \; 2 \end{aligned}$$