Mixing Transactions with Arbitrary Values on Blockchains

Wangze Ni †, Peng Cheng *, Lei Chen †,
†The Hong Kong University of Science and Technology, Hong Kong, China
{wniab, leichen}@cse.ust.hk
*East China Normal University, Shanghai, China
pcheng@sei.ecnu.edu.cn

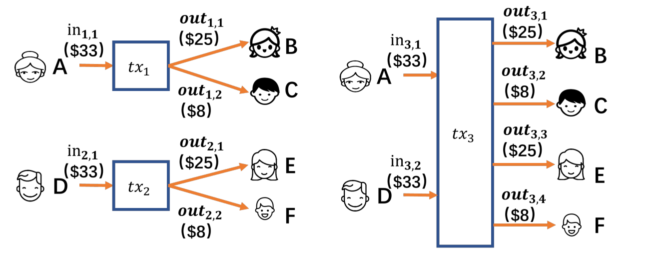
***Abstract*—Due to the transparency of blockchain, adversaries can observe the details of a transaction, and then utilize the amount as a unique quasi-identifier to make deanonymization. Nowadays, to obscure the linkages between receivers and senders within a transaction on the blockchain, mixing services are widely applied in many real applications to enhance cryptocurrencies' anonymity. The basic idea of mixing services is to hide an output within several other outputs in a transaction such that adversaries cannot distinguish them by their amounts since they are purposely selected to have the same amount. For a set of original outputs with different amounts, mixing services need to decompose them into a set of decomposed outputs, where any decomposed output has some other decomposed outputs with the same amount. Since the transaction fee is related to the number of outputs, we are motivated to decompose original outputs into a minimal set of decomposed outputs, which is challenging to guarantee the privacy-preserving effect at the same time. In this paper, we formally define the anonymity-aware output decomposition (AA-OD) problem, which aims to find a c-decomposition with a minimum number of decomposed outputs for a given original output set. A c-decomposition guarantees that for any original output o, there are at most c of all decomposed outputs with an amount of x coming from o. We prove that the AA-OD problem is NP-hard. Thus, we propose an approximation algorithm, namely Boggart[1], to solve the AA-OD problem with a $(\frac{2}{c}+3)$-approximation bound on the number of decomposed outputs. We verify the efficiency and effectiveness of our approach through comprehensive experiments on both real and synthetic data sets.**

***Index Terms*—Blockchain, Mixing Service, Privacy**

I. Introduction

As a promising method to protect users' privacy on the blockchain, mixing services draw much attention from both academia [1]–[3] and industry [4], [5]. The basic idea of mixing services is to mix several purposely selected transactions from different users into one transaction, such that the linkages of original transactions' senders and receivers are obscured, hence transaction flows are hard to trace.

As shown in Figure 1(a), account A wants to make a transaction tx_1 to transfer \$25 to account B by an output $out_{1,1}$ and \$8 to account C by an output $out_{1,2}$. In blockchains, an output transfers some tokens to an account. If the user directly proposes the transaction tx_1 to the blockchain, due to the transparency of blockcahin, adversaries can observe tx_1 to reveal the transactional linkage between the account A and B. The information of transactional linkages between accounts can be further used to mine their private information (e.g., transaction history and social network) [6]–[8].

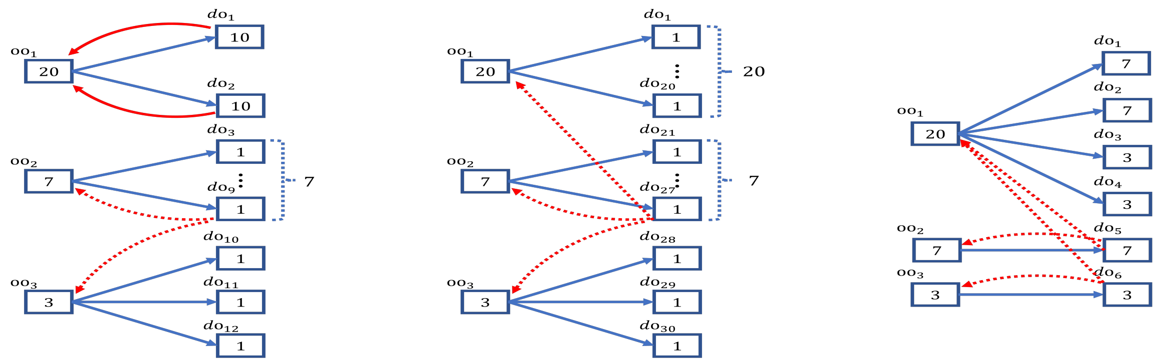


(a) Making transactions individually (b) Making transactions together
Fig. 1. An Example of Mixing Services.

To prevent this kind of attack, researchers propose some mixing methods to mix several similar transactions from different users into one transaction [1]–[5]. For example, assume the user of account D wants to make a transaction tx_2 transferring \$25 and \$8 to account E and F, respectively. With a mixing method, tx_1 and tx_2 are first mixed in tx_3 offline (as shown in Figure 1(b)), and only tx_3 is proposed to blockchain. Then, even if adversaries know that A transferred \$25 to B in tx_3, they cannot determine which pseudonym between B and E belongs to B. In other words, mixing methods can obscure the *intra-transaction linkages*, such that adversaries cannot determine the linkages between inputs and outputs within a transaction. For the *cross-transaction linkages* between transactions, researchers have proposed some methods, such as ring signatures [9], [10]. Mixing methods and ring signature methods complement each other to protect the anonymity of users in the blockchain. For the details of mixing services, please refer to Subsection II-B.

Since in practice it is rare to simultaneously have several transactions whose outputs' amounts are the same, the existing mixing methods decompose the original outputs into standard denominations, like 0.001, 0.01, 0.1, 1, and 10 [11]. To distinguish, we term the outputs before decomposition as *original outputs* and those after decomposition as *decomposed outputs*. The receiver account of each decomposed output is different, such that adversaries cannot know which decomposed outputs transfer tokens to the same receiver. However, the existing solutions have two critical shortcomings: (1) the privacy-preserving effect is not theoretically guaranteed; and (2) the

[1]Boggart is a magical creature in J. K. Rowling's Harry Potter series who can shift his shape and no one knows what it looks like.



(a) Solution 1 (b) Solution 2 (c) Solution 3

Fig. 2. The Motivation Example.

transaction fee is high. Thus, how to find a decomposition solution with a low fee satisfying users' privacy requirements is an important problem. Here is a motivation example.

Example 1. *There are three original outputs from different transactions, $oo_1 \sim oo_3$, whose amounts are 20, 7, and 3, respectively. We assume, due to their background knowledge, adversaries know the original output sets.*

The first decomposition solution is shown in Figure 2(a). Specifically, oo_1 is decomposed into two decomposed outputs, do_1 and do_2, whose amounts are both 10. Besides, oo_2 and oo_3 is decomposed into 7 and 3 decomposed outputs with an amount of 1, respectively. Note that adversaries can only observe the set of decomposed outputs on the blockchain, and they cannot observe the linkages between an original output and its decomposed outputs (as shown in blue lines). However, since the amount of an original output is equal to the summation of its decomposed outputs' amounts, adversaries can still infer some linkages. For example, as shown in red solid lines, adversaries can easily find that do_1 and do_2 are decomposed from oo_1, because the amounts of oo_2 and oo_3 are both smaller than the amounts of do_1 and do_2. Thus, this solution cannot protect users' privacy.

The second solution, as shown in Figure 2(b), decomposes each original output into decomposed outputs with an amount of 1. As shown by the read dot lines, any decomposed output may come from any original outputs. Thus, given any decomposed output, adversaries cannot accurately determine its original output. However, there are 30 decomposed outputs. The transaction fee is related to the number of decomposed outputs. For example, when the transaction fee is 10 satoshi per byte, it costs extra 340 satoshi if the number of outputs increases by one [12]. Therefore, by this solution, the privacy-preserving effect is good, but the transaction fee is high.

A good solution, as shown in Figure 2(c), is to decompose oo_1 into two decomposed outputs with an amount of 7 (i.e., do_1 and do_2) and two decomposed outputs with an amount of 3 (i.e., do_3 and do_4). The other two original outputs are directly turned into two decomposed outputs whose amounts are 7 and 3. With this solution, there are 3three decomposed outputs with an amount of 7. Since the amount of oo_1 is 20 and the amount of oo_3 is less than 7, adversaries can know that the decomposed outputs with an amount of 7 cannot all be decomposed from oo_1. In other words, they can conclude that, two of the decomposed outputs with an amount of 7 are from oo_1, and one of them is from oo_2. Furthermore, they can conclude that, two of the decomposed outputs with an amount of 3 are from oo_1, and one of them is from oo_3. However, even they can infer this information, given any decomposed output, adversaries still cannot determine its original output. For example, given do_5, adversaries only know one of oo_1 and oo_2 is its original output, but cannot determine exactly which one is its original output. Thus, by this solution, the linkages between the original outputs and the decomposed outputs can be obscured. In addition, there are only 6 decomposed outputs, which is much smaller than the size of the second solution.

Therefore, to overcome the shortcomings in existing mixing solutions, we are motivated to find a decomposition solution with a minimum number of decomposed outputs to satisfy users' anonymity requirements. Inspired by the idea of confidence bounding [13], [14], we first define a novel anonymity concept, c-decomposition, to measure the anonymity of a decomposition solution. A c-decomposition requires that in a transaction less than c of decomposed same-amount outputs are from the same original output. For example, the solution shown in Figure 2(c) is a $\frac{2}{3}$-decomposition, since $\frac{2}{3}$ of the decomposed outputs amount of seven and three are decomposed from oo_1. But the solution shown in Figure 2(a) is a 1-decomposition since all the decomposed outputs amount of ten are from oo_1. We prove that with a c-decomposition, an adversary's posterior belief that a decomposed output is decomposed from an original output can be bounded by c. Therefore, we can use the parameter c to measure the anonymity of a decomposition solution. Then, we formally define the anonymity-aware output decomposition (AA-OD) problem. Given a set of original outputs, the AA-OD problem aims to find a decomposition solution with a minimum number of decomposed outputs to satisfy the anonymity constraint. We prove that the AA-OD problem is NP-hard, therefore intractable. To solve the AA-OD problem, we propose a $(\frac{2}{c}+3)$-approximate algorithm, namely Boggart. Through the experiments over real data sets in Section VI, we illustrate that, the size of decomposition outputs obtained by Boggart can be only 10^{-8} of the size of results decomposed with denominations in the best scenarios.

To the best of our knowledge, this is the first study proposing a decomposition solution with minimal size of

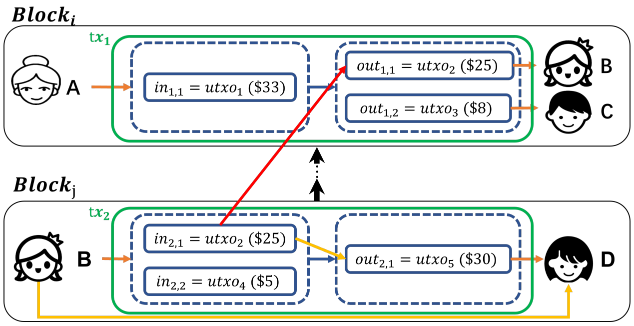


Fig. 3. An example of the UTXO blockchain.

outputs to theoretically guarantee the privacy-preserving effect in mixing services. In summary, we have made the following contributions:

- We define a novel anonymity concept, c-decomposition, formulate the anonymity-ware output decomposition (AA-OD) problem and prove its NP-hardness in Section III.
- We propose a $(\frac{2}{c}+3)$-approximate algorithm for AA-OD, namely Boggart, in Section IV
- We conduct comprehensive experiments on real and synthetic data sets to evaluate the efficiency as well as the effectiveness of our proposed solution in Section VI.

Besides, we introduce preliminaries in Section II, discuss the related work in Section VII, and conclude our work in Section VIII.

II. Preliminaries

In this section, we review some background knowledge.

A. UTXO-model Blockchain

In the UTXO model, each UTXO is an output that is generated in a previous transaction and has not been used. Each UTXO contains a positive number of tokens. An account may have multiple UTXOs. If a user wants to make a transaction, she/he needs to specify which UTXOs are used as the transaction's inputs. The sum of the inputs' amounts is equal to the sum of the outputs' amounts in the transaction. In other words, a transaction consumes some UTXOs from previous transactions and creates some new UTXOs that can be used in future transactions. Figure 3 shows an example, where $in_{i,j}$ indicates the j^{th} input in transaction tx_i, and $out_{i,j}$ indicates the j^{th} output in tx_i. In $block_i$, the user of account A generates a transaction tx_1. In tx_1, the user consumes the $33 tokens in an UTXO $utxo_1$ and generates two outputs $out_{1,1}$ and $out_{1,2}$ transferring $25 and $8 tokens to account B and account C, which are two new UTXOs, $utxo_2$ and $utxo_3$. In addition to $utxo_2$, account B has another $utxo_4$. Latter, in $block_j$, the user of account B generates a transaction tx_2, which consumes the $30 tokens in $utxo_2$ and $utxo_4$ and transfers the tokens to account D. We formally define the primary concepts of transactions as follows.

Definition 1 (A transaction)**.** A transaction, denoted by $t_i = (In_i, Out_i)$, consumes the tokens in inputs $in_{i,j}$ in In_i and transfers tokens to accounts by the outputs $out_{i,j}$ in Out_i. An input $in_{i,j}$ is denoted by $in_{i,j} = (iu_{i,j}, iv_{i,j}, ia_{i,j})$, where $iu_{i,j}$ is the UTXO consuming in $in_{i,j}$, $iv_{i,j}$ is the number of tokens in $iu_{i,j}$, $ia_{i,j}$ is the account that owns $iu_{i,j}$. An output $out_{i,j}$ is denoted by $out_{i,j} = (ou_{i,j}, ov_{i,j}, oa_{i,j})$, where $ou_{i,j}$ is the UTXO that $out_{i,j}$ generates, $ov_{i,j}$ is the number of tokens in $ou_{i,j}$, and $oa_{i,j}$ is the payee's account of $ou_{i,j}$. For each transaction t_i, the sum of tokens in In_i is equal to the sum of tokens in Out_i, i.e., $\sum\limits_{in_{i,j} \in In_i} iv_{i,j} = \sum\limits_{out_{i,j} \in Out_i} ov_{i,j}$.

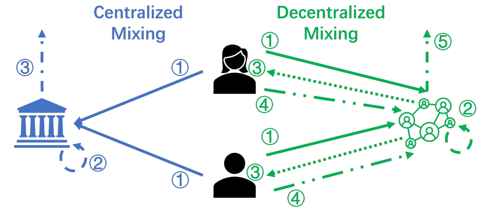


Fig. 4. Mixing Methods

For the Example illustrated in 3, $Out_1 = \{out_{1,1}, out_{1,2}\}$, $ou_{1,1} = utxo_2$, $ov_{1,1} = 25$, $oa_{1,1} = B$, $In_2 = \{in_{2,1}, in_{2,2}\}$, $iu_{2,1} = utxo_2$, $iv_{2,1} = 25$, and $ia_{2,1} = B$. The size of a transaction is related to the number of inputs and outputs, not their amounts. The size of each output is about 32 bytes. Since the transaction fee is proportional to the transaction size, when the transaction fee is 10 satoshi per byte, it costs extra 340 satoshi if the number of outputs increases by one [12].

B. Mixing Methods

Due to the transparency of blockchain, adversaries can observe the details of transactions and infer some transactional linkages between accounts. The transactional linkages can be classified as cross-transaction linkages and intra-transaction linkages. The cross-transaction linkages reveal the correlations between transactions. For example, when adversaries observe tx_1 and tx_2 on the blockchain, since $in_{2,1}$ is $utxo_2$, which is generated in tx_1, they can know the sender of tx_2 is the receiver of tx_1 (shown in red line in Figure 3). The intra-transaction linkages reveal the correlations between the senders and the receivers within a transaction. For example, when adversaries observe tx_2 on the blockchain, and they can know that the owner of account B knows the owner of D, and the tokens of $out_{2,1}$ are from $in_{2,1}$ and $in_{2,2}$ (shown in yellow line). Researchers have proposed some privacy protection methods to obscure the cross-transaction linkages, such as ring signatures [9], [10]. A ring signature hide the actually spent UTXO of a transaction's input within a set of other UTXOs. For example, suppose the user uses a ring signature $rs = \{utxo_2, utxo_6\}$ to hide $utxo_2$ in $in_{2,1}$, where $utxo_6$ is generated in tx_3. Thus, after observing tx_2, adversaries cannot tell which one of tx_1's receiver and the tx_3's receiver is the sender of tx_2. However, these methods cannot obscure the intra-linkages. Even if the user uses rs to hide $utxo_2$ in $in_{2,1}$, adversaries still can know the token of $utxo_5$ is transferred from $in_{2,1}$ and $in_{2,2}$. Mixing methods are proposed as promising methods to obscure the intra-linkages between senders and receivers within a transaction. For example, in Figure 1(b), by mixing methods, adversaries cannot tell which one of $in_{3,1}$ and $in_{3,2}$ transfers tokens to $out_{3,1}$. Mixing methods and ring signature methods complement each other to protect the anonymity of users in the blockchain.

TABLE I
POSSIBLE MATCHES OF FIGURE 2(C).

mh_i	$MO_{i,j}$	mh_i	$MO_{i,j}$	mh_i	$MO_{i,j}$
mh_1	$MO_{1,1} = \{do_1, do_2, do_3, do_4\}$	mh_2	$MO_{2,1} = \{do_1, do_2, do_3, do_6\}$	mh_3	$MO_{3,1} = \{do_1, do_2, do_4, do_6\}$
	$MO_{1,2} = \{do_5\}$		$MO_{2,2} = \{do_5\}$		$MO_{3,2} = \{do_5\}$
	$MO_{1,3} = \{do_6\}$		$MO_{2,3} = \{do_4\}$		$MO_{3,3} = \{do_3\}$
mh_4	$MO_{4,1} = \{do_1, do_5, do_3, do_4\}$	mh_5	$MO_{5,1} = \{do_1, do_5, do_3, do_6\}$	mh_6	$MO_{6,1} = \{do_1, do_5, do_4, do_6\}$
	$MO_{4,2} = \{do_2\}$		$MO_{5,2} = \{do_2\}$		$MO_{6,2} = \{do_2\}$
	$MO_{4,3} = \{do_6\}$		$MO_{5,3} = \{do_4\}$		$MO_{6,3} = \{do_3\}$
mh_7	$MO_{7,1} = \{do_2, do_5, do_3, do_4\}$	mh_8	$MO_{8,1} = \{do_2, do_5, do_3, do_6\}$	mh_9	$MO_{9,1} = \{do_2, do_5, do_4, do_6\}$
	$MO_{7,2} = \{do_1\}$		$MO_{8,2} = \{do_1\}$		$MO_{9,2} = \{do_1\}$
	$MO_{7,3} = \{do_6\}$		$MO_{8,3} = \{do_4\}$		$MO_{9,3} = \{do_3\}$

As shown in Figure 4, based on the operation mechanisms, mixing services can be classified into two classes: centralized mixing methods and decentralized mixing methods. For *centralized mixing methods* (e.g., Bitcoin Fog [15]), users first transfer their tokens to the accounts of central mixing servers (i.e., ① in blue) and ask the servers to transfer some tokens to some accounts under privacy requirements. Then, the servers will group several users' requests in a transaction and decompose the original outputs into some decomposed outputs with the same amounts satisfying the privacy requirements (i.e., ② in blue in Figure 4). Finally, they propose the transaction to the blockchain (i.e., ③ in blue). Servers will charge some mixing fees from users. For example, suppose the mixing fee is \$2, and a user wants to transfer \$20 to another user. Then, in Step ①, the user needs to transfer \$22 tokens to the server. When servers propose transactions to the blockchain, they should pay transaction fees, and the differences between mixing fees and transaction fees are their profits. Thus, to maximize their profits, they are motivated to minimize the number of decomposed outputs to save transaction fees.

For *decentralized mixing services* (e.g., Wasabi [16]), users first send messages stating their original transactions and privacy requirements to coordinators (i.e., ① in green). Then, coordinators group several similar requests in a transaction and decompose the original outputs into decomposed outputs with the same amount satisfying the privacy requirements (i.e., ② in green). Then, the mixing transaction is sent back to participants involved in the transaction for agreement (i.e., ③ in green). Then, the participants sign the transaction and send it back to coordinators again (i.e., ④ in green). If all participants sign the transaction, the mixing transaction is proposed to the blockchain (i.e., ⑤ in green). If one participant does not sign the transaction and the waiting timer is time out, all participants go back to Step ① in green and repeat the aforementioned process. Since users need to pay transaction fees when the mixing transaction is proposed to the blockchain, they are also motivated to find a solution with a minimized number of decomposed outputs to save transaction fees.

In this paper, we only consider how to decompose a set of given original outputs. Thus, our solution can be used in Step ② of both centralized and decentralized methods. Formally, we define the concept of decomposed output as follows.

Definition 2 (A decomposed output)**.** A decomposed output is denoted by $do_i = (ds_i, dv_i, dr_i)$, where ds_i is the original output where do_i is decomposed from, dv_i is its amount, and dr_i is the receiver account of dr_i.

Besides, we use oo_i to indicate an original output and use ov_i to indicate its amount. In Figure 2(c), $ds_1 = oo_1$, $dv_1 = 7$, and $ov_1 = 20$. A decomposed output do_i transfers dv_i tokens to a receiver account of dr_i. Once the mixing transaction is recorded on the blockchain, adversaries can observe dv_i and dr_i of each decomposed output, but they cannot know ds_i. In the blockchain, a user can have multiple accounts, and senders will assign decomposed outputs with different receiver accounts. Suppose the receiver account of oo_1 in Figure 2(c) is A. Meanwhile, the owner of A has another three accounts, B, C, D, and E. Then, the receiver addresses of oo_i's decomposed outputs are $dr_1 = B$, $dr_2 = C$, $dr_3 = D$, $dr_4 = E$. Furthermore, to prevent adversaries from knowing all accounts of the receiver, senders can randomly make new accounts for receivers by the Diffie-Hellman key exchange method [17], [18], and assign decomposed outputs with these new receiver accounts. Thus, by the receiver accounts, adversaries cannot find which decomposed outputs are from the same original output or are sent to the same user [19].

C. The Attack Model

In this paper, we consider an adversary model where adversaries have enough background such that they accurately know the original output set. Moreover, they know the algorithm that is used to retrieve the decomposition solution. However, they do not know the receivers' addresses of decomposed outputs from an original output. In other words, adversaries can know that the amounts of decomposed outputs from an original output, but they cannot know which particular decomposed outputs they are. For example, in Figure 2(c), adversaries can know oo_1 is decomposed into two decomposed outputs with amount of 7 and two decomposed outputs with amount of 3. However, adversaries cannot tell which two of do_1, do_2, and do_5 are decomposed from oo_1.

Given a mixing transaction and its original output set, to infer the original output of each decomposed output, adversaries first infer all possible matches between the original and decomposed output. Then, based on the matches, they update their posterior belief, which is a set of conditional probabilities that a decomposed output is from an original output when the original output set and the decomposed output set are as given. Finally, for a target decomposed output, adversaries select the original output with the highest probability as its original output.

Definition 3 (A match)**.** A match between an original output set OO and a decomposed output set DO is denoted by $mh_i(OO, DO) = \{MO_{i,1}, \cdots, MO_{i,|OO|}\}$, where $MO_{i,j}$ is the set of decomposed outputs that are considered from oo_j, $\bigcup_{oo_j \in OO} MO_{i,j} = DO$, and $\forall oo_j \in OO$, $\sum_{do_k \in MO_{i,j}} dv_k = ov_j$.

For simplicity, when OO and DO are clear, we abbreviate $mh_i(OO, DO)$ as mh_i. Given an original output set OO and a decomposed output set DO, there may be a set of possible matches, denoted by M. For Figure 2(c), $OO = \{oo_1, oo_2, oo_3\}$ and $DO = \{do_1, \cdots, do_6\}$. Table I illustrates nine possible matches between OO and DO. Then, adversaries calculate posterior belief by the retrieved matches set.

Definition 4 (Posterior belief)**.** The posterior belief of adversaries is the set of condition probabilities that a decomposed output is from an original output when the original output is OO and the decomposed output is DO, i.e., $P(oo_i, do_j|OO, DO) = \frac{N_{i,j}(M)}{|M|}$, where $|M|$ is the number of possible matches, and $N_{i,j}(M)$ is the number of matches where do_j is from oo_i.

In other words, $P(oo_i, do_j|OO, DO)$ is the ratio of the number of matches where do_j is from oo_i to the number of all possible matches between OO and DO. As shown in Table I, $N_{1,1}(M) = 6$ and $P(oo_1, do_1|OO, DO) = \frac{N_{1,1}(M)}{|M|} = \frac{2}{3}$. Besides, $P(oo_2, do_1|OO, DO) = \frac{1}{3} < P(oo_1, do_1|OO, DO)$. Thus, oo_1 is the most likely original output of do_1.

III. Problem Definition

In this section, we first propose a novel anonymity concept, namely the c-decomposition, and prove its privacy-preserving effect. Then, we formulate the anonymity-aware output decomposition (AA-OD) problem and prove its NP-hardness.

A. c-Decomposition

Denote $d_i(x)$ as the number of decomposed outputs amount of x that are decomposed from oo_i, i.e., $d_i(x) = |\{do_j | do_j \in DO, dv_j = x, ds_j = oo_i\}|$. Thus, we can calculate the conditional probability in Definition 4 by $P(oo_i, do_j|OO, DO) = \frac{d_i(dv_j)}{\sum_{oo_i \in OO} d_i(dv_j)}$. Thus, to limit $P(oo_i, do_j|OO, DO)$, we should bound the percentage of the decomposed outputs from the same original output, which fits the idea of confidence bounding [13], [14]. Thus, motivated by the idea of confidence bounding, we define a novel concept, namely c-decomposition.

Definition 5 (c-decomposition)**.** A decomposed output set DO of an original output set OO is a c-decomposition, if for any original output oo_i and any decomposed output do_j, among the decomposed outputs amount of dv_j, the percentage of the decomposed output from oo_i is not higher than c, i.e., $\frac{d_i(dv_j)}{\sum_{k=1}^{n} d_k(dv_j)} \leq c$.

For the example in Figure 2(c), DO is a $\frac{2}{3}$-decomposition. Here, c is a positive decimal smaller than one. If a decomposition solution is not a c-decomposition, some users' privacy will be revealed. For example, as shown in Figure 2(a), since all decomposed outputs with an amount of 10 are from oo_1, it is not a c-decomposition, and adversaries can reveal the linkages between oo_1 and its decomposed outputs. By Definition 4, a c-decomposition guarantee that the posterior belief that adversities have is bounded by c. The smaller the c is, the better the privacy-preserving effect is. Thus, for a set of users whose privacy requirements are different, we can make a c^*-decomposition for them, where c^* is the smallest privacy requirement parameter.

B. The AA-OD Problem

Thus, to get a good privacy-preserving effect, users are motivated to get a c-decomposition of the original outputs. In this subsection, we formally define the anonymity-aware output decomposition (AA-OD) problem as follows.

Definition 6 (The anonymity-aware output decomposition problem)**.** Given a set of original outputs OO and a privacy requirement c, the anonymity-aware output decomposition (AA-OD) problem aims to find a c-decomposition DO with a minimal number of decomposed outputs.

In other words, the answer for the AA-OD problem satisfies the users' privacy requirements and minimizes the transaction fee. For simplicity, in this paper, we assume the original output set OO is sorted by the amount from high to low, i.e., $\forall i \in [2, |OO|]$, $ov_i \leq ov_{i-1}$.

C. NP-hardness

However, unfortunately, the AA-OD problem is intractable. In this subsection, we theoretically prove that the AA-OD problem is NP-hard by reducing from the subset sum problem [20].

Theorem III.1. *The AA-OD problem is NP-hard.*

Proof. We first prove the NP-completeness of the decision version of the AA-OD problem (D-AA-OD) by a reduction from the subset sum problem [20]. Given an original output set, the D-AA-OD problem aims to decide if there is a c-decomposition whose carnality is G. Besides, the subset sum problem can be described as follows: Given a set of different and positive integers $Q = \{q_1, q_2, \cdots, q_n\}$ and an integer K, the subset sum problem aims to find a subset Q' of Q, such that the sum of integers in Q' is equal to K, i.e., $\sum_{q_i \in Q'} q_i = K$.

Given a subset sum problem instance, we reduce it to a D-AA-OD problem instance as follows: Generate a set of original outputs $OO = \{oo_1, oo_2, \cdots, oo_n, oo_{n+1}, oo_{n+2}\}$, where $\forall i \in [1, n]$, $ov_i = q_i$, $ov_{n+1} = K$, and $ov_{n+2} = \sum_{q_i \in Q} q_i - K$. Let c be 0.5 and $G = 2 \cdot n$. Thus, if there is a subset Q' of Q such that $\sum_{q_i \in Q'} q_i = K$, the answer for the transformed D-AA-OD problem instance is yes; otherwise, the answer is no. Thus, if the transformed D-AA-OD problem instance can be solved, the SS problem instance also can be solved. Since the SS problem is NP-complete [20], the D-AA-OD problem is NP-complete. Thus, the AA-OD problem is NP-hard, which completes the proof. □

Table II summarizes the commonly used symbols.

TABLE II
SYMBOLS AND DESCRIPTIONS.

Symbol	Description
OO	the set of original outputs
oo_i	an original output
ov_i	the amount of oo_i
c	the privacy requirement
DO	a set of decomposed output
do_i	a decomposed output
dv_i	the amount of do_i

TABLE III
EXAMPLE OF DECOMPOSITION AS THEOREM IV.2.

ov_1	ov_2	ov_3	ov_4	δ
50	10	7	1	
10	10	-33	-9	40
-33	-33	-33	-82	43
-82	-82	-82	-82	49

IV. THE BOGGART ALGORITHM

In practice, the amount of an original input varies widely. For example, in the Wasabi data sets detected by [2], the average amount of inputs is 46856228 Satoshi, and the highest amount of an input is 71075834767 Satoshi. Thus, it is costly to enumerate all possible amounts for decomposed outputs.

In this section, we propose an approximation algorithm, Boggart, to set the amounts of decomposed outputs by the difference between the amounts of original outputs. We first introduce the basic ideas of the Boggart algorithm. Then, we describe the algorithm in detail and demonstrate a running example. Finally, we theoretically analyze its performance.

A. Basic Ideas

The Boggart algorithm is inspired by three theorems and one observation. The first theorem states that, in a c-decomposition, the decomposed outputs whose amounts are the same come from at least $\lceil \frac{1}{c} \rceil$ original outputs.

Theorem IV.1. *In a c-decomposition, the decomposed outputs whose amounts are the same come from at least $\lceil \frac{1}{c} \rceil$ original outputs.*

Proof. We denote the number of different original outputs that the decomposed outputs amount of x come from as $v(x)$, i.e., $v(x) = |\{ds_i | do_i \in DO, dv_i = x\}|$. Besides, we denote the maximum number of decomposed outputs amount of x which are from the same original output as $d_{\#}(x)$, i.e., $d_{\#}(x) = \max\{d_k(x) | k \in [1, n]\}$. Thus, by Definition 5, $d_{\#}(x) \le c \cdot \sum_{k=1}^{n} d_k(x)$. Thus, $c \ge \frac{d_{\#}(x)}{\sum_{k=1}^{n} d_k(x)} \ge \frac{d_{\#}(x)}{v(x) \cdot d_{\#}(x)} = \frac{1}{v(x)}$. Since $v(x)$ is an integer, $v(x) \ge \lceil \frac{1}{c} \rceil$. □

In other words, if we decompose a decomposed output amount of x from each of $\lceil \frac{1}{c} \rceil$ original outputs, the obtained decomposed output set is a c-decomposition. Inspired by Theorem IV.1, we make the second theorem, which proposes a way to retrieve a c-decomposition of $\lceil \frac{1}{c} \rceil$ original outputs.

Theorem IV.2. *Suppose $OO = \{oo_1, oo_2, \cdots, oo_{\lceil \frac{1}{c} \rceil + 1}\}$ is a set of $\lceil \frac{1}{c} \rceil + 1$ original outputs sorted by the amount from the highest to the slowest, i.e., $\forall i \in [1, \lceil \frac{1}{c} \rceil]$, $ov_i \ge ov_{i+1}$. Then, we repeatedly obtain some decomposed outputs from these original outputs as flows: (1) at the $i^{th}(i \in [1, \lceil \frac{1}{c} \rceil])$ round, we decompose a decomposed output amount of $ov_1^i - ov_{i+1}^i$ from each original output except oo_{i+1}, where ov_j^i is the value of oo_j at the beginning of the i^{th} round; and (2) after $\lceil \frac{1}{c} \rceil$ rounds, for each original output, we turn it to a decomposed output with its updated amount. In this way, we can get a c-decomposition.*

Proof. Suppose ϵ_i is the amount of decomposed outputs obtained in the $(i-1)^{th}$ round, i.e., $\epsilon_i = ov_1^i - ov_i^i$. For each $i \ge 2$, since oo_1 and oo_i both are decomposed with the same amount in each of the first $i-1$ rounds, the difference between ov_1^i and ov_i^i is the original difference, i.e., $\epsilon_i = ov_1^i - ov_i^i = ov_1 - ov_i$. Since oo_1 is decomposed in each round, after $\lceil \frac{1}{c} \rceil$ rounds, the amount of oo_1 is $ov_1 - \sum_{i=2}^{\lceil \frac{1}{c} \rceil + 1} \epsilon_i$. Besides, for any $i \ge 2$, since oo_i is decomposed in each round except the $(i-1)^{th}$ round, after $\lceil \frac{1}{c} \rceil$ rounds, the amount of oo_i is $ov_i - \sum_{j=2}^{\lceil \frac{1}{c} \rceil + 1} \epsilon_j + \epsilon_i = ov_1 - \sum_{j=2}^{\lceil \frac{1}{c} \rceil + 1} \epsilon_j$. Thus, after $\lceil \frac{1}{c} \rceil$ rounds, the updated amount of each original output is the same. Thus, in Step (2), we can get decomposed outputs from more than $\lceil \frac{1}{c} \rceil$ sources. Besides, in each round of Step (1), we get decomposed outputs from $\lceil \frac{1}{c} \rceil$ sources. Thus, by Definition 5, we can get a c-decomposition, which completes our proof. □

Table IV.2 demonstrates an example of decomposition as Theorem IV.2, where δ is the amount of decomposed outputs in each round. The first row illustrates the original amount of each original output, and the $j^{th}(j \ge 2)$ row illustrates the updated amount of each original output after the j^{th} round of decomposition. For instance, since $ov_1 - ov_2 = 42$, in the first round, each original output except oo_2 is decomposed with a decomposed output amount of 42.

Thus, by Theorem IV.2, we can set the amounts of decomposed outputs by the difference between the amounts of original outputs. However, a challenge needs to be solved. As shown in Table III, after decomposition, finally, the amount of ov_i may be negative, and we will get a set of decomposed output with a negative amount in Step (2). As introduced in Subsection II-A, the amount of each output should be positive. Next, we will introduce our observation, which can help to solve this challenge.

Observation: When there are only a few original outputs, centralized servers or decentralized coordinators cannot make a mixing transaction satisfying users' privacy requirements. In this scenario, users have to wait a long time until some original outputs are proposed and a mixing transaction can be made [22]. If the waiting time is too long, users may abandon the mixing, and servers/coordinators cannot gain the mixing fees. Thus, if the original output set only needs a few more original outputs to make a mixing transaction, servers and coordinators will make up for the needed original outputs. These outputs transfer tokens back to the accounts of servers and coordinators. Thus, they only loss a few transaction fees, but they can own much mixing fees from the mixing transaction. Thus, they are motivated to do that. Besides, they have the abilities to do that. Usually, they have some spare tokens in their accounts. For example, researchers fond

TABLE IV
SOLVE THE NEGATIVE AMOUNT CHALLENGE IN TABLE III.

ov_1	ov_2	ov_3	ov_4	cv_3	cv_4	δ
50	10	7	1	43	49	
49	10	6	0	43	49	1
43	10	0	0	43	43	6
10	10	0	0	10	10	33

that the address[2] has been used to store Wasabi coordinators' profits [2]. From September 8, 2019, to September 20, 2019, there were more than 9.88 spared bitcoins in this account.

To distinguish, we term the original outputs that servers or coordinators make up as *compensatory outputs*. For the negative amount challenge shown in Table III, we can solve it by adding some compensatory outputs. Thus, we get the third theorem, which proposes a way to add compensatory outputs and retrieve a c-decomposition.

Theorem IV.3. *Suppose α is the minimum integer such that $\sum_{i=2}^{\alpha} c_i > ov_{\alpha+1}$. Then, for each original output $oo_j (j \geq \alpha + 1)$, we assign it with a compensatory output amount of $ov_1 - ov_j$. We denote the compensatory output of oo_j as co_j and denote its amount as cv_j. Then, we repeatedly obtain some decomposed outputs from these original outputs and compensatory outputs as follows: (1) at the $i^{th} (i \in [1, \alpha - 1])$ round, we decompose a decomposed output amount of $\epsilon_{i+1} = ov_1^i - ov_{i+1}^i$ from each original output except oo_{i+1}. In particular, if at the beginning of the i^{th} round, the amount ov_j of an original output is less than ϵ_{i+1}, we first decompose ov_j from each original output except oo_{i+1}. Then, we decompose $\epsilon_{i+1} - ov_j$ from each original output except oo_{i+1} and oo_j, and we decompose $\epsilon_{i+1} - ov_j$ from oo_j's compensatory output co_j; and (2) after $\alpha - 1$ rounds, for each original output and compensatory output whose amount is positive, we turn it to a decomposed output with its updated amount. In this way, we can get a c-decomposition and the amount of each decomposed output is positive.*

Proof. Since oo_1 is decomposed in each round, after Step (1), the amount of oo_1 is $ov_1 - \sum_{i=2}^{\alpha} \epsilon_i$. Besides, for any $i \in [2, \alpha]$, since oo_i is decomposed in each round except the $(i-1)^{th}$ round, after Step (1), the amount of oo_i is $ov_i - \sum_{j=2}^{\alpha} \epsilon_j + \epsilon_i$ = $ov_1 - \sum_{j=2}^{\alpha} \epsilon_j$. For any $i \in [\alpha + 1, \lceil \frac{1}{c} \rceil + 1]$, the original output oo_i or its compensatory output co_i is decomposed in each round. Thus, we totally subtract $\sum_{j=2}^{\alpha} \epsilon_j$ from ov_i and cv_i. For any $i \in [\alpha + 1, \lceil \frac{1}{c} \rceil + 1]$, since its value is less than $\sum_{i=2}^{\alpha} \epsilon_i$, after Step (1), the amount of oo_i is zero, and the amount of its compensatory output co_i is $cv_i - (\sum_{j=2}^{\alpha} \epsilon_j - ov_i) = ov_1 - \sum_{j=2}^{\alpha} \epsilon_j$.

Thus, after Step (1), the amounts of original outputs $oo_i (i \leq \alpha)$ and the amounts of all compensatory outputs are the same. Besides, since α is the minimum integer such that $\sum_{i=2}^{\alpha} \epsilon_i > ov_{\alpha+1}$, $\sum_{i=2}^{\alpha-1} \epsilon_i \leq ov_\alpha$, $ov_1 - \sum_{i=2}^{\alpha} \epsilon_i$ = $ov_1 - \sum_{i=1}^{\alpha-1} \epsilon_i - \epsilon_\alpha$ $\geq ov_1 - ov_\alpha - (ov_1 - ov_\alpha)$ = 0. Thus, after Step (1), the amount of each original output and each compensatory output is not negative. When $ov_1 - \sum_{i=2}^{\alpha} \epsilon_i$ is positive, in Step (2), we can get decomposed outputs from α original outputs and $\lceil \frac{1}{c} \rceil - \alpha + 1$ compensatory outputs. Thus, in Step (2), we can get decomposed outputs from more than $\lceil \frac{1}{c} \rceil$ sources. Besides, in each round of Step (1), we get decomposed outputs from $\lceil \frac{1}{c} \rceil$ sources. Thus, by Definition 5, in this way, we can get a c-decomposition, which completes our proof. □

[2] address: bc1qs604c7jv6amk4cxqlnvuxv26hv3e48cds4m0ew

Algorithm 1: The Boggart algorithm.

Input: a set of original outputs OO and a privacy requirement c
Output: a c-decomposition DO

partition OO into z groups where each group contains $m = \lceil \frac{1}{c} \rceil + 1$ outputs;
foreach *group* G_i **do**
  foreach $oo_{i,j}$ *in* G_i **do**
    $\epsilon_{i,j} = ov_{i,1} - ov_{i,j}$;
  calculate α_i;
  for $j = \alpha_i + 1$ *to* m **do**
    add a compensatory output $co_{i,j}$ whose value is $cv_{i,j} = ov_{1,1} - ov_{i,j}$;
  for $j = 2$ *to* α_i *do* **do**
    $\delta = \min\{\epsilon_{i,j}, \min\{ov_{i,k} | ov_{i,k} > 0\}\}$;
    while $\epsilon_{i,j} > 0$ **do**
      $ov_{i,1} = ov_{i,1} - \delta$, $\epsilon_{i,j} = \epsilon_{i,j} - \delta$;
      foreach $k \in [2, j) \cup (j, m]$ **do**
        if $ov_{i,k} <= 0$ **then**
          $cv_{i,k} = cv_{i,k} - \delta$;
        else
          $ov_{i,k} = ov_{i,k} - \delta$;
        update DO, δ and $\epsilon_{i,j}$;
  foreach $oo_{i,j}$ *and* $co_{i,j}$ **do**
    if *its value is not zero* **then**
      turn it to a decomposed output, update DO;
return DO;

Table IV illustrates how to solve the negative amount challenge in Table III by adding compensatory outputs as the way in Theorem IV.3. Since $\sum_{i=2}^{2} \epsilon_i > ov_3$, $\alpha = 2$. Thus, we add two compensatory outputs, cv_3 and cv_4, for ov_3 and ov_4. In the first round, since $ov_4 < \epsilon_1 = 40$, we first decompose 1 from oo_1, oo_3, and oo_4 (as shown in the first row). Then, we continue to decompose 39 from oo_1, oo_3, and cv_4. However, since the updated amount of ov_3 is $6 < 39$, we first decompose 6 from oo_1, oo_3, and cv_4 (as shown in the second row). After that, we decompose 33 from oo_1, cv_3, and cv_4 (as shown in the third row). Finally, $ov_1 = ov_2 = cv_3 = cv_4$, and we turn them to decomposed outputs amount of 10.

B. The Boggart Algorithm

Inspired by the basic ideas, we design the Boggart algorithm. Algorithm 1 illustrates the pseudocode of the Boggart algorithm. It partitions the original output set Out into groups,

TABLE V
POSSIBLE MATCHES OF FIGURE 2(C).

G_1				G_2			
oo_1	oo_2	oo_3	$cv_{1,3}$	oo_4	oo_5	oo_6	$cv_{2,3}$
50	10	7	43	1	0	0	1
43	10	**0**	43	**0**	0	0	**0**
10	10	0	**10**				
0	**0**	0	**0**				

where each group G_i contains $m = \lceil \frac{1}{c} \rceil + 1$ original outputs (line 1). Since $n = |OO|$ may not be a multiple of m, the last group G_z may not contain m original outputs. We consider the missing original outputs in G_z are those amount of zero. Thus, each group contains $\lceil \frac{1}{c} \rceil + 1$ original outputs. Next, the algorithm independently decomposes the original outputs in each group G_i (line 2-20). We denote $oo_{i,j}$ as the j^{th}-biggest output in G_i and denote $ov_{i,j}$ as its amount. For each $oo_{i,j}$ in G_i, the algorithm first calculates the difference between $ov_{i,j}$ and $ov_{i,1}$, i.e., $\epsilon_{i,j} = ov_{i,1} - ov_{i,j}$ (line 3-4). Then, we calculate the minimum integer α_i such that $\sum_{j=2}^{\alpha_i} \epsilon_{i,j} > ov_{i,\alpha_i+1}$ (line 5). For each $oo_{i,j}$ where $j > \alpha_i$, we add a compensatory output $co_{i,j}$ whose value is $cv_{i,j} = vo_{i,1} - vo_{i,j}$ (line 6-7). Next, we decompose these original outputs and compensatory outputs as the way in Theorem IV.3 (line 8-17). Since the amount of any $oo_{i,k}$ cannot be negative, we set the value of a decomposed output as $\delta = \min\{\epsilon_{i,j}, \min\{ov_{i,k} | ov_{i,k} > 0\}\}$ (line 9). If the current amount of an original output is zero, we get a decomposed output amount of δ from its compensatory output (line 13-14); otherwise, we get a decomposed output amount of δ from itself (line 15-16). Then, for each original output and compensatory output whose amount is positive, we turn it to a decomposed output (line 18-20). Finally, we return the set of decomposed outputs (line 21).

Here is an example of running Boggart to retrieve a $\frac{1}{2}$-decomposition of the instance in Figure III. As shown in Table V, we partition OO into two groups where each contains $m = 3$ outputs. Specifically, the first group G_1 is $\{oo_1, oo_2, oo_3\}$ and the second group G_2 is $\{oo_4, oo_5, oo_6\}$, where oo_5 and oo_6 are two virtual outputs whose value is 0. Thus, $\alpha_1 = \alpha_2 = 2$, and we add $co_{1,3}$ and $co_{2,3}$, where $cv_{1,3} = 43$ and $cv_{2,3} = 1$. Each row represents the remaining value of each output, where the values that are updated in the round is in bold. Finally, we obtain a decomposed output amount of 7 from oo_1 and oo_3, respectively. We obtain a decomposed output amount of 33 from oo_1 and $co_{1,3}$, respectively. We obtain a decomposed output amount of 10 from oo_1, oo_2, and $co_{1,3}$, respectively. We obtain a decomposed output amount of 1 from oo_4 and $co_{2,3}$, respectively. Thus, the decomposed output set is a $\frac{1}{2}$-decomposition.

C. Theoretical Analysis

In this subsection, we theoretically analyze the performance of our Boggart algorithm. We first prove that the Boggart algorithm can return the result within polynomial time.

Theorem IV.4. *The time complexity of the Boggart algorithm is $\mathcal{O}(\frac{n}{c} + \frac{1}{c^2})$, where n is the number of original outputs in OO, and c is the privacy requirement.*

Proof. In each round of the while-loop at line 10, at least one $\epsilon_{i,j}$ or one ov_k becomes zero. For each group, when the for-loop at line 8 terminates, at most $m-1$ $\epsilon_{i,j}$ and $m-1$ $ov_{i,k}$ have been zero. Thus, for each group, when the for-loop at line 8 terminates, the while-loop at line 10 runs at most $2 \cdot m - 1$ rounds. Thus, the time complexity of each round of the for-loop at line 2 is $\mathcal{O}(m^2)$. Since there are z groups, the time complexity of the Boggart algorithm is $\mathcal{O}(m^2 \cdot z)$. Since $m \cdot z < n + m$ and $m = \lceil \frac{1}{c} \rceil + 1$, the time complexity of the Boggart algorithm is $\mathcal{O}(m^2 \cdot z) = \mathcal{O}(\frac{n}{c} + \frac{1}{c^2})$. $\square$

When c is fixed, the time complexity of the Boggart algorithm is linear with the number of original outputs. Besides, when the privacy requirement is higher (i.e., c is smaller), or the number of original outputs is larger (i.e., n is larger), it is harder to retrieve a c-decomposition, and the running time is higher, which fits our Theorem IV.4. Next, we prove the approximation ratio of our Boggart algorithm.

Theorem IV.5. *The approximation ratio of the Boggart algorithm is $\frac{2}{c} + 3$, where c is the privacy requirement.*

Proof. Denote S^* as the minimal size of the optimal c-decomposition, and denote $S^\#$ as the size of the c-decomposition obtained by the Boggart algorithm. As proved in Theorem IV.4, for each group, the while-loop at line 10 runs at most $2m - 2$ rounds, and each round generates $m - 1$ decomposed outputs. Besides, as proved in Theorem IV.3, the for-loop at line 18 generates at most m decomposed outputs. Thus, for each group, the Boggart algorithm generates at most $2 \cdot (m-1)^2 + m$ decomposed outputs.

When there is only one group, $m \geq n$. By Theorem IV.1, there are at least $m - 1$ decomposed outputs in a c-decomposition. Thus, $S^* \geq m-1$ and $S^\# \leq 2 \cdot (m-1)^2 + m$. Thus, $\frac{S^\#}{S^*} \leq \frac{2\cdot(m-1)^2+m}{m-1} < 2 \cdot (m-1) + 1$. Since $m < \frac{1}{c} + 2$, $\frac{S^\#}{S^*} < 2 \cdot m - 1 < \frac{2}{c} + 3$. When there is more than one group, $n > m \cdot (z-1)$. Since $S^* \geq n$, $\frac{S^\#}{S^*} \leq \frac{2\cdot(m-1)^2+m}{m\cdot(z-1)}$ = $\frac{2\cdot m-3}{z-1} + \frac{2}{m\cdot(z-1)}$. Since $2 \leq m < \frac{1}{c} + 2$ and $z \geq 2$, $\frac{S^\#}{S^*} \leq 2 \cdot (m-3) + 1 < \frac{2}{c} + 2$. Thus, the theorem is proved. $\square$

Thus, the number of decomposed outputs in DO returned by the Boggart algorithm will never exceed the $\frac{2}{c} + 3$ times the minimal number of decomposed outputs in the optimal decomposition solution. Besides, when the privacy requirement is stricter, it is harder to retrieve a c-decomposition, and the difference between the number of decomposed outputs in the solution obtained by the Boggart algorithm and that in the optimal solution will be larger, which fits our Theorem IV.5.

Besides, as we introduced in Subsection IV-A, only when the tokens of extra compensatory outputs are few, servers and coordinators will make up the compensatory outputs. Since the sum of original outputs' amounts differs widely, to fairly estimate the tokens of compensatory outputs that a solution needs, we define the concept of compensatory ratio.

Definition 7 (Compensatory ratio)**.** Given an original output set OO and a set of compensatory outputs CO that a solution needs, the compensatory ratio cr of this solution is the ratio of the sum of compensatory outputs' amounts to the sum of original outputs' amounts, i.e., $cr = \frac{\sum_{co_i \in CO} cv_i}{\sum_{oo_j \in OO} ov_j}$.

For the example in Table IV, the compensatory ratio is $\frac{43+49}{50+8+7+1} = \frac{46}{33}$. The smaller the compensatory ratio, the easier the solution is to be implemented. For example, if the compensatory ratio is 100, to mix a set of original outputs whose total amount is 10 bitcoin, servers/coordinators need to use 1000 bitcoin. Although these bitcoins will be transferred back to servers/coordinators, it is still difficult to execute. We prove that the compensatory ratio of the decomposition solution obtained by our Boggart algorithm is limited.

Theorem IV.6. *For any AA-OD problem instance, the compensatory ratio of the decomposition solution obtained by the Boggart algorithm is bounded by* $\frac{1}{c}$.

Proof. Denote Φ_i as the sum of compensatory outputs' amounts in the group G_i, and denote Φ as the sum of compensatory outputs' amounts in z groups. Thus, $\Phi_i = \sum_{j=\alpha_i+1}^{m}(ov_{i,1} - ov_{i,j})$. Since $\alpha_i \geq 2$, $\Phi_i \leq \sum_{j=3}^{m}(ov_{i,1} - ov_{i,j}) = (m-2) \cdot ov_{i,1} - \sum_{j=3}^{m} ov_{i,j}$. Thus, the sum of compensatory outputs' amounts in z groups is $\Phi = \sum_{i=1}^{z} \Phi_i = (m-2) \cdot \sum_{i=1}^{z} ov_{i,1} - \sum_{i=1}^{z} \sum_{j=3}^{m} ov_{i,j}$. Since the original outputs are sorted from biggest to smallest, $\forall i \geq 2, \forall j \in [1,m]$, $ov_{i,1} \leq ov_{i-1,j}$. Thus, $\forall i \geq 2$, $ov_{i,1} \leq \frac{\sum_{j=3}^{m} ov_{i-1,j}}{m-2}$. Thus, $\Phi \leq (m-2) \cdot ov_{1,1} - \sum_{j=3}^{m} ov_{z,j} < (m-2) \cdot ov_{1,1} = (m-2) \cdot ov_1$. Thus, the compensatory ratio is $\frac{\Phi}{\sum_{oo_i \in Out} ov_i} \leq m-2 \leq \frac{1}{c}$. □

Thus, the number of tokens in compensatory outputs that the algorithm needs is bounded by the $\frac{1}{c}$ times the sum of original outputs' amounts. When the privacy requirement is stricter, it is harder to retrieve a c-decomposition and more compensatory outputs are needed, which fits our Theorem IV.6.

Discussion. In practice, some users do not need a high privacy-preserving effect, particularly when they do not want to pay extra fess [23]. For example, researchers have found that many users only mix their identities with another identity [24], whose privacy preserving effect is equal to the case where $c = 0.5$. In other words, these users only need that the attacker cannot 100% determine their transactions. In 80% of the transactions in the Wasabi data sets detected by [2], some decomposed outputs are mixed with only one decomposed output with the same amount. For example, in the transaction [3], there are only two decomposed outputs amount of 348645952 Satoshi. In our technical report [21], we define this special case of the general AA-OD problem as the M-AA-OD problem, where the privacy requirement c is at least $\frac{1}{2}$ (i.e., the majority of the decomposed outputs whose amounts are the same can be from the same original output). For the M-AA-OD problem, we propose a 2-approximation algorithm, namely Polyjuice[4].

[3]hash value: 038ef30e6b51a08b834b0ae2f8a5b39d24c2c5390ef6526b0c4b-a7af49d92451

[4]The Polyjuice Potion is a potion in J. K. Rowling's Harry Potter series that allows a drinker to shift her/his shape.

Algorithm 2: The Polyjuice algorithm.

Input: a set of original outputs OO
Output: a $\frac{1}{2}$-decomposition D

1 **if** $ov_1 > \sum_{i=2}^{n} ov_i$ **then**
2   add a compensatory output co_1 whose value $cv_1 = ov_1 - \sum_{i=2}^{n} ov_i$;
3   **for** $i = 2$ *to* n *do* **do**
4         $ov_1 = ov_1 - ov_i$, $ov_i = 0$;
5   update D;
6     $ov_1 = ov_1 - cv_1$, $cv_1 = 0$;
7 **else**
8   add a compensatory output co_1 whose value $cv_1 = \sum_{i=2}^{n} ov_i - ov_1$;
9   **for** $i = 2$ *to* n *do* **do**
10   **if** $ov_1 > ov_i$ **then**
11             $ov_1 = ov_1 - ov_i$, $ov_i = 0$;
12   **else**
13   **if** $ov_1 > 0$ **then**
14                 $ov_i = ov_i - ov_1$, $ov_1 = 0$;
15             $cv_1 = cv_1 - ov_i$, $ov_i = 0$;
16   update D;
17 **return** D;

Compared with Boggart, the number of decomposed outputs in the solution obtained by Polyjuice is 15% smaller.

V. The Polyjuice Approach

In practice, some users do not need high privacy-preserving effect, particularly when they do not want to pay extra fess [23]. Thus, in this section, we propose an approximation method for a special AA-OD problem instance, named M-AA-OD problem, where the privacy requirement c is at least 0.5.

Definition 8 (The M-AA-OD problem)**.** Given a set of original outputs $OO = \{oo_1, \cdots, oo_n\}$, and a privacy requirement $c \geq 0.5$, the majority-anonymity-aware output decomposition (M-AA-OD) problem aims to find a c-decomposition D with the minimal size.

A. The Polyjuice Algorithm

We observe that when c is at least 0.5, by Theorem IV.1, the decomposed outputs with the same amounts can only be decomposed from two original outputs. Besides, by Definition 5, $\forall c \geq 0.5$, we observe that if one decomposition is a 0.5-decomposition, it must be a c-decomposition. Inspired by these two observations, we propose a 2-approximation algorithm, the Polyjuice algorithm, whose pseudocode is shown in Algorithm 2, which repeatedly gets two same-amount decomposed outputs from two sources.

Specifically, when the amount of oo_1 is larger than the sum of the other original outputs' amounts, it first adds a compensatory output co_1 whose amount is $cv_1 = ov_1 - \sum_{i=2}^{n} ov_i$ (line

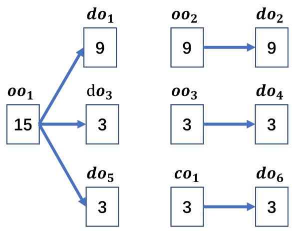


Fig. 5. The decomposition of Example 2.

2). Then, it decomposes oo_1 into $n+1$ decomposed outputs whose amounts are equal to the amounts of the outputs in $OO \backslash oo_1 \cup co_1$ (line 3-6). When the amount of oo_1 is not larger than the sum of the other original outputs' amounts, it first adds a compensatory output co_1 whose amount is $vc_1 = \sum_{i=2}^{n} ov_i - ov_1$ (line 8). Then, it gets a decomposition where the decomposed outputs whose amounts are the same come from two outputs, one is oo_1 or co_1, and one of the other original outputs (line 9-16). Specifically, if the amount of oo_1 is not less than the amount of oo_i, we get two decomposed outputs from oo_1 and oo_i whose amounts are both ov_i (line 11). If the amount of oo_1 is positive but less than the amount of oo_i, we first get two decomposed outputs from oo_1 and oo_i whose amounts are both ov_1 (line 14). Next, we get two decomposed outputs from co_1 and oo_i whose amounts are both ov_i (line 15). If the amount of oo_1 is zero, we get two decomposed outputs from co_1 and oo_i whose amounts are both ov_i (line 15). Since $cv_1 = |ov_1 - \sum_{i=2}^{n} ov_i|$, when the Polyjuice algorithm terminates, all original outputs and the compensatory output are fully decomposed. Finally, the algorithm returns D (line 17). Since it always generates two decomposed outputs with the same amount coming from two sources, D must be a $\frac{1}{2}$-decomposition. Since $c \geq 0.5$, D is a c-decomposition. Here is an example.

Example 2. *There is a set of original outputs* $OO = \{oo_1, oo_2, oo_3\}$*, where* $ov_1 = 15$*,* $ov_2 = 9$*, and* $ov_3 = 3$*. The privacy requirement is* $c = 0.8$*.*

For Example 2, since $ov_1 > ov_2 + ov_3$, we add a compensatory output co_1 whose amount is 3. Then, we get a decomposition shown in Figure 5. The decomposition satisfies the requirement of 0.8-decomposition.

B. Theoretical Analysis

In this subsection, we theoretically analyze the effectiveness and the efficiency of our Polyjuice algorithm. We first prove the time complexity of the Polyjuice algorithm.

Theorem V.1. *The time complexity of the Polyjuice algorithm is* $\mathcal{O}(n)$*, where* n *is the number of original outputs in* OO*.*

Proof. The time complexity of each round of the for-loop at line 3 is $\mathcal{O}(1)$, and the for-loop runs $\mathcal{O}(n)$ rounds. Thus, when $ov_1 > \sum_{i=2}^{n} ov_i$, the time complexity is $\mathcal{O}(n)$. The time complexity of each round of the for loop at line 9 is $\mathcal{O}(1)$, and the for-loop runs $\mathcal{O}(n)$ rounds. Thus, when $ov_1 \leq \sum_{i=2}^{n} ov_i$, the time complexity is $\mathcal{O}(n)$. Thus, the theorem is proved. □

Thus, the time complexity of the Polyjuice algorithm is linear with the number of original outputs. Next, we prove the approximation ratio of the Polyjuice algorithm.

TABLE VI

EXPERIMENTAL SETTINGS.

Parameters	Values
privacy requirement c	0.01, 0.2, **0.4**, 0.6, 0.8
number of original outputs n	**80**, 160, 240, 320, 400
mean μ of original outputs' amounts	$5 \cdot 10^3$, $5 \cdot 10^5$, $\mathbf{5 \cdot 10^7}$, $5 \cdot 10^9$, $5 \cdot 10^{11}$
variance σ of original outputs' amounts	$0.02 \cdot \mu$, $0.2 \cdot \mu$, $\mathbf{2 \cdot \mu}$, $20 \cdot \mu$, $200 \cdot \mu$

Theorem V.2. *The approximation ratio of the Polyjuice algorithm is 2.*

Proof. Denote S^* as the size of the optimal decomposition and $S^\#$ as the size of the decomposition obtained by the Polyjuice algorithm. The for-loop at line 3 runs $n-1$ rounds, and each round generates two decomposed outputs. At line 6, the algorithm also generates two decomposed outputs. Thus, when $ov_1 > \sum_{i=2}^{n} ov_i$, $S^\#$ is $2 \cdot n$. The for-loop at line 9 runs $n-1$ rounds. There is at most one round of the for-loop generates four decomposed outputs, and each of other rounds generates two decomposed outputs. Thus, when $ov_1 > \sum_{i=2}^{n} ov_i$, $S^\#$ is at most $2 \cdot n$. Since $S^* \geq n$, $\frac{S^\#}{S^*} \leq \frac{2 \cdot n}{n} = 2$. □

In other words, in the worst case, the size of the decomposition D obtained by the Polyjuice algorithm will not exceed the twice the size of the optimal decomposition. Finally, we prove that, the amount of the added compensatory output is bounded.

Theorem V.3. *For any AA-OD problem instance where* $c \geq 0.5$*, the compensatory ratio of the Polyjuice algorithm is bounded by* 1*.*

Proof. When ov_1 is smaller than $\sum_{i=2}^{n} ov_i$, the compensatory output's amount is $\sum_{i=2}^{n} ov_i - ov_1$. The compensatory ratio is $\frac{\sum_{i=2}^{n} ov_i - ov_1}{ov_1 + \sum_{i=2}^{n} ov_i} = 1 - \frac{2}{1 + \frac{\sum_{i=2}^{n} ov_i}{ov_1}} \leq 1 - \frac{2}{n} < 1$. Similarly, when ov_1 is larger than $\sum_{i=2}^{n} ov_i$, the compensatory ratio is also less than 1. Thus, the theorem is proved. □

In other words, the number of tokens that the compensatory output needed in the decomposition solution obtained by the Polyjuice algorithm is less than the the sum of original outputs' amounts.

VI. EXPERIMENTAL STUDY

To test our proposed algorithm, we conduct comprehensive experiments over both real and synthetic data sets. In this section, we first introduce the baseline solutions that will be compared with our Boggart algorithm. Then, we introduce the configuration of our experiments. Next, we illustrate the experimental results on both real and synthetic data sets to show the advance of our Boggart algorithm, compared with two baseline solutions. Finally, we summarize our finds from the experiments. All experiments were run on an Intel CPU @1.3 GHz with 32GB RAM in Java.

A. Baseline Solutions

As proved in Theorem III.1, the AA-OD problem is NP-hard. Thus, it is infeasible to get the optimal result as the ground truth. As an alternative, we will compare our Boggart algorithm with two baseline solutions, the *Decimalism-Greedy (DG)* approach and the *Decimalism-Random (DR)* approach.

In practice, the original outputs might not be enough to make a $c-$decomposition. For the example in Table IV, we can find that we cannot get a $\frac{1}{3}$-decomposition for them. We theoretically prove that when the maximum amount of an original output is higher than c times the sum of all original outputs' amounts, there does not exist a c-decomposition.

Theorem VI.1. *It is a necessary and sufficient condition of having a valid c-decomposition of OO that* $ov_1 <= c \cdot \sum_{oo_i \in OO} ov_i$.

Proof. We first prove the necessity. Assume there exists a c-decomposition for OO. Suppose $d_{max}(x) = \max\{d_i(x)|oo_i \in OO\}$, where $d_i(x)$ is the number of decomposed outputs amount of x that are decomposed from oo_i, i.e., $d_i(x) = |\{do_j|do_j \in DO, dv_j = x, ds_j = oo_i\}|$. Thus, by Definition 5, for any x, $d_{max}(x) \leq c \cdot \sum_{oo_i \in OO} d_i(x)$. Therefore, $ov_1 \leq \sum_{x=1}^{ov_1} d_{max}(x) \cdot x \leq \sum_{x=1}^{ov_1} c \cdot x \cdot \sum_{oo_i \in OO} d_i(x) = c \cdot \sum_{oo_i \in OO} ov_i$. It violates the fact that $ov_1 > c \cdot \sum_{oo_i \in OO} ov_i$. Thus, there is no c-decomposition for OO. Thus, the necessity is proved. Then, we prove the sufficiency. When $ov_1 <= c \cdot \sum_{oo_i \in OO} ov_i$, it is a valid c-decomposition that decomposing each original output into a set of decomposed outputs whose values are all 1, (i.e., $\forall oo_i \in OO, d_i(1) = ov_i$). Thus, the theorem is proved. □

As we introduced in Subsection IV-A, when the original outputs are not enough to make a mixing transaction, services and coordinators will make up the compensatory outputs. Thus, the DG approach and the DR approach first check if the original outputs are enough to make a c-decomposition. If not, they will add a set of compensatory outputs $CO = \{co_1, \cdots, co_h\}$, where $h = \lfloor \frac{\lceil \frac{ov_1}{c} \rceil - \sum_{oo_i \in OO} ov_i}{ov_1} \rfloor + 1$, $\forall j \in [1, h-1]$, $cv_j = ov_1$, and $cv_h = \frac{\lceil \frac{ov_1}{c} \rceil - \sum_{oo_i \in OO} ov_i}{ov_1} - (h-1) \cdot ov_1$. Then, the DG approach gets a c-decomposition in a greedy manner, and the DR approach gets a c-decomposition in a random manner:

- The DG approach is adapted from existing works (e.g., Dash), which decompose original outputs and compensatory outputs into standard denominations, like 1, 10, 100, and so on [11]. It obtains decomposed outputs with standard denominations from the highest to the lowest. Specifically, for a denomination x, it first tries to obtain as many as decomposed outputs amount of x from each original output and compensatory output, denoted as DO'. In other words, in DO', $d_i(x) = \lfloor \frac{ov_i}{x} \rfloor$. If DO' is a valid $\frac{1}{c}$-decomposition and the remaining amounts of the original outputs and compensatory outputs satisfy Theorem VI.1, we decompose the original outputs and compensatory outputs as DO'; otherwise, we abandon to obtain decomposed outputs with this denomination. Then, the DG solution turns to obtain decomposed outputs with the next denomination.
- The DR approach works like the DG solution except that (1) in each round except the last round, it randomly set $d_i(x)$ between zero and $\lfloor \frac{ov_i}{x} \rfloor$; and (2) in the last round, for each original output and compensatory output, the DR approach turns it into x decomposed outputs amount of one, where x is its remaining amount.

For the **example** in Figure IV, the DG approach first adds two compensatory outputs co_1 and co_2, where $cv_1 = 50$ and $cv_2 = 32$. Then, since the highest amount is less than one hundred, the DG approach first tries to decompose these outputs into decomposed outputs amount of ten. Since only ov_1, ov_2, cv_1, and cv_2 are not less then tan, we can get five, one, five, and three decomposed outputs amount of ten from ov_1, ov_2, cv_1, and cv_2 respectively. By Definition 5, these fourteen decomposed outputs violate the requirement of $\frac{1}{3}$-decomposition. Thus, the DG approach abandons the denomination of ten and tries to decompose the outputs into the denomination of one.

For the example in Figure IV, the DR approach also first adds two compensatory outputs co_1 and co_2. Then, since the amounts of ov_1, ov_2, cv_1, and cv_2 are at least ten, the DR approach first tries to decompose these four outputs into decomposed outputs amount of ten. Suppose the DR randomly gets a decomposition solution DO' which decomposes one decomposed output amount of ten from each of these four outputs. If we decompose like DO', the updated amount of outputs are $ov_1 = 40$, $ov_2 = 0$, $ov_3 = 7$, $ov_4 = 1$, $cv_1 = 40$, and $cv_2 = 22$. By Theorem VI.1, since $ov_1 > \frac{1}{3} \cdot (\sum_{i=1}^{4} ov_i + \sum_{i=1}^{2} cv_i)$, we cannot get a $\frac{1}{3}$-decomposition from the remaining outputs. Thus, the DR approach abandons to decompose like DO' and tries to decompose the outputs into the denomination of one.

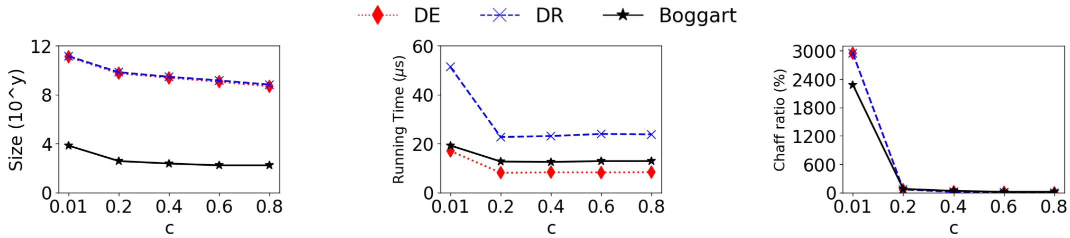
Since the largest denomination is related to the largest amount of the original output, the time complexities of these two baselines are not polynomial to the number of original outputs.

B. Experiment Configuration

We test our Boggart algorithm over real and synthetic data sets.

Real data sets. We use Wasabi mixing transaction data sets from [2] as the real data sets. The authors of [2] proposed an approach to identify mixing transactions. They implemented the approach on Bitcoin and crawled transaction data sets where mixing services are provided by Wasabi [4]. The data sets contain 13581 transactions. For each experiment, we randomly select a transaction in the data sets as an AA-OD problem instance. We take the input sets of the transaction as the original outputs that need to be decomposed. In the data sets, a transaction contains at most 100 decomposed outputs with the same amount. For example, the transaction[5] contains

[5]hash value:16141e9c4f4b1ffdef970d96cf5cd39e6acd6101219bc22ad1cc80-3fb7da2586



(a) Size of varying c (b) Time of varying c (c) Compensatory ratio of varying c

Fig. 6. Results of varying c (Real).

100 decomposed outputs whose amounts are all 10011638 Satoshi. By Definition 5, the privacy requirement c is at least 0.01. Thus, the privacy requirement c in the real data sets is at least 0.01. Since c is a decimal less than 1, we vary the privacy requirement c from 0.01 to 0.8.

Synthetic data sets. Furthermore, to test the performances of our Boggart algorithm in different distributions of original outputs' amounts and the number of original outputs, we generate the synthetic data sets and conduct experiments on them. For each synthetic problem instance, we generate n original outputs. The amount of each original output is randomly set with a Gaussian distribution. The mean value of the Gaussian distribution is μ, and the variance is σ. In the Wasabi mixing transaction data sets, the number of inputs in a transaction is at most 385, and the average number of inputs in a transaction is 74. Thus, we vary the number of original outputs from 80 to 400. In the real data sets, the average amount of inputs is 46856228 Satoshi, and the highest amount of an input is 71075834767 Satoshi. Thus, we vary the mean value of the Gaussian distribution from $5 \cdot 10^3$ to $5 \cdot 10^{11}$ Satoshi. For the distribution of inputs' amount in a transaction in the real data sets, the highest standard deviation is around 123 times the mean amount of these inputs. Thus, we vary σ from $0.02 \cdot \mu$ to $200 \cdot \mu$. Like setting in the real data sets, we vary the privacy requirement c from 0.01 to 0.8.

Comparison metric. For each experiment, we sample 10000 problem instances. We report the average amount of the algorithms' running time, the sizes of obtained c-decompositions, and compensatory ratios:

- The running time of an algorithm is the time that the algorithm used to get an answer for the problem instance. The less the running time, the more efficient the algorithm.
- A decomposition's size is the number of decomposed outputs. The smaller the size, the more effective the algorithm.
- As defined in Definition 7, the compensatory ratio of an algorithm is the ratio of the sum of compensatory outputs' amounts to the sum of original outputs' amounts representing how many tokens in compensatory outputs an algorithm needs. The lower the compensatory ratio, the easier it is to implement the c-decomposition returned by the algorithm.

Testing parameters. In our theoretical analyses, the performances of algorithms (e.g., running time, the size of obtained decomposition, the chaff ratio) are related to the privacy requirement c, the number of original outputs n, and the amount of original outputs. Thus, in our experiments, we test the effects of the privacy requirement c, the number of original outputs n, the mean amount of original outputs, and the variance of original outputs' amounts. Table VI illustrates experiment settings on two data sets, where we mark the default values of parameters in bold font. In each group of experiments, we vary the value of one parameter while setting other parameters' values to their default values.

C. *Effectiveness test on Real data sets.*

To exam the effects of the privacy requirement c, we run the experiments on the real data sets.

Effect of the privacy requirement c. As shown in Figure 6(a), when users' privacy requirements are more relaxed (i.e., c is bigger), the sizes of the c-decompositions obtained by three approaches all decrease. The reason is that when the privacy requirement is more relaxed, more candidate c-decompositions satisfy the privacy requirement, and approaches can return c-decompositions with smaller sizes. Compared with the c-decompositions obtained by the two baseline algorithms, the sizes of c-decompositions obtained by our Boggart algorithm are much smaller. As shown in Figure 6(b), with the increase of c, in the beginning, the running time of three approaches all drops dramatically, and then it almost keeps stable. When c is bigger, it is easier for the solutions to find decomposition satisfying the c-decomposition requirement. Thus, in the beginning, with the increase of c, the running time of the three algorithms decrease. However, when c is large enough, c-decomposition constraint is relaxed, and other features dominate the running time of algorithms, like the number of original outputs. Thus, when c is large enough, with the increase of c, the running time of the three algorithms almost keep stable. As shown in Figure 6(c), with the increase of c, the compensatory ratios of the three solutions all decrease. When c gets larger, by Theorem VI.1, it is more likely that there exists a c-decomposition of original outputs. Thus, two baseline algorithms need fewer tokens in compensatory outputs. For our Boggart algorithm, when c gets larger, m is smaller, and each group contains fewer outputs. Thus, the sum of compensatory outputs' amounts is smaller, which fits our theoretical analysis in Theorem IV.6.

In the experiments on the real data sets, we find that the sizes of the c-decompositions obtained by our Boggart approach are much smaller than those of the c-decompositions obtained by the two baselines. By Theorem IV.1, when $\frac{1}{c}$ is close to the number of outputs, the sources of the decomposed outputs whose amounts are the same will cover almost all original outputs. Since the DG algorithm decomposes original outputs by decimal bits, the decimal bits of some original outputs in a round may be zero. For the example in Subsection VI-A, in the round of the denomination of ten,

$d_3(10) = 0$. Then, when the number of original outputs is close to $\frac{1}{c}$, the decomposed outputs in this case will violate the privacy requirement. Then the DG algorithm will turn to obtain the decomposed outputs amount of smaller denominations, which increases the sizes of c-decompositions dramatically.

D. Effectiveness test on Synthetic data sets.

Then, to show the effect of the number of original outputs and the distribution of their amounts, we generate the synthetic data sets and conduct experiments over them. Besides, we also examine the effects of the privacy requirement over the synthetic data sets, whose results, shown in Figures 7(d), 7(h), and 7(l), are similar to those over the real data sets.

Effect of the number of original outputs n. As shown in Figure 7(a), when the number of original outputs increases, the sizes of the c-decompositions obtained by the three algorithms all increase. The reason is that when n gets larger, more original outputs need to be decomposed, which increases the number of decomposed outputs. Since the DR algorithm makes c-decompositions randomly, the sizes of the c-decompositions obtained by it are extremely high. Since in the experiments of varying n, the number of original outputs is much bigger than $\frac{1}{c}$, the sizes of the c-decompositions obtained by the DG algorithm are much better than those obtained by the DR algorithm. However, they are still much bigger than those obtained by the Boggart algorithm. As shown in Figure 7(e), when n increases, the running time of the three algorithms increases since more original outputs need to be decomposed. The running time of our Boggart algorithm increases linearly with n, which fits our theoretical analysis in Theorem IV.4. As shown in Figure 7(i), when n gets larger, the compensatory ratio of our Boggart algorithm decreases. When n gets larger, the Boggart algorithm may need more compensatory outputs. But meanwhile, compared with the increase of compensatory outputs' amounts, the sum of original outputs' amounts increases much more. Thus, the compensatory ratio decreases. In the experiments of varying n, in most instances, there exists c-decomposition of original outputs. Since the two baselines just need compensatory outputs to satisfy Theorem VI.1, the compensatory outputs they used are almost zero.

Effect of the mean amount of original outputs μ. As shown in Figure 7(b), when the mean amount of original outputs increases, the sizes of the c-decompositions obtained by the two baseline algorithms both increase. Since the two baseline algorithms decompose original outputs into standard denominations, when the mean amount of original outputs becomes larger, they will get more decomposed outputs. However, with the increase of μ, the sizes of the c-decompositions obtained by our Boggart algorithm almost keep stale. The reason is that our Boggart algorithm makes c-decompositions by the differences between original outputs. Thus, the change of μ has no effect on our Boggart algorithm. As shown in Figure 7(f), when μ increases, the running time of the two baseline algorithms increases. Thus, when μ increases, the number of candidate denominations increases, which makes the running time larger. However, with the increase of μ, the running time of our Boggart algorithm almost keeps stable, which fits our theoretical analysis in Theorem IV.4. As shown in Figure 7(j), when μ becomes larger, the compensatory ratios of our Boggart algorithm almost keep stable. The reason is that the Boggart algorithm adds compensatory tokens by the differences between amounts. Thus, the change of μ has no effect on the compensatory ratio of the Boggart algorithm.

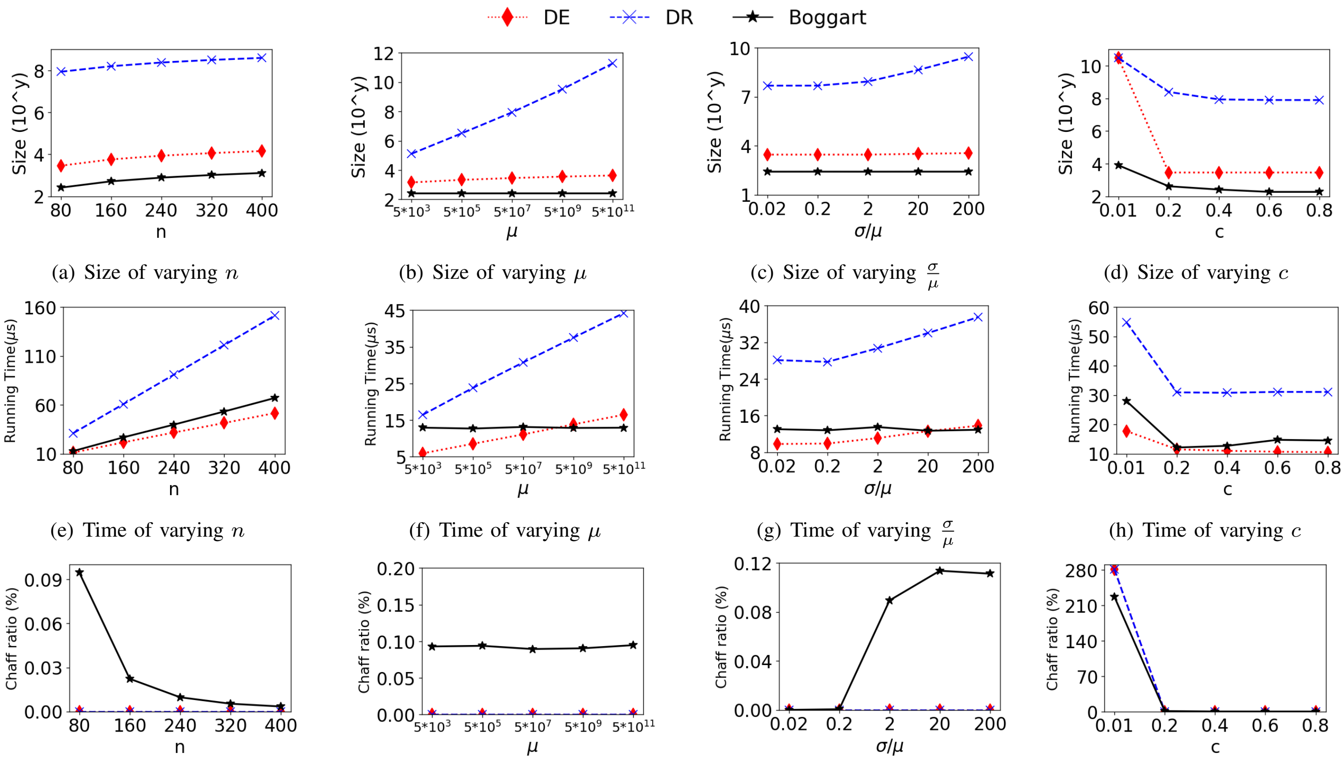
Effect of the variance of original outputs' amounts σ. As shown in Figures 7(c) and 7(g), when σ increases, the sizes of the c-decompositions obtained by the two baseline algorithms and the running time of the two baseline algorithms all increase. The reason is that when σ is larger, the differences between original outputs' amounts are larger, it is more difficult to make c-decompositions. However, since our Boggart algorithm can handle the differences between original outputs' amounts by adding compensatory outputs, when σ increases, the sizes of the c-decompositions obtained by the Boggart algorithm and the running time of the Boggart algorithm almost keeps stable. As shown in Figure 7(k), when σ increases, the compensatory ratio of our Boggart algorithm increases. The reason is when σ increases, the differences between original outputs' amounts get larger, and our Boggart algorithm needs more compensatory outputs.

E. Experiment Summary

Finally, in this subsection, we summarize our findings. Although when c is small, the approximation ratio of Boggart is large, it still performs well in real data sets. As shown in Figure 6, when c is 0.01, the size of c-decomposition obtained by Boggart is only 10^{-8} of the size of results obtained by the baseline algorithms. When c is too small, the transaction size will be very large and the transaction fee is very high. Thus, owe to the budget of users' transaction fees and the limited size of each block, in practice, c is not very small. In the Wasabi mixing transaction data sets detected by [2] from Bitcoin network, an output is mixed with another at most 99 decomposed outputs with the same amounts, which is equal to $c = 0.01$. Thus, our Boggart can perform well in real applications. Besides, the running time of our Boggart approach is small, which is only several microseconds. In addition, the performance of our Boggart approach is not affected by the amounts of original outputs, which changes dramatically in practice. Thus, our Boggart approach is robust for real-world applications. Moreover, the compensatory ratios of our Boggart approach are less than the naive solution, which adds $\frac{1}{c}$ times of original outputs to make c-decompositions. In addition, the amounts of compensatory outputs are acceptable. In [2], the authors calculate that in August 2019, the profit of Wasabi is at least 16 BTC, which is enough to support the compensatory outputs that our Boggart algorithm needs.

VII. Related Work

Currently, with the population of cryptocurrencies, mixing services attract much attention. Researchers have conducted many works on developing mixing services. Researchers propose many centralized mixing services [3], [25] and decen-



(a) Size of varying n (b) Size of varying μ (c) Size of varying $\frac{\sigma}{\mu}$ (d) Size of varying c

(e) Time of varying n (f) Time of varying μ (g) Time of varying $\frac{\sigma}{\mu}$ (h) Time of varying c

(i) Compensatory ratio of varying n (j) Compensatory ratio of varying μ (k) Compensatory ratio of varying $\frac{\sigma}{\mu}$ (l) Compensatory ratio of varying c

Fig. 7. Results of varying n, μ, $\frac{\sigma}{\mu}$, and c (Synthetic).

tralized mixing protocols [26]–[29]. However, these works focus on the security issues, and their decomposition solutions are naive. There are two common decomposition approaches in these works. The first approach is to ask users to set the values of decomposed outputs themselves, like Wasabi [30]. As we proved in Theorem III.1, the AA-OD problem is NP-hard. Users usually cannot set the values of decomposed outputs intelligently. The second approach (e.g., Dash [31]) is to decompose original outputs into standard denominations as the baseline solution DG Algorithm does in Section VI. As we illustrated in Section VI, this solution will generate massive decomposed outputs when making a c-decomposition, which increases users' transaction fees. Moreover, since the running time is related to the number of candidate denominations, the time complexity of the solution is related to the values of original outputs. Thus, this solution is a pseudo-polynomial-time algorithm. Thus, our work is necessary. As we evaluated in Section VI, our solutions are much better than the baselines.

There is another recent work studying the decomposition approach [32]. It decomposes original outputs' amounts by the difference between the inputs' amounts. This approach can increase the difficulty of inferring the possible original cases of transaction. However, by this approach, the amount of a decomposed output may be unique, and when adversaries have some background knowledge, the linkages between receivers and senders will be revealed. In addition, their approach does not minimize the number of decomposed outputs to save users' transaction fees. In other words, this approach cannot generate c-decompositions with minimized number of decomposed outputs. Thus, our work is novel and valuable.

VIII. Conclusion

In this paper, we target proposing an efficient anonymity-aware output decomposing solution for mixing services on blockchains. Specifically, we propose a novel anonymity concept, namely c-decomposition. We prove the privacy-preserving effect of a c-decomposition. Besides, we define the anonymity-aware output decomposition (AA-OD) problem and prove its NP-hardness. To solve the AA-OD problem, we propose an algorithm, namely Boggart, whose approximation ratio is $\frac{2}{c}+3$. By the Boggart algorithm, we can efficiently mix transactions with arbitrary values on blockchains with a guaranteed privacy-preserving effect.

References

[1] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[2] L. Wu, Y. Hu, Y. Zhou, H. Wang, X. Luo, Z. Wang, F. Zhang, and K. Ren, "Towards understanding and demystifying bitcoin mixing services," in *Proceedings of the Web Conference 2021*, pp. 33–44, 2021.

[3] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *International Conference on Financial Cryptography and Data Security*, pp. 486–504, Springer, 2014.

[4] "[online] Wasabi Wallet." https://wasabiwallet.io/, 2017.

[5] "[online] Bitmix." https://bitmix.biz, 2017.

[6] J. Vornberger, "Marker addresses: Adding identification information to bitcoin transactions to leverage existing trust relationships," *INFORMATIK 2012*, 2012.

[7] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and privacy in social networks*, pp. 197–223, Springer, 2013.

[8] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *International Conference on Financial Cryptography and Data Security*, pp. 6–24, Springer, 2013.

[9] 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," in *European Symposium on Research in Computer Security*, pp. 456–474, Springer, 2017.

[10] W. Ni, P. Cheng, L. Chen, and X. Lin, "When the recursive diversity anonymity meets the ring signature," in *Proceedings of the 2021 International Conference on Management of Data*, pp. 1359–1371, 2021.

[11] E. Duffield and D. Diaz, "Dash: A privacycentric cryptocurrency," 2015.

[12] "[online] Understanding bitcoin transaction fee per byte." https://metamug.com/article/security/bitcoin-transaction-fee-satoshi-per-byte.html, 2021.

[13] B. C. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys (Csur)*, vol. 42, no. 4, pp. 1–53, 2010.

[14] B. C. Fung, *PRIVACY-PRESERVING DATA PUBLISHING*. PhD thesis, SIMON FRASER UNIVERSITY, 2007.

[15] "[online] Bitcoin Fog." https://bitcoinfog.info/, 2017.

[16] "[online] Bitcoin Official. Choose Your Wallet.." https://bitcoin.org/en/choose-your-wallet?step=5&platform=windows, 2017.

[17] N. Li, "Research on diffie-hellman key exchange protocol," in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 4, pp. V4–634, IEEE, 2010.

[18] "[online] Monero." https://www.getmonero.org/, 2017.

[19] T. Ruffing and P. Moreno-Sanchez, "Mixing confidential transactions: Comprehensive transaction privacy for bitcoin, 2017."

[20] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2006.

[21] "[online] Technical Report." cspcheng.github.io/pdf/Boggart.pdf, 2017.

[22] J. Pakki, Y. Shoshitaishvili, R. Wang, T. Bao, and A. Doupé, "Everything you ever wanted to know about bitcoin mixers (but were afraid to ask)," in *International Conference on Financial Cryptography and Data Security*, pp. 117–146, Springer, 2021.

[23] A. Miller, M. Möser, K. Lee, and A. Narayanan, "An empirical analysis of linkability in the monero blockchain," *arXiv preprint arXiv:1704.04299*, 2017.

[24] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, *et al.*, "An empirical analysis of traceability in the monero blockchain," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 143–163, 2018.

[25] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *International Conference on Financial Cryptography and Data Security*, pp. 112–126, Springer, 2015.

[26] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *European Symposium on Research in Computer Security*, pp. 345–364, Springer, 2014.

[27] J. H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, and K. Wehrle, "Secure and anonymous decentralized bitcoin mixing," *Future Generation Computer Systems*, vol. 80, pp. 448–466, 2018.

[28] M. Xu, C. Yuan, X. Si, G. Yu, J. Fu, and F. Gao, "Coinmingle: A decentralized coin mixing scheme with a mutual recognition delegation strategy," in *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, pp. 160–166, IEEE, 2018.

[29] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, "Coinparty: Secure multi-party mixing of bitcoins," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pp. 75–86, 2015.

[30] "[online] Wasabi Docs." https://docs.wasabiwallet.io/using-wasabi/CoinJoin.html#input-registration, visited 2021.

[31] "[online] Dash's features." https://docs.dash.org/en/stable/introduction/features.html, visited 2021.

[32] F. K. Maurer, T. Neudecker, and M. Florian, "Anonymous coinjoin transactions with arbitrary values," in *2017 IEEE Trustcom/BigDataSE/ICESS*, pp. 522–529, IEEE, 2017.