A Faster Cryptographer's Conspiracy Santa[⋆]

Xavier Bultel[a], Jannik Dreier[b], Jean-Guillaume Dumas[c], Pascal Lafourcade[d]

[a] *Université d'Orléans, LIFO, Institut National des Sciences Appliquées Centre Val de Loire, 88 boulevard Lahitolle CS 60013 - 18022 Bourges, cedex, France*
[b] *Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France*
[c] *Université Grenoble Alpes, Laboratoire Jean Kuntzmann, UMR CNRS 5224, 700 avenue centrale, IMAG - CS 40700, 38058 Grenoble, cedex 9, France*
[d] *University Clermont Auvergne, LIMOS, CNRS UMR 6158, Campus Universitaire des Cézeaux, 1 rue de la Chebarde, 63170 Aubière, France*

Abstract

In Conspiracy Santa, a variant of Secret Santa, a group of people offer each other Christmas gifts, where each member of the group receives a gift from the other members of the group. To that end, the members of the group form conspiracies, to decide on appropriate gifts, and usually divide the cost of each gift among all participants of that conspiracy. This requires to settle the shared expenses per conspiracy, so Conspiracy Santa can actually be seen as an aggregation of several shared expenses problems.

First, we show that the problem of finding a minimal number of transaction when settling shared expenses is NP-complete. Still, there exist good greedy approximations. Second, we present a greedy distributed secure solution to Conspiracy Santa. This solution allows a group of n people to share the expenses for the gifts in such a way that no participant learns the price of his gift, but at the same time notably reduces the number of transactions to $2 \cdot n + 1$ with respect to a naïve aggregation of $n \cdot (n-2)$. Furthermore, our solution does not require a trusted third party, and can either be implemented physically (the participants are in the same room and exchange money using envelopes) or, over Internet, using a cryptocurrency.

Keywords: Sharing expenses, Conspiracy Santa, Secret Santa, Secure Multi-Party Computation, Cryptocurrency, Physical Cryptography, Privacy-preserving Protocols, Formal Security Models

[⋆]This research was conducted with the support of the FEDER program of 2014-2020, the region council of Auvergne-Rhône-Alpes, the support of the "Digital Trust" Chair from the University of Auvergne Foundation, the Indo-French Centre for the Promotion of Advanced Research (IFCPAR), the Center Franco-Indien Pour La Promotion De La Recherche Avancée (CEFIPRA) through the project DST/CNRS 2015-03 under DST-INRIA-CNRS Targeted Programme, and the OpenDreamKit Horizon 2020 European Research Infrastructures project (#676541).

1. Introduction

Secret Santa is a Christmas tradition, where members of a group are randomly assigned to another person, to whom they have to offer a gift. The identity of the person offering the present is usually secret, as well as the price of the present. Moreover, the participants often determine a common bound for the gift's prices.

In *Conspiracy Santa*, a variant of Secret Santa, for each participant, the other members of the group collude and jointly decide on an appropriate gift. The gift is then usually bought by one of the colluding participants, and the expenses are shared among the colluding participants.

In this setting, the price of the gift must remain secret and, potentially, also who bought the present. At the same time, sharing the expenses usually results in numerous transactions. Existing results in the literature (e.g., [3, 4, 5, 13]) aim at minimizing the number of transactions, but they assume that all expenses are public, that all participants are honest, and that communications are safe. Our goal is instead to propose a secure Conspiracy Santa algorithm for paranoid cryptographers that do not want to disclose the prices. Further, they want to keep their privacy at all cost, so they might rely on external third parties but only in case they do not need to trust them.

1.1. Contributions

Our results can be split into the following 3 contributions:

- We show that the general problem of finding a solution with a minimal number of transactions when sharing expenses (Shared Expenses Problem, or SEP) is NP-complete.
- We provide secure protocols for Conspiracy Santa for arbitrary n participants. The algorithms ensure that no participant learns the price of his gift, nor who bought it. Moreover, the algorithms reduce the number of transactions necessary to $3 \cdot n$ or $2 \cdot n + 1$ (depending on the largest authorized amount for a given transaction) compared to a naïve solution of $n \cdot (n - 2)$.
- Our secure algorithms are entirely distributed and do not require any trusted third party. To also realize the payments in a distributed fashion, a secure peer-to-peer cryptocurrency can be used. Additionally, we present a physical payment solution where all participants need to be in the same place, using envelopes and bank notes.

Our algorithms can also be used in the case where expenses are shared within multiple groups. There, some people belong to several of these groups and the goal is to reduce the number of transactions while still ensuring privacy: all participants only learn about the expenses of their groups, not the other groups. One can also see this problem as a variant of the dining cryptographers [8]. However, instead of respecting the cryptographers' right to anonymously invite everybody, we here want to respect the cryptographers' right to privately share expenses of multiple diners with different groups.

Table 1 summarizes the number of transactions, as well as the order of magnitude of the largest amount per transaction, required for the different algorithms considered in order to realize a conspiracy Santa.

Algorithm	Peer-to-peer	Transactions	Largest amount per transaction
n instances of SEP	✔	$n \cdot (n-2)$	constant
With a trusted third-party	✘	n	constant
Here (Protocols 1, 2, 3)	✔	$3 \cdot n$	constant
Here (Protocols 1, 6, 7)	✔	$2 \cdot n + 1$	linear

Table 1: Number of transactions for Conspiracy Santa

With respect to the conference version of this paper [7], we here provide the faster algorithm requiring only $2 \cdot n + 1$ transactions, instead of $3 \cdot n$. With respect to an upper bound B on the price of any gift, in this faster protocol, the amount of each transaction can then grow up to $n \cdot B$, where it stayed below B in our other protocol. We also provide a physical variant as well as complexity and security proofs for this novel protocol.

1.2. Outline

The remainder of the paper is structured as follows: in Section 2, we analyze the complexity of the general problem of sharing expenses. In Section 3, we present our protocol to solve the problem of privately sharing expenses in Conspiracy Santa, in a peer-to-peer setting. We also discuss further applications of our solution, and how to realize the anonymous payments required by the algorithm, either physically or online. In Section 4, we then present our second algorithm, with less transactions. We finally conclude in Section 5.

2. The Shared Expenses Problem and its Complexity

Before analyzing the Conspiracy Santa problem in more detail, we first discuss the problem of settling shared expenses with a minimal number of transactions. This problem frequently arises, for example when a group of security researchers attends a FUN conference and wants to share common expenses such as taxis, restaurants etc. Reducing the overall number of transactions might then reduce the overall currency exchange fees paid by the researchers.

In such a case, each participant covers some of the common expenses, and in the end of the conference, some transactions are necessary to ensure that all participants payed the same amount. Note for this first example, there are no privacy constraints, as all amounts are public.

Example 1. *Jannik, Jean-Guillaume, and Pascal attended FUN'16. The first night, Jannik payed the restaurant for* 155 €*, and Jean-Guillaume the drinks at* *the bar for* 52 €. *The second day Pascal payed the restaurant and drinks for a total of* 213 €.

The total sum is then $155 + 52 + 213 = 420$ €, *meaning* 140 € *per person. This means that Jannik payed* $140 - 155 = -15$ € *too much, Jean-Guillaume needs to pay* $140 - 52 = 88$ € *more, and Pascal has to receive* $140 - 213 = -73$ €. *In this case, the optimal solution uses two transactions: Jean-Guillaume gives* 15 € *to Jannik, and* 73 € *to Pascal.*

There are numerous applications implementing solutions to this problem (e.g., [3, 4, 5]), but it is unclear how they compute the transactions. Moreover, in these applications all expenses are public, making them unsuitable for Conspiracy Santa.

David Vávra wrote a master's thesis [13] about a similar smartphone application that allows to settle expenses within a group. He discusses a greedy approximation algorithm (see below), and conjectures that the problem is $\mathcal{NP}$-complete, but without giving a formal proof. We start by formally defining the problem.

Definition 1. Shared Expenses Problem (SEP).

Input: Given a multiset of values $K = \{k_1, \ldots, k_n\}$ such that $\sum_{i=1}^{n} k_i = 0$, where a positive k_i means that participant i has to pay money, and a negative k_i means that i has to be reimbursed.

Question: Is there a way to do all reimbursements using (strictly) less than $n - 1$ transactions?

Note that there is always a solution using $n - 1$ transactions using a greedy approach: given the values in $K = \{k_1, \ldots, k_n\}$, let i be the index of the maximum value of K ($i = \arg\max_i(k_i)$) and let j be the index of the minimum value of K ($j = \arg\min_j(k_j)$), we use one transaction between i and j such that after the transaction either the participant i or j ends up at 0. I.e., if $|k_i| - |k_j| > 0$, then the participant j ends up at 0, otherwise the participant i ends up at 0. By then recursively applying the same procedure on the remaining $n - 1$ values, we can do all reimbursements. Overall, this greedy solution uses $n - 1$ transactions in the worst case.

It is easy to see that $SEP \in \mathcal{NP}$: guess a list of (less than $n-1$) transactions, and verify for each participant that in the end there are no debts or credits left.

We show that SEP is $\mathcal{NP}$-complete, for this we use a reduction from the *Subset Sum Problem* [11] which can be seen as a special case of the well known knapsack problem [10].

Definition 2. Subset Sum Problem (SSP)

Input: Given a multiset of values $K = \{k_1, \ldots, k_n\}$.

Question: Is there a subset $K' \subseteq K$ such that $\sum_{k' \in K'} k' = 0$?

The Subset Sum Problem is known to be $\mathcal{NP}$-complete (see, e.g., [9]).

Theorem 1. *The Shared Expenses Problem is $\mathcal{NP}$-complete.*

Proof. Consider the following reduction algorithm:

Given a Subset Sum Problem (SSP) instance, i.e., a multiset of values $K = \{k_1, \ldots, k_n\}$, compute $s = \sum_{k \in K} k$. If $s = 0$ then return yes, otherwise let $K' = K \cup \{-s\}$ and return the answer of an oracle for the Shared Expenses Problem for K'.

It is easy to see that the reduction is polynomial, as computing the sum is in $\mathcal{O}(n)$. We now need to show that the reduction is correct. We consider the two following cases:

- Suppose the answer to the SSP is yes, then there is a subset $K'' \subseteq K$ such that $\sum_{k \in K''} k = 0$. If $K'' = K$, then the check in the reduction is true, and the algorithm returns yes. If $K'' \neq K$, then we can balance the expenses in the sets K'' and $K' \setminus K''$ independently using the greedy algorithm explained above. This results in $|K''| - 1$ and $|K'| - |K''| - 1$ transactions respectively, for a total of $|K'| - |K''| - 1 + |K''| - 1 = |K'| - 2 < |K'| - 1$ transactions. Thus there is a way to do all reimbursements using strictly less than $|K'| - 1$ transactions, hence the answer will be yes.

- Suppose the answer to the SSP is no, then there is no subset $K'' \subseteq K$ such that $\sum_{k \in K''} k = 0$. This means that there is no subset $K_3 \subseteq K'$ such that the expenses within this set can be balanced independently of the other expenses. To see this, suppose it were possible to balance the expenses in K_3 independently, then we must have $\sum_{k \in K_3} k = 0$, contradicting the hypothesis that there is no such subset (note that w.l.o.g. $K_3 \subseteq K$, if it contains the added value one can simply choose $K' \setminus K_3$).

  Hence any way of balancing the expenses has to involve all n participants, but building a connected graph with n nodes requires at least $n - 1$ edges. Thus there cannot be a solution with less than $n - 1$ transactions, and the oracle will answer no.

□

3. Cryptographer's Conspiracy Santa

Consider now the problem of organizing Conspiracy Santa, where no participant shall learn the price of his gift. Obviously we cannot simply apply, e.g., the greedy algorithm explained above on all the expenses, as this would imply that everybody learns all the prices.

More formally, an instance of Conspiracy Santa with n participant consists of n shared expenses problem (sub-SEP), each with $n - 1$ participants and with non-empty intersections of the participants. In each sub-SEP, the $n - 1$ participants freely discuss, decide on a gift, its value v_i and who pays it; then agree that their share for this gift is $v_i/(n-1)$. Overall the share of each participant j is

$$\frac{\sum_{i=1,i\neq j}^{n} v_i}{n-1}.$$

A participants *balance* p_j is this share minus the values of the gifts he bought.

A simple solution would be to use a trusted third party, but most cryptographers are paranoid and do not like trusted third parties. A distributed solution would be to settle the expenses for each gift within the associated conspiracy group individually, but this then results in n instances of the problem, with $n-2$ transactions each (assuming that only one person bought the gift), for a total of $n \cdot (n-2)$ transactions.

Moreover, the problem becomes more complex if several groups with non-empty intersections want to minimize transactions all together while preserving the inter-group privacy.

Example 2. Example 1 continued. *For the same conference, FUN'16, Jannik, Jean-Guillaume and Xavier shared a taxi from the airport and Jean-Guillaume paid for a total of* 60€, *that is* 20€ *per person. There are two possibilities. Either Jannik and Xavier make two new transactions to reimburse Jean-Guillaume. Or, to minimize the overall number of transactions, they aggregate both accounts, i.e. those from Example 1 with those of the taxi ride. That is* $[-15, 88, -73, 0] + [20, -40, 0, 20] = [5, 48, -73, 20]$. *Overall Jannik thus gives* 5 € *to Pascal, Jean-Guillaume reduces his debt to Pascal to only* 48€ *and Xavier gives* 20 € *to Pascal. The security issue, in this second case, is that maybe Jannik and Jean-Guillaume did not want Xavier to know that they were having lunch with Pascal, nor that they had a debt of more than* 20 €, *etc.*

In the next part, we present our solution for the generalization of Conspiracy Santa as the aggregation of several shared expenses problems with non-empty intersections between the participants. This solution uses $3 \cdot n$ transactions, preserves privacy, and does not require a trusted third party.

3.1. A Distributed Solution using Cryptocurrencies

Assumptions.

1. We suppose that all participants know a **fixed upper bound** B for the value of any gift.

2. We suppose that each participant **balance is an integral number**. A simple solution for this is to express everything in cents (¢), and make users agree that shares could be unevenly distributed up to a difference of one cent.

3. We consider **semi-honest** participants in the sense that the participants follow *honestly* the protocol, but they try to exploit all intermediate information that they have received during the protocol to break privacy.

Apart from the setup, the protocol has 3 rounds, each one with n transactions, and one initialization phase.

Initialization Phase. In the setup phase, the participants learn the price of the gifts in which they participate and can therefore compute their overall balance, p_i. They also setup several anonymous addresses in a given public transaction cryptocurrency like Bitcoin [1], ZCash [6] or Monero [2].

Finally the participants create one anonymous address which is used as a piggy bank. They all have access to the secret key associated to that piggy bank address. For instance, they can exchange encrypted emails to share this secret key. Protocol 1 presents the details of this setup phase.

Protocol 1 SEP broadcast setup

```
Require: An upper bound $B$ on the value of any gift;
Require: All expenses.
Ensure: Each participant learns his balance $p_i$.
Ensure: Each participant creates 1 or several anonymous currency addresses.
Ensure: A shared anonymous currency address.
 1: One anonymous currency address is created and the associated secret key is
    shared among all participants.
 2: for each exchange group do
 3:     for each payment within the group do
 4:         broadcast the amount paid to all members of the group;
 5:     end for
 6:     for each participant in the group do
 7:         Sum all the paid amounts of all the participants;
 8:         Divide by the number of participants in the group;
 9:         This produces the in-group share by participant.
10:     end for
11: end for
12: for each overall participant do
13:     Add up all in-group shares;
14:     Subtract all own expenses to get $p_i$;
15:     if $p_i < 0$ then
16:         Create $\lfloor \frac{p_i}{B} \rfloor$ anonymous currency addresses.
17:     end if
18: end for
```

First Round. The idea is that the participants will round their debts or credits so that the different amounts become indistinguishable. For this, the participants perform transactions to adjust their balance to either 0, B or a negative multiple of B. The first participant randomly selects an initial value between 1 and B ¢, and sends it to the second participant. This transaction is realized via any private payment channel between the two participants. It can be a physical payment, a bank transfer, a cryptocurrency payment, . . . , as long as no other participant learns the transferred amount. Then the second participant adds his balance to the received amount modulo B, and forwards the money[1] to the next participant, and so on. The last participant also adds his balance and sends the resulting amount to the first participant. In the end, all participants obtain a balance of a multiple of B, and the random amount chosen by the first participant has hidden the exact amounts. The details are described in Protocol 2.

Protocol 2 Secure rounding to multiple of the bound

```
Require: An upper bound $B$ on the value of any gift;
Require: Each one of $n$ participants knows his balance $p_i$;
Require: $\sum_{i=1}^{n} p_i = 0$.
Ensure: Each one of $n$ participants has a new balance $p_i$, either 0, $B$ or a
    negative multiple of $B$;
Ensure: $\sum_{i=1}^{n} p_i = 0$;
Ensure: Each transaction is between 1 and $B$ ¢;
Ensure: The protocol is zero-knowledge.
 1: $P_1$: $t_1 \xleftarrow{\$} [1..B]$ uniformly sampled at random;
 2: $P_1$: $p_1 = p_1 - t_1$;
 3: $P_1$ sends $t_1$ ¢ to $P_2$;          ▷ Random transaction $1..B$ on a secure channel
 4: $P_2$: $p_2 = p_2 + t_1$;
 5: for $i = 2$ to $n - 1$ do
 6:     $P_i$: $t_i = p_i \mod B$;
 7:     $P_i$: if $t_i = 0$ then $t_i = t_i + B$; end if                ▷ $1 \leq t_i \leq B$
 8:     $P_i$: $p_i = p_i - t_i$;
 9:     $P_i$ sends $t_i$ ¢ to $P_{i+1}$;    ▷ Random transaction $1..B$ on a secure channel
10:     $P_{i+1}$: $p_{i+1} = p_{i+1} + t_i$;
11: end for
12: $P_n$: $t_n = p_n \mod B$;
13: $P_n$: if $t_n = 0$ then $t_n = t_n + B$; end if                   ▷ $1 \leq t_n \leq B$
14: $P_n$: $p_n = p_n - t_n$;
15: $P_n$ sends $t_n$ ¢ to $P_1$;          ▷ Random transaction $1..B$ on a secure channel
16: $P_1$: $p_1 = p_1 + t_n$;
```

Second Round. The second and third rounds of the protocol require anonymous payments, for which we use anonymous cryptocurrency addresses. These two rounds are presented in Protocol 3, where $parfor$ is a notation for parallel execution of a for command where the order is not important. In the second round, every participant makes one public transaction of B ¢ to the piggy bank.

Third Round. Each creditor recovers their assets via $\lfloor \frac{p_i}{B} \rfloor$ public transactions of B ¢ from the piggy bank. Note that if a participant needs to withdraw more than B ¢ he needs to perform several transactions. To ensure anonymity, he

[1]up to B, or such that its credit becomes a multiple of B.

Protocol 3 Peer-to-peer secure debt resolution

```
Require: An upper bound $B$ on the value of any gift;
Require: $n$ participants each with a balance $p_i$, either 0, $B$ or a negative multiple of $B$.
Ensure: All balances are zero;
Ensure: The protocol is zero-knowledge.
1: parfor $i = 1$ to $n$ do                                ▷ Everybody sends $B$ to the piggy bank
2:     $P_i$: $p_i$ -= $B$;
3:     $P_i$ sends $B$ ¢ to the shared anonymous address;     ▷ Public transaction of $B$
4: end parfor
5: parfor $i = 1$ to $n$ do
6:     if $p_i < 0$ then                                    ▷ Creditors recover their assets
7:         parfor $j = 1$ to $\frac{-p_i}{B}$ do
8:             $P_i$ makes the shared anonymous address pay $B$¢ to one of his own anonymous addresses;   ▷ Public transaction of $B$
9:         end parfor
10:        $P_i$: $p_i = 0$.
11:    end if
12: end parfor
```

needs to use a different anonymous address for each transaction. In the end, the account is empty and the number of transactions corresponds exactly to the number of initial transactions used to credit the piggy bank's account.

Theorem 2. *For n participants, Protocols 1, 2, 3 are correct and, apart from the setup, require $3 \cdot n$ transactions.*

Proof. Including the piggy bank, all the transactions are among participants, therefore the sum of all the debts and credits is invariant and zero. There remains to prove that in the end of the protocol all the debts and credits are also zero. The value of any gift is bounded by B, thus any initial debt for any gift is at most $B/(n-1)$. As participants participate to at most $n-1$ gifts, the largest debt is thus lower than B ¢. Then, during the first round, all participants, except P_1, round their credits or debts to multiples of B. But then, by the invariant, after the first round, the debt or credit of P_1 must also be a multiple of B. Furthermore, any debtor will thus either be at zero after the first round or at a debt of exactly B ¢. After the second round any debtor will then be either at zero or at a credit of exactly B ¢. Thus after the second round only the piggy bank has a debt. Since the piggy bank received exactly $n \cdot B$ ¢, exactly n transactions of B ¢ will make it zero and the invariant ensures that, after the third round, all the creditors must be zero too. □

Remark 1. It is important to use a cryptocurrency such as Bitcoin, Monero or ZCash in order to hide both the issuer and the receiver of each transaction in the third round. This ensures that nobody can identify the users.

Note that when using Bitcoin, users can potentially be tracked if the addresses are used for other transactions. Using Monero or Zcash can offer more privacy since the exchanged amount can also be anonymized. Moreover, to avoid leaking the fact that some persons need to withdraw $B¢$ multiple times, and are thus doing multiple transaction at the same time, all the withdrawals should be synchronized. If exact synchronization is difficult to achieve, one can decide on a common time interval, e.g., an hour, and all the transactions have to be done at random time points during this interval, independently, whether they are executed from the same or a different participant.

Example 3. *We now have a look at the algorithm for our example with Jannik, Jean-Guillaume, Pascal and Xavier. As in Example 2, the initial balance vector is* $[5, 48, -73, 20]$[2]*. They decide on an upper bound of* $B = 50$ € *(note that to provably* ensure *exactly* $3 \cdot n = 12$ *transactions they should take an upper bound larger than any expense, that is larger than* 213 €*, but* 50 *is sufficient for our example here). For the first round, Jannik randomly selects* $1 \leq t_1 = 12 \leq 50$ *and makes a first private transaction of* $t_1 = 12$ € *to Jean-Guillaume. Jean-Guillaume then makes a private transaction of* $t_2 = 12 + 48 \mod 50 = 10$ € *to Pascal; Pascal makes a private transaction of* $t_3 = 10 - 73 \mod 50 = 37$ € *to Xavier; who makes a private transaction of* $t_4 = 37 + 20 \mod 50 = 7$ € *to Jannik. All these transactions are represented in Figure 1. The balance vector is thus now* $[0, 50, -100, 50]$*, because for instance Jean-Guillaume had a balance of* 48 €*, received* 12 € *from Jannik and sends* 10 € *to Pascal, hence his new balance is* $48 + 12 - 10 = 50$ €*. Everybody sends* 50 € *to the piggy bank address, so that the balance vector becomes* $[-50, 0, -150, 0]$*. Finally there are four* 50 € *transactions, one to an address controlled by Jannik and three to (different) addresses controlled by Pascal. These two last rounds are illustrated in Figure 2. Note that we have exactly* $n = 4$ *transactions per round.*

3.2. Security Proof

We now provide a formal security proof for our protocol. We use the standard multi-party computations definition of security against semi-honest adversaries [12]. As stated above, we consider *semi-honest* adversaries in the sense that the entities run *honestly* the protocols, but they try to exploit all intermediate information that they have received during the protocol.

We start by formally defining the *indistinguishability* and the *view* of an entity.

Definition 3 (Indistinguishability). Let η be a security parameter and X_η and Y_η two distributions. We say that X_η and Y_η are *indistinguishable*, denoted $X_\eta \equiv Y_\eta$, if for every probabilistic distinguisher $\mathcal{D}$ we have:

$$\Pr[x \leftarrow X_\eta : 1 \leftarrow \mathcal{D}(x)] - \Pr[y \leftarrow Y_\eta : 1 \leftarrow \mathcal{D}(y)] = 0$$

[2]In our running example the values are integral numbers in euros, so for simplicity we continue in euros instead of cents as in the protocol description.

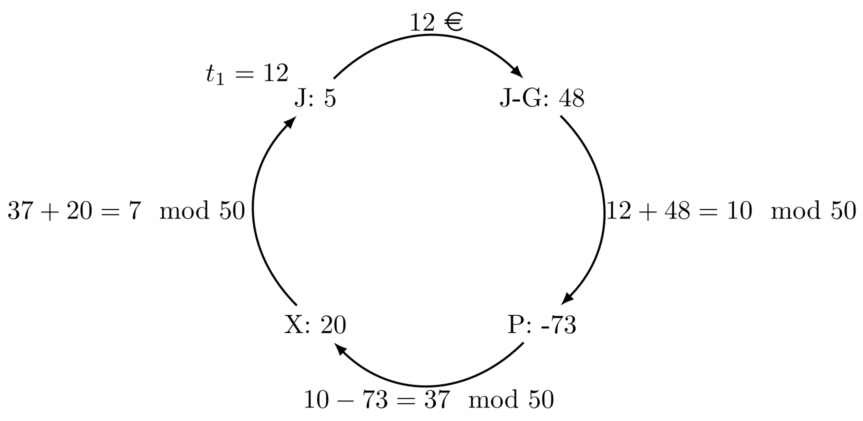

Figure 1: First round (private transactions) of Example 3 (starting at $t_1 = 12$, ending with $5 - 12 + 7 = 0 \mod 50$).

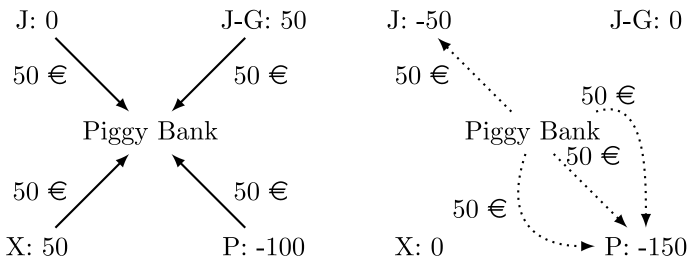

Figure 2: On the left: second round of Example 3. On the right: third round of Example 3. Dotted arrows represent anonymous transactions, in particular Pascal uses three different anonymous addresses.

Definition 4 (view of a party). Let $\pi(I)$ be an n-parties protocol for the entities $(P_i)_{1\leq i\leq n}$ using inputs $I = (I_i)_{1\leq i\leq n}$. The view of a party $P_i(I_i)$ (where $1 \leq i \leq n$) during an execution of π, denoted $\textsc{view}_{\pi(I)}(P_i(I_i))$, is the set of all values sent and received by P_i during the protocol.

To prove that a party P learns nothing during execution of the protocol, we show that P can run a *simulator* algorithm that simulates the protocol, such that P (or any polynomially bounded algorithm) is not able to differentiate an execution of the simulator and an execution of the real protocol. The idea is the following: since the entity P is able to generate his view using the simulator without the secret inputs of other entities, P cannot extract any information from his view during the protocol. This notion is formalized in Definition 5.

Definition 5 (security with respect to semi-honest behavior). Let $\pi(I)$ be an n-parties protocol between the entites $(P_i)_{1\leq i\leq n}$ using inputs $I = (I_i)_{1\leq i\leq n}$.

We say that π is *secure in the presence of semi-honest adversaries* if for each P_i (where $1 \leq i \leq n$) there exists a protocol $\mathsf{Sim}_i(I_i)$ where P_i interacts with a polynomial time algorithm $S_i(I_i)$ such that:

$$\text{VIEW}_{\mathsf{Sim}_i(I_i)}(P_i(I_i)) \equiv \text{VIEW}_{\pi(I)}(P_i(I_i))$$

Theorem 3. *Our conspiracy Santa protocol is secure with respect to semi-honest behavior.*

Proof. We denote our protocol by $\mathsf{SCS}_n(I)$ (for *Secure Conspiracy Santa*). For all $1 \leq i \leq n$, each entity P_i has the input $I_i = (n, B, p_i)$, where $I = (I_i)_{1 \leq i \leq n}$, p_i is the balance of each of the n participants. For all $1 \leq i \leq n$, we show how to build the protocol Sim_i such that:

$$\text{VIEW}_{\mathsf{Sim}_i(I_i)}(P_i(I_i)) \equiv \text{VIEW}_{\mathsf{SCS}_n(I)}(P_i(I_i))$$

Sim_1 is given in Simulator 4, and Sim_i for $1 < i \leq n$ is given in Simulator 5.

Simulator 4 Algorithm S_1 of the protocol $\mathsf{Sim}_1(I_1)$.

```
Require: $S_1$ knows $I_1 = (n, B, p_1)$
 1: $S_1$ receives $t_1$ ¢ from $P_1$;
 2: if $0 \leq (p_1 - t_1)$ then
 3:     $S_1$ sends $(B - (p_1 - t_1))$ ¢ to $P_1$;
 4: else if $(p_1 - t_1) < 0$ then
 5:     $S_1$ sends $(B - ((t_1 - p_1) \mod B))$ ¢ to $P_1$;
 6: end if
 7: for $j = 1$ to $n - 1$ do
 8:     $S_1$ sends $B$ ¢ to the shared anonymous address;
 9: end for
10: if $0 \leq (p_1 - t_1)$ then
11:     $x = n$;
12: else if $(p_1 - t_1) < 0$ then
13:     $x = n + \frac{(p_1-t_1)-((t_1-p_1) \mod B)}{B}$;
14: end if
15: for $j = 1$ to $x$ do
16:     $S_1$ makes the shared anonymous address pay $B$ ¢ to an anonymous
        address;
17: end for
```

We first show that the view of P_1 in the real protocol SCS_n is the same as in the protocol Sim_1:

- At Instruction 1 of Simulator 4, S_1 receives t_1 ¢ from P_1 such that $1 \leq t_1 \leq B$, as at Instruction 3 of Protocol 2.
- At Instruction 15 of Protocol 2, P_n sends t_n ¢ to P_1 such that:
    - $1 \leq t_n \leq B$

Simulator 5 Algorithm S_i of the protocol $\mathsf{Sim}_i(I_i)$, where $1 < i \leq n$.

```
Require: $S_i$ knows $I_1 = (n, B, p_i)$
1: $t_{i-1} \xleftarrow{\$} [1..B]$ ;
2: $S_i$ sends $t_{i-1}$ ¢ to $P_i$;
3: $S_i$ receives $t_i$ ¢ from $P_i$;
4: for $j = 1$ to $n - 1$ do
5:     $S_i$ sends $B$ ¢ to the shared anonymous address;
6: end for
7: $x = n + \frac{p_i + t_{i-1} - t_i - B}{B}$;
8: for $j = 1$ to $x$ do
9:     $S_i$ makes the shared anonymous address pay $B$ ¢ to an anonymous address;
10: end for
```

- The balance of P_1 is a multiple of B.

We show that these two conditions hold in the simulator. At Instruction 2 of Protocol 2, the balance of P_1 is $(p_1 - t_1)$.

1. If the balance is positive, then $0 \leq (p_1 - t_1) < B$ and S_1 sends $B - (p_1 - t_1)$ ¢ to P_1. We then have:
   - $1 \leq B - (p_1 - t_1) \leq B$
   - The balance of P_1 is $B - (p_1 - t_1) + (p_1 - t_1) = B$ which is multiple of B.
2. If the balance is negative, then S_1 sends $(B - ((t_1 - p_1) \mod B))$ ¢ to P_1. We then have:
   - $1 \leq B - ((t_1 - p_1) \mod B) \leq B$
   - The balance of P_1 is: $B - ((t_1 - p_1) \mod B) + (p_1 - t_1) = B + \left\lfloor \frac{p_1 - t_1}{B} \right\rfloor \cdot B = \left(\left\lfloor \frac{p_1 - t_1}{B} \right\rfloor + 1 \right) \cdot B$, which is a multiple of B.

- At Instruction 8 of Simulator 4, S_1 sends B ¢ to the shared anonymous address $(n-1)$ times, and P_1 sends B ¢ to the shared anonymous address 1 time, so together they send B ¢ n times to the shared anonymous address, as at Instruction 3 of Protocol 3.

- At Instruction 8 of Protocol 3, the users make the shared anonymous address pay B ¢ to n anonymous addresses. At Instruction 16 of Simulator 4, the balance of P_1 is:
  - 0 if $0 \leq (p_1 - t_1)$ (because P_1 had B ¢ and sent B ¢ to the shared address).
  - Otherwise, the balance of P_1 is $B - ((t_1 - p_1) \mod B) + (p_1 - t_1) - B = ((t_1 - p_1) \mod B) + (p_1 - t_1)$. Hence P_1 receives B ¢ from the shared anonymous address $\left| \frac{((t_1 - p_1) \mod B) + (p_1 - t_1)}{B} \right|$ times, and S_1 receives

B ¢ from the shared anonymous address $n + \frac{((t_1 - p_1) \mod B) + (p_1 - t_1)}{B}$ times. We note that $((t_1 - p_1) \mod B) + (p_1 - t_1) \leq 0$ because $(p_1 - t_1) \leq 0$ and $((t_1 - p_1) \mod B) \leq -(p_1 - t_1)$. Finally, P_1 and S_1 make the shared anonymous address pay B ¢ to n anonymous addresses because:

$$n + \frac{(t_1 - p_1) \mod B + (p_1 - t_1)}{B} + \left| \frac{(t_1 - p_1) \mod B + (p_1 - t_1)}{B} \right| = n$$

Finally, we deduce that the view of P_1 in the real protocol SCS_n is the the same as in the simulator Sim_1:

$$\text{VIEW}_{\mathsf{Sim}_1(I_1)}(P_1(I_1)) \equiv \text{VIEW}_{\mathsf{SCS}_n(I)}(P_1(I_1))$$

We then show that the view of P_i in the real protocol SCS_n is the same as in the protocol Sim_i for any $1 \leq i \leq n$:

- At instruction 3 and 9 of Protocol 2, each user P_i receives t_{i-1} ¢ from P_{i-1} for any $1 \leq i \leq n$ such that $1 \leq t_{i-1} \leq B$. We note that each t_{i-1} depends on the value t_1 chosen by P_1. Moreover, t_1 comes form a uniform distribution and acts as a one-time pad on the values t_{i-1}, i.e., it *randomizes* t_{i-1} such that P_i cannot distinguish whether t_{i-1} was correctly generated or comes from the uniform distribution on $\{1, \ldots, B\}$. At instruction 1 of Simulator 5, S_i chooses t_{i-1} at random in the uniform distribution on $\{1, \ldots, B\}$ and sends t_{i-1} to P_i.

- At Instruction 3 of Simulator 5, S_i receives t_i ¢ from P_i such that $1 \leq t_1 \leq B$, like at Instruction 9 of Protocol 2.

- At Instruction 5 of Simulator 5, S_i sends B ¢ to the shared anonymous address $(n-1)$ times, and P_i sends B ¢ to the shared anonymous address 1 time, so together they send B ¢ n times to the shared anonymous address, as at Instruction 3 of Protocol 3.

- At Instruction 8 of Protocol 3, the users make the shared anonymous address pay B ¢ to n anonymous addresses. At Instruction 9 of Simulator 5, the balance of P_i is $p_i + t_{i-1} - t_i - B$. Hence P_i receives B ¢ from the shared anonymous address $\left|\frac{p_i + t_{i-1} - t_i - B}{B}\right|$ times, and S_i receives B ¢ from the shared anonymous address $n + \frac{p_i + t_{i-1} - t_i - B}{B}$ times. We note that $p_i + t_{i-1} - t_i - B \leq 0$; indeed, we have $t_i = (p_i + t_{i-1}) \mod B$ (Instruction 6 of Protocol 2). Since $p_i \leq B$ and $t_{i-1} \leq B$, then we have $(p_i + t_{i-1}) - t_i \leq B$, so we have $p_i + t_{i-1} - t_i - B \leq 0$. Finally, P_i and S_i make the shared anonymous address pay B ¢ to n anonymous addresses because:

$$n + \frac{p_i + t_{i-1} - t_i - B}{B} + \left| \frac{p_i + t_{i-1} - t_i - B}{B} \right| = n$$

Finally, to conclude the proof, we deduce that for all $1 \leq i \leq n$ the view of P_i in the real protocol SCS_n is the the same as in the simulator Sim_i:

$$\text{VIEW}_{\mathsf{Sim}_i(I_i)}(P_i(I_i)) \equiv \text{VIEW}_{\mathsf{SCS}_n(I)}(P_i(I_i)).$$

□

3.3. Physical Variant

If one does not wish to use cryptocurrencies, one can use the following physical variant of the protocol. In the first round each participant needs to transfer some money to another participant using a private channel. A simple physical solution is that they meet and perform the transfer face to face, while ensuring that nobody spies on them. For the second round, the balance of all participants is a multiple of B ¢. During the first part of this algorithm, everyone puts an envelope containing B ¢ onto a stack that is in a secure room. By *secure room*, we mean a place where no other participants can spy what is going on inside. In the second part all participants enter this secure room one after the other and do the following according to their balance:

- If the balance is 0 then the participant does nothing.
- If the balance is a multiple k of B ¢, the participant takes k envelopes from the top of the stack, opens them and collects the corresponding $k \cdot B$ ¢. Then he places, in each of the now empty k envelopes, a piece of paper that have the same shape and weight as a the B ¢. These envelopes are placed under the stack of envelopes.

This method allows everyone to collect his money without revealing to the other ones how much they have taken.

We show that this protocol is secure with respect to semi-honest behavior. For this, we physically simulate the protocol for any participant. We first note that the first round of the protocol is the same as Protocol 2, so this round can be simulated exactly as in the proof of Theorem 3. We simulate the second round for any participant as follows. During the first part of the algorithm, the simulator enters $n-1$ times the secure room and puts an envelope containing B ¢ onto the stack. When it is his turn, the participant enters the room and puts an envelope containing B ¢ onto the stack. Finally, there are n envelopes containing B ¢ on a stack. In the second part the simulator enters the room $n-1$ times and does nothing. When it is his turn, the participant enters the room and takes k envelopes from the top of the stack, opens them and collects the corresponding $k \cdot B$ ¢ as in the real protocol, where $0 \leq k \leq n$. Since each of the n envelopes contains B ¢, the simulation works for any $0 \leq k \leq n$.

We deduce that the view of the participant during the simulation is the same as during the real protocol, which implies that our physical protocol is secure with respect to semi-honest behavior.

Remark 2. This physical protocol mimics exactly the solution using cryptocurrencies. One advantage, though, of the physical world is that it is easier to perform transactions with 0 ¢. Therefore there exists a simpler solution for the second round, where creditors do not have to give B ¢ in advance: if the participant is in debt he puts an envelope containing B ¢ onto the stack, otherwise he puts an envelope containing a piece of paper under the stack.

The first and third rounds are not modified, and the simulator for the security proof is not modified either.

4. A Faster Protocol

To the price of having much larger transactions, there exist a protocol for conspiracy Santa that requires less transactions: $2 \cdot n + 1$ instead of $3 \cdot n$.

4.1. Merging the first two rounds

We now propose a variant of the cryptocurrency protocol where the initialization phase and the third round are unchanged but the first and second round are merged.

The idea is that the participants will round their debts or credits so that the different amounts become indistinguishable, and at the same time they will give B ¢ to the next player: this is what they would have given to the Piggy bank in the second round. At the end the first player will receive the amount needed to round his debt or credit as previously, *plus* $n - 1$ times B ¢. It is then sufficient for him to give $n \cdot B$ ¢ to the Piggy bank, for the third round to take place as previously. The details are described in Protocols 6 and 7.

Theorem 4. *For n participants, Protocols 1, 6, 7 are correct and, apart from the setup, require $2 \cdot n + 1$ transactions.*

Proof. Including the piggy bank, all the transactions are among participants, therefore the sum of all the debts and credits is invariant and zero. There remains to prove that in the end of the protocol all the debts and credits are also zero. Any initial debt for any gift is strictly lower than $B/(n-1)$ and the largest debt is thus strictly lower than B ¢, that is all integral debts are between 0 and $B - 1$. Then, during the first round, all participants, except P_1, still round their credits or debts to multiples of B: for $i \geq 2$, $p_i = (p_i - 1) - (p_i - 1 \mod B) - (i \cdot 1) \cdot B \equiv 0 \mod B$. Also, every participant sends more than he received: $1 + t_i + (i - 1) \cdot B < 1 + t_{i+1} + i \cdot B$, so any creditor remains creditor. Furthermore, any debtor will thus either be at zero after the first round or at a credit of exactly B ¢: any debtor $(i + 1) \neq 1$ has an initial debt $1 \leq p_{i+1} \leq B$; then he receives $1 + t_i + (i - 1) \cdot B$ so that his new debt is between $2 + (i - 1) \cdot B$ and $(i + 1) \cdot B$; finally he sends the residue mod B plus $i \cdot B$, so that his credit must be either B or 0. All the money sent is immediately added to the debt of the receiver and removed from that of the sender (including an aggregation for P_1 at the end, $(n - 1) \cdot B - n \cdot B = -B$) so, overall, including the piggy bank, the sum of all debts and credits remains zero. We just showed that except P_1 all the credits or debts are now multiples of B so that of P_1 must also be a multiple B. Further, if P_1 is a debtor then $1 \leq p_1 \leq B$. He sends $1 + t_1$, $n \cdot B$ and receives $1 + t_n + (n - 1) \cdot B$

Protocol 6 Secure rounding with an extra multiple

```
Require: An upper bound $B$ on the value of any gift;
Require: Each one of $n$ participants knows his integer balance $p_i$;
Require: $\sum_{i=1}^n p_i = 0$.
Ensure: Each one of $n$ participants has a new balance $p_i$, either 0, $B$ or a
    negative multiple of $B$;
Ensure: $\sum_{i=1}^n p_i = 0$;
Ensure: The credit of the piggy bank is $nB$ ¢;
Ensure: The protocol is zero-knowledge.
 1: $P_1$: $t_1 \xleftarrow{\$} [0..B-1]$ uniformly sampled at random;
 2: $P_1$: $p_1 = p_1 - 1 - t_1$;
 3: $P_1$ sends $1 + t_1$ ¢ to $P_2$;    ▷ Random transaction $1..B$ on a secure channel
 4: $P_2$: $p_2 = p_2 + 1 + t_1$;
 5: for $i = 2$ to $n-1$ do
 6:     $P_i$: $t_i = p_i - 1 \mod B$;                         ▷ $0 \le t_i \le B-1$
 7:     $P_i$: $p_i = p_i - 1 - t_i - (i-1) \cdot B$;
 8:     $P_i$ sends $(1 + t_i + (i-1) \cdot B)$ ¢ to $P_{i+1}$;
                        ▷ Transaction $(i-1) \cdot B + 1..i \cdot B$ on a secure channel
 9:     $P_{i+1}$: $p_{i+1} = p_{i+1} + 1 + t_i + (i-1) \cdot B$;
10: end for
11: $P_n$: $t_n = p_n - 1 \mod B$;                             ▷ $0 \le t_n \le B-1$
12: $P_n$: $p_n = p_n - 1 - t_n - (n-1) \cdot B$;
13: $P_n$ sends $(1 + t_n + (n-1) \cdot B)$ ¢ to $P_1$; ▷ Transaction $(n-1) \cdot B + 1..n \cdot B$
    on a secure channel
14: $P_1$ sends $n \cdot B$ ¢ to the piggy bank;            ▷ Public transaction of $nB$
15: $P_1$: $p_1 = p_1 + 1 + t_n - B$;
```

so that his new balance is $2 - 2 \cdot B \le p_1 - t_1 - n \cdot B + t_n + (n-1) \cdot B \le B - 1$. As we have shown that this balance must be a multiple of B, then either his balance is zero or he has a credit of B. Therefore, at the end of Protocol 6, only the piggy bank has a debt. Since the piggy bank received exactly $n \cdot B$ ¢, exactly n transactions of B ¢ will make it zero and the invariant ensures that, after Protocol 7, all the creditors must be zero too. □

Example 4. *We now have a look at the algorithm for our example with Jannik, Jean-Guillaume, Pascal and Xavier. As in Example 3, the initial balance vector is* $[5, 48, -73, 20]$ *and the upper bound is* $B = 50$ €*. For the first round, Jannik randomly selects* $0 \le t_1 = 11 < 50$ *and makes a first private transaction of* $1 + t_1 = 12$ € *to Jean-Guillaume. Jean-Guillaume then makes a private transaction of* $1 + (12 + 48 - 1 \mod 50) + 50 = 60$ € *to Pascal; Pascal makes a private transaction of* $1 + (60 - 73 - 1 \mod 50) + 2 \cdot 50 = 137$ € *to Xavier; who makes a private transaction of* $1 + (137 + 20 - 1 \mod 50) + 3 \cdot 50 = 157$ € *to Jannik and the balance vector is* $[150, 0, -150, 0]$ *Finally Jannik gives* 200 € *to the piggy bank so that the player's balance vector becomes* $[-50, 0, -150, 0]$*. Indeed, for instance, Jean-Guillaume had a balance of* 48 €*, received* 12 € *from Jannik*

Protocol 7 Secure recovering from the piggy bank

```
Require: An upper bound $B$ on the value of any gift;
Require: $n$ participants each with a balance $p_i$, either 0, or a negative multiple
    of $B$, their sum being $-nB$;
Require: The piggy bank has a credit of $nB$ ¢.
Ensure: All balances are zero;
Ensure: The protocol is zero-knowledge.
1: parfor $i = 1$ to $n$ do
2:     if $p_i < 0$ then                                  ▷ Creditors recover their assets
3:         parfor $j = 1$ to $\frac{-p_i}{B}$ do
4:             $P_i$ makes the shared anonymous address pay $B$¢ to one of his own
    anonymous addresses;                                  ▷ Public transaction of $B$
5:         end parfor
6:         $P_i$: $p_i = 0$.
7:     end if
8: end parfor
```

and sends 60 € *to Pascal, hence his new balance is* $48 + 12 - 60 = 0$ €*. All these transactions are represented in Figure 3. Finally there are four* 50 € *transactions, one to an address controlled by Jannik and three to (different) addresses controlled by Pascal. This last round is exactly the one illustrated in Figure 2, right. Note that we have* $n = 4$ *participants and exactly* $1 + (n - 2) + 1 + 1 = 5$ *transactions in Protocol 6 and* 4 *transactions in Protocol 7, hence the total is* $9 = 5 + 4 = 2 \cdot 4 + 1 = 2 \cdot n + 1$.

4.2. Security Proof of the Faster Variant

Theorem 5. *Our faster conspiracy Santa protocol is secure with respect to semi-honest behavior.*

Proof. We denote our protocol by $\mathsf{FSCF}_n(I)$ (for *Fast Secure Conspiracy Santa*). For all $1 \le i \le n$, each entity P_i has the input $I_i = (n, B, p_i)$, where $I = (I_i)_{1 \le i \le n}$. For all $1 \le i \le n$, we show how to build the protocol Sim'_i such that:

$$\text{VIEW}_{\mathsf{Sim}'_i(I_i)}(P_i(I_i)) \equiv \text{VIEW}_{\mathsf{FSCF}_n(I)}(P_i(I_i))$$

Sim'_1 is given in Simulator 8, and Sim'_i for $1 < i \le n$ is given in Simulator 9.

We first show that the view of P_1 in the real protocol FSCS_n is the same as in the protocol Sim'_1:

- At Instruction 1 of Simulator 8, S_1 receives $(1 + t_1)$ € from P_1 such that $0 \le t_1 \le B - 1$, as at Instruction 3 of Protocol 6.
- At Instruction 13 of Protocol 6, P_n sends $(1 + t_n + (n - 1) \cdot B)$ € to P_1 such that:
    - $0 \le t_n \le B$

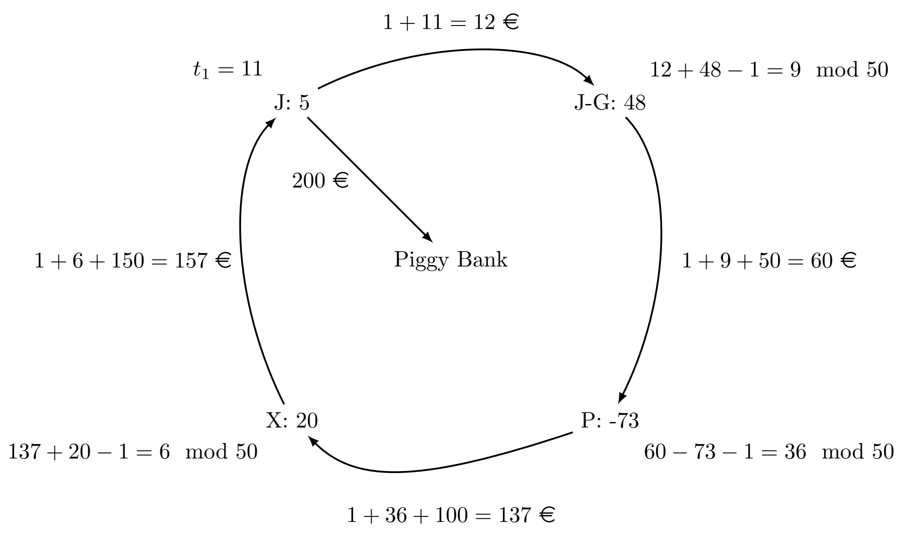

Figure 3: Merged round of Example 4 (starting at $t_1 = 11$, ending with $5 - 12 + 157 = 0 \mod 50$, and a final transaction of 200 € to the Piggy bank).

  – The balance of P_1 is a multiple of B.

  We show that these two conditions hold in the simulator. At Instruction 1 of Protocol 6, the balance of P_1 is $(p_1 - 1 - t_1)$.

  1. If $(p_1 - t_1) \geq 1$, then $1 \leq (p_1 - t_1) \leq B$ and S_1 sends $1 + (B - (p_1 - t_1)) + (n-1) \cdot B$ € to P_1. We then have:
     – $0 \leq B - (p_1 - t_1) \leq B - 1$
     – At Instruction 3 of Simulator 8, The balance of P_1 is $p_1 - 1 - t_1 + 1 + (B - (p_1 - t_1) + (n-1) \cdot B = n \cdot B$, as at Instruction 13 of Protocol 6, which is multiple of B.
  2. If $(p_1 - t_1) \leq 0$, then S_1 sends $1 + ((t_1 - p_1) \mod B) + (n-1) \cdot B$ € to P_1. We then have:
     – $0 \leq ((t_1 - p_1) \mod B) \leq B - 1$
     – At Instruction 5 of Simulator 8, The balance of P_1 is: $p_1 - 1 - t_1 + 1 + ((t_1 - p_1) \mod B) + (n-1) \cdot B = (p_1 - t_1) + ((t_1 - p_1) \mod B) + (n-1) \cdot B = \lceil \frac{p_1 - t_1}{B} \rceil \cdot B + (n-1) \cdot B$, as at Instruction 13 of Protocol 6, which is a multiple of B.

- P_1 sends $n \cdot B$ € to the shared anonymous address, so his balance is a multiple of B and is at most 0, as at the end of Protocol 6.

- At Instruction 1 of Protocol 7, the users make the shared anonymous address pay B € to n anonymous addresses. At Instruction 13 of Simulator 8, the balance of P_1 is:

Simulator 8 Algorithm S_1 of the protocol $\mathsf{Sim}'_1(I_1)$.

```
Require: $S_1$ knows $I_1 = (n, B, p_1)$
1:  $S_1$ receives $(1 + t_1)$ € from $P_1$;
2:  if $1 \leq (p_1 - t_1)$ then
3:      $S_1$ sends $(1 + B - (p_1 - t_1) + (n - 1) \cdot B))$ € to $P_1$;)
4:  else if $(p_1 - t_1) \leq 0$ then
5:      $S_1$ sends $(1 + ((t_1 - p_1) \mod B) + (n - 1) \cdot B))$ € to $P_1$;
6:  end if
7:  if $1 \leq (p_1 - t_1)$ then
8:      $x = n$;
9:  else if $(p_1 - t_1) \leq 0$ then
10:     $x = n + \frac{(p_1 - t_1) + ((t_1 - p_1) \mod B) - B}{B}$;
11: end if
12: for $j = 1$ to $x$ do
13:     $S_1$ makes the shared anonymous address pay $B$ € to an anonymous address;
14: end for
```

Simulator 9 Algorithm S_i of the protocol $\mathsf{Sim}'_i(I_i)$, where $1 < i \leq n$.

```
Require: $S_i$ knows $I_1 = (n, B, p_i)$
1: $t_{i-1} \xleftarrow{\$} [0..B - 1]$ ;
2: $S_i$ sends $(1 + t_{i-1} + (i - 2) \cdot B)$ € to $P_i$;
3: $S_i$ receives $(1 + t_i + (i - 1) \cdot B)$ € from $P_i$;
4: $S_i$ sends $n \cdot B$ € to the shared anonymous address;
5: $x = n + \frac{p_i + t_{i-1} - t_i - B}{B}$;
6: for $j = 1$ to $x$ do
7:     $S_i$ makes the shared anonymous address pay $B$ € to an anonymous address;
8: end for
```

- 0 if $1 \leq (p_1 - t_1)$ (because P_1 had $n \cdot B$ € and sent $n \cdot B$ € to the shared address).
- Otherwise, the balance of P_1 is $\left\lceil \frac{p_1 - t_1}{B} \right\rceil \cdot B - B$. Hence P_1 receives B € from the shared anonymous address $x = n + \frac{(p_1 - t_1) + ((t_1 - p_1) \mod B) - B}{B}$ $= n + \left\lceil \frac{p_1 - t_1}{B} \right\rceil - 1$ times, and S_1 receives B € from the shared anonymous address $n - x$ times, as in Instruction 3 of Protocol 7. Finally, P_1 and S_1 make the shared anonymous address pay B € to n anonymous addresses.

Finally, we deduce that the view of P_1 in the real protocol FSCS_n is the the same as in the simulator Sim'_1:

$$\text{VIEW}_{\mathsf{Sim}'_1(I_1)}(P_1(I_1)) \equiv \text{VIEW}_{\mathsf{FSCS}_n(I)}(P_1(I_1))$$

We then show that the view of P_i in the real protocol FSCS_n is the same as in the protocol Sim'_i for any $1 \leq i \leq n$:

- At instruction 3, 8 and 13 of Protocol 6, each user P_i receives $(1+t_{i-1}+(i-2)\cdot B)$ € from P_{i-1} for any $1 \leq i \leq n$ such that $0 \leq t_{i-1} \leq B-1$. As in Sim_i (see proof of Theorem 3) P_i cannot distinguish whether t_{i-1} was correctly generated or comes from the uniform distribution on $\{0, \dots, B-1\}$.
- At Instruction 3 of Simulator 9, S_i receives $(1+t_i+(i-1)\cdot B)$ € from P_i such that $0 \leq t_1 \leq B-1$, like at Instruction 3 of Protocol 6.
- At Instruction 4 of Simulator 9, S_i sends $n \cdot B$ € to the shared anonymous address, as P_1 at Instruction 14 of Protocol 6.
- At Instruction 1 of Protocol 7, the users make the shared anonymous address pay B € to n anonymous addresses. At Instruction 7 of Simulator 9, the balance of P_i is $p_i+1+t_{i-1}+(i-2)\cdot B-1-t_i-(i-1)\cdot B = p_i+t_{i-1}-t_i-B$. Hence, setting $x = n + \frac{p_i+t_{i-1}-t_i-B}{B}$, P_i receives B € from the shared anonymous address $n-x = -\frac{p_i+t_{i-1}-t_i-B}{B}$ times, and S_i receives B € from the shared anonymous address x times. Finally, P_i and S_i make the shared anonymous address pay B € to n anonymous addresses, as in Sim'_i.

Finally, to conclude the proof, we deduce that for all $1 \leq i \leq n$ the view of P_i in the real protocol FSCS_n is the the same as in the simulator Sim'_i:

$$\text{VIEW}_{\mathsf{Sim}'_i(I_i)}(P_i(I_i)) \equiv \text{VIEW}_{\mathsf{FSCS}_n(I)}(P_i(I_i)).$$

□

4.3. Physical Variant

Similarly, one can adapt the physical variant as follows. For Protocol 6, they perform a face to face transfer. At the end, the first player puts n envelopes containing B ¢ onto a stack in the secure room. The last part is unchanged, all participants enter this secure room one after the other and either does nothing or takes as many envelopes as needed to zero out his credit.

The security proof of this physical variant also uses the simulator of Theorem 5 for Protocol 6 and that of Section 3.3 for Protocol 7.

Remark 3. Note that in the physical word, in some sense only $2 \cdot n$ transactions are required: n face to face transfers and n trips to the secure room.

5. Conclusion

In this paper we showed that the Shared Expenses Problem (SEP) is $\mathcal{NP}$-complete. Moreover, we devised two privacy-preserving protocols to share expenses in a Conspiracy Santa setting where members of a group offer each other gifts.

Our protocols ensure that no participant learns the price of his gift, while reducing the number of transactions compared to a naive solution, and not relying on a trusted third party. We formally prove the security of our protocol and propose two variants, one relying on cryptocurrencies for anonymous payments, the other one using physical means, such as envelopes, to achieve anonymous payments.

Our protocol can also be used to share expenses among different groups with non-empty intersections, while still ensuring that each participant only learns the expenses of his group(s).

The next step is to design a practical implementation using Paypal or a cryptocurrency. We would like to both have a platform on a website and an application on smartphone, which however requires to solve synchronization problems between them when not using a central server (which was one of the design goals!). There, we also would need also to fix random time points during an allowed interval for transactions, set up some ciphered communications (at setup), etc.

Finally, another avenue of research to explore could be the determination of a lower bound on the number of transactions in a secure peer-to-peer setting.

Acknowledgements

Many thanks to Marie-Béatrice, Anne-Catherine, Marc, Jacques and Luc for such great conspiracy Santas! A big thanks also to the Gilbert family for having big instances of the Shared Expenses Problem problem regularly, and to Cyprien for asking the question of its complexity. More thanks go to Mathilde and Gwénaël for the discussions on efficient algorithms.

References

References

[1] Bitcoin. https://bitcoin.org/. Accessed: 2018-02-13.

[2] Monero. https://getmonero.org/. Accessed: 2018-02-13.

[3] Settle up. https://settleup.io/. Accessed: 2018-02-13.

[4] Splitwise. https://www.splitwise.com/. Accessed: 2018-02-13.

[5] Tricount. https://www.tricount.com/. Accessed: 2018-02-13.

[6] Zcash. https://z.cash/. Accessed: 2018-02-13.

[7] Xavier Bultel, Jannik Dreier, Jean-Guillaume Dumas, and Pascal Lafourcade. A cryptographer's conspiracy Santa. In Hiro Ito, Stefano Leonardi, Linda Pagli, and Giuseppe Prencipe, editors, *9th International conference on Fun with algorithms (FUN 2018), Maddalena, Italy*, volume 100 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:13, Dagstuhl, Germany, June 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: https://hal.archives-ouvertes.fr/hal-01777997, doi:10.4230/LIPIcs.FUN.2018.13.

[8] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, Jan 1988. doi:10.1007/BF00206326.

[9] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] Richard M. Karp. Reducibility among combinatorial problems. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 219–241. Springer, Berlin, Heidelberg, 2010.

[12] Qingkai Ma and Ping Deng. Secure multi-party protocols for privacy preserving data mining. In Yingshu Li, Dung T. Huynh, Sajal K. Das, and Ding-Zhu Du, editors, *Wireless Algorithms, Systems, and Applications*, pages 526–537, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-88582-5_49.

[13] David Vávra. Mobile Application for Group Expenses and Its Deployment. Master's thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Graphics and Interaction, 2012.