Acta Cybernetica **24** (2019) 29–41.

Multi Party Computation Motivated by the Birthday Problem*

Péter Hudoba[a] and Péter Burcsi[b]

Abstract

Suppose there are n people in a classroom and we want to decide if there are two of them who were born on the same day of the year. The well-known birthday paradox is concerned with the probability of this event and is discussed in many textbooks on probability. In this paper we focus on cryptographic aspects of the problem: how can we decide if there is a collision of birthdays without the participants disclosing their respective date of birth. We propose several procedures for solving this generally in a privacy-preserving way and compare them according to their computational and communication complexity.

Keywords: secure multi-party computation, birthday paradox, privacy-preserving, communication complexity

1 Introduction

1.1 Description of the problem

The birthday paradox or birthday problem [14, 18, 1] investigates the following question: n people are selected at random from a large population. What is the probability that at least r people share the same birthday? It's usually referred to as a paradox because of the unintuitively large probability of over 50% already for the relatively small value of $n = 23$ and $r = 2$.

In the present paper we focus on cryptographic aspects of the problem. We examine whether and how the n participants can decide if r of them share the same date of birth without any of them publicly announcing his or her birthday,

*Péter Hudoba's was work supported by EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies. The Project is supported by the Hungarian Government and co-financed by the European Social Fund. Péter Burcsi's work was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

[a]Eötvös Loránd University, Budapest, Hungary, E-mail: peter.hudoba@inf.elte.hu

[b]ELTE 3in External Research Group, E-mail: bupe@inf.elte.hu

DOI: 10.14232/actacyb.24.1.2019.4

using secure communication. This is a so-called multi-party computation, see e.g. Chapter 7 of [9] or [10].

The $n = 2$ case is well-known and named Tiercé or socialist millionaires' problem. This is similar to Yao's millionaires' problem originally introduced in [19, 20] where the two participants want to compare their secrets (decide which one is larger). Later, other solutions were proposed, e.g. [5],[13],[16], [15] but all of them consider the case of 2 participants.

A first idea would be to use pairwise socialist millionaires' protocols for the general n-participant version. However, in case of equality the two participants involved would instantly learn each other's secrets which we want to avoid when $n \geq 3$. In what follows, we deal with the $r = 2$ case but general n.

More generally and formally we have a finite but possibly large set of possible values V (corresponding to possible birthdays) and each of n participants holding a secret value $x_i \in V$ (their respective birthdays). We want to compute, using a secure multi-party computation, the following function:

$$f(x_1, \ldots, x_n) = \begin{cases} 1, & \text{if } \exists i, j \in \{1, \ldots, n\} : i \neq j \wedge x_i = x_j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

1.2 Security assumptions and comparison of protocols

In this paper we make the assumption on the participants' behavior called *honest but courious* or semi-honest. The participants are honest in following the protocol which means they do not poison or dilate the data, but if they can gain information without poisoning the algorithm, they will do it. With these conditions we want to make sure no one learns any other participant's secret.

We can characterize the level of privacy of a secure multi-party scheme with numbers adv_a (respectively adv_p) corresponding to the minimal number of active (resp. passive) coordinated adversary participants who are able to gain access to secrets of the others, while still following the protocol. Below, when we call a scheme "adv out of n" scheme, we'll always mean $adv_a = adv$.

At the end of each section, we briefly discuss the running time and communication complexity of the scheme. We always consider only the slowest participant, unless the others are idle. We also restrict our attention to data sending (rather than receiving), because all communication is symmetric in all of the described algorithms.

We will express the running time in terms of basic operations, using the following notations. $T(M, l)$ and $T(A, l)$ are the running time in order of the multiplication and the adding for l-word unsigned integers. $T(C, l)$ is the running time of sending a l-word message, $T(R, l)$ is the running time of generating a l-word random number. As a shorthand for integers that fit in one word we write: $T(M) = T(M, 1)$, $T(C) = T(C, 1)$, $T(R) = T(R, 1)$. Finally, W is the number of bits in one word.

2 Multi-party protocols for the birthday problem

2.1 Voting based

Below, by voting protocol we mean a multi-party computation where each participant casts a 'yes' or 'no' vote, and the protocol computes the number of 'yes' and 'no' votes. The birthday problem can be solved using voting protocols [6]. In a naïve approach, we perform a voting for all possible values in V. Whenever a value receives more than one vote, we know there is a collision. Unfortunately, this is unfeasible when $|V|$ is large (for birthdays it could still work).

In order to improve the efficiency of the approach, we can partition the set V of possible values into subsets S_i, which we call *slots*. First we perform the votes for the subsets and then focus on values from those subsets S_i that have received at least two votes. This approach can reduce the number of the required voting rounds. Clearly, if the number of slots is too small, then there might be a lot of slots with at least two values and we have to test all values in these slots. On the other hand, if the slots are two small, then the number of slots is not much smaller than the number of possible values.

In the following we analyze the possible slot numbers in worst and average cases. We denote the number of possible values by $k \in \mathbb{Z}^+$, the number of participants by $n \in \mathbb{Z}^+$ and the number of slots by $q \in \mathbb{Z}^+$. We try to distribute the possible values among the slots as equally as possible and analyze the optimal choice of parameter q.

WorstCase If we distribute the participant values equally to the slots, each slot will contain $\left\lceil \frac{k}{q} \right\rceil$ or $\left\lfloor \frac{k}{q} \right\rfloor$ values. Let's call the slots that have $\left\lceil \frac{k}{q} \right\rceil$ values "full" slots. Denote the number of full slots by T. Then

$$T = \begin{cases} q, & \text{if } q \mid k \\ k - q\left\lfloor \frac{k}{q} \right\rfloor, & \text{otherwise} \end{cases}$$

The maximal number of slots with at least 2 participant values is $r = min\{\left\lfloor \frac{n}{2} \right\rfloor, q\}$. Denote the maximal number of full slots with at least 2 participant values by $T_r = min\{r, T\}$. The number of necessary voting rounds in the worst case is $q + T_r \left\lceil \frac{k}{q} \right\rceil + (r - T_r) \cdot \left\lfloor \frac{k}{q} \right\rfloor$.

In the case when $q \mid k$, we have $T = q$, so $T_r = min\{r, q\} = min\{min\{\left\lfloor \frac{n}{2} \right\rfloor, q\}, q\} = min\{\left\lfloor \frac{n}{2} \right\rfloor, q\}$. We get $r - T_r = 0$, so the number of voting rounds is $q + min\{\left\lfloor \frac{n}{2} \right\rfloor, q\}\frac{k}{q} = q + k \cdot min\{\left\lfloor \frac{n}{2} \right\rfloor \frac{1}{q}, 1\}$. The derivative w.r.t. q is $1 + k \cdot min\{\left\lfloor \frac{n}{2} \right\rfloor (-\frac{1}{q^2}), 0\}$ showing that in case of $\left\lfloor \frac{n}{2} \right\rfloor \geq q$ we do not have an optimal value of q. If $\left\lfloor \frac{n}{2} \right\rfloor \leq q$, then we have a minimum at $q = \sqrt{k \cdot \left\lfloor \frac{n}{2} \right\rfloor}$.

If we assume that $q \nmid k$, then $T = k - q\left\lfloor \frac{k}{q} \right\rfloor$, so $T_r = min\{r, T\} = min\{r, k - q\left\lfloor \frac{k}{q} \right\rfloor\} = min\{min\{\left\lfloor \frac{n}{2} \right\rfloor, q\}, k - q\left\lfloor \frac{k}{q} \right\rfloor\} = min\{\left\lfloor \frac{n}{2} \right\rfloor, k - q\left\lfloor \frac{k}{q} \right\rfloor\}$. In this case the formula for the rounds gives $q + T_r\left\lceil \frac{k}{q} \right\rceil + (r - T_r) \cdot \left\lfloor \frac{k}{q} \right\rfloor = q + T_r(\left\lceil \frac{k}{q} \right\rceil - \left\lfloor \frac{k}{q} \right\rfloor) + r \cdot \left\lfloor \frac{k}{q} \right\rfloor = q + T_r + r \cdot \left\lfloor \frac{k}{q} \right\rfloor = q + min\{\left\lfloor \frac{n}{2} \right\rfloor, k - q\left\lfloor \frac{k}{q} \right\rfloor\} + min\{\left\lfloor \frac{n}{2} \right\rfloor, q\} \cdot \left\lfloor \frac{k}{q} \right\rfloor$. Below we approximate $q + \left\lfloor \frac{k}{q} \right\rfloor$ by $q + \frac{k}{q}$ in order to simplify the calculation.

- If $\left\lfloor \frac{n}{2} \right\rfloor \geq q$, then $q + min\{\left\lfloor \frac{n}{2} \right\rfloor, k - q\left\lfloor \frac{k}{q} \right\rfloor\} + q \cdot \left\lfloor \frac{k}{q} \right\rfloor$. If we remove the floor functions then the derivative is 1 so the minimum is at one of the boundaries.
- If $\left\lfloor \frac{n}{2} \right\rfloor \leq q$, then $q + min\{\left\lfloor \frac{n}{2} \right\rfloor, k - q\left\lfloor \frac{k}{q} \right\rfloor\} + \left\lfloor \frac{n}{2} \right\rfloor \cdot \left\lfloor \frac{k}{q} \right\rfloor$. The derivative after removing floor functions is $1 + min\{0, 0\} - \left\lfloor \frac{n}{2} \right\rfloor \cdot \frac{k}{q^2}$, so the minimum is at $q = \sqrt{k \cdot \left\lfloor \frac{n}{2} \right\rfloor}$ which is usually better than the first case.

Average case We compute the expected number of slots with at least two participant values. This can be formulated as follows. Let $f : A \to B$ where $|A| = n$, $|B| = q$, $q < n$, f chosen uniformly among all such functions. We are interested in $E(\#\{b \in B | \left|f^{-1}(b)\right| > 1\}) = \sum_{b \in B} P(\left|f^{-1}(b)\right| > 1) = |B| \cdot P\left(\left|f^{-1}(b_1)\right|\right)$, where b_1 denotes the first slot. $P\left(\left|f^{-1}(b_1)\right|\right) = 1 - P(\left|f^{-1}(b_0)\right| = 1) - P(\left|f^{-1}(b_0)\right| = 0) = 1 - n\left(\frac{q-1}{q}\right)^{n-1}\frac{1}{q} - \left(\frac{q-1}{q}\right)^n$. We approximate this by $1 - \frac{n}{q}\left(\frac{1}{e}\right)^{\frac{n}{q}} - \left(\frac{1}{e}\right)^{\frac{n}{q}} = 1 - \frac{n+q}{q}\left(\frac{1}{e}\right)^{\frac{n}{q}}$. So the estimated expected number of slots with at least 2 values is: $q\left(1 - \frac{n+q}{q}e^{-\frac{n}{q}}\right) = q - (n+q)e^{-\frac{n}{q}}$. The number of voting rounds is $q + k/q\left(q - (n+q)e^{-\frac{n}{q}}\right)$. Deriving and solving for zero we get $n^2/q^3 = ke^{n/q}$ and thus we can compute the optimal choice for q.

If we assume that $\left\lfloor \frac{n}{2} \right\rfloor \leq q$ and $q \mid k$, then the worst case needs $q + \left\lfloor \frac{n}{2} \right\rfloor \frac{k}{q}$ voting rounds. Since we assumed semi-honest behavior, we can use a simple voting algorithm with leader (we show runtime in parenthesis): every participant sends a fragment to two others ($2T(C)$), everyone receives two shares and adds to their remaining share ($2T(A)$) and sends the fragment of the solution to the leader node ($T(C)$) who combines all received values ($nT(A)$). The running time is $\left(q + \left\lfloor \frac{n}{2} \right\rfloor \frac{k}{q}\right)(3T(C) + 2T(A) + nT(A))$.

Remark 1. We can use a multiple hashing (several orthogonal sets of slots) too, if k is small relative to n.

Remark 2. If user behavior is more complicated and we insist on more privacy, there are several voting protocols to be considered, e.g. [3, 6].

2.2 Pots

A folklore method for privately computing the average age of participants is the following. Start with one of the participants, called the seeder, putting a piece of paper containing a secret random value, the seed, into a pot. The seed could be chosen e.g. uniformly among the first one thousand positive integers. Then the participants secretly increment the value by their respective ages, one-by-one. At the end the seeder subtracts the seed and we get the sum of the ages.

We adapt this method for the birthday problem: let there be n participants, m seeders ($m \leq n$), and k pots, initially containing 0. We start by every seeder getting the pots and adding some random number to the number found in it (independently for every pot). They remember that number for later. To each participant, we assign a pot that will be responsible for taking into account the participant's value (birthday). When inserting their seed into the pots, the seeders also increment by 1 the value in the pot holding their secret.

Next, all non-seeder participants take the pots and add 1 to the pot assigned to their secret, and 0 to the other pots. Finally, the seeders subtract the random numbers they added at the beginning. The order in which the seeders perform the final phase is shuffled compared to the initial phase in order to have different predecessors and successors for extra privacy. We can always achieve this when $n \geq 5$ (we can find two disjoint Hamiltonian cycles in the complete graph with at least 5 vertices).

The adding/subtracting functions can come from an arbitrary Abelian group, e.g. exclusive or operation on a fixed length word, or a simple unsigned integer addition/subtraction in a $\mathbb{Z}_m$. In order to detect collisions for values from a set of size k, we could use a bit vector of length k. Adding a secret value of m to the pot means flipping the mth bit of the bit vector. If all values of the participants are distinct, then the number of the 1 bits in the final result is exactly n, otherwise we have flipped at least one bit back to 0, reducing the number of 1 bits.

To illustrate this method with an example, imagine that 3 people want to know if any two of them share the same favourite Star Wars movie from the original trilogy. To indicate which movie they prefer, everyone sets a bit vector of length three: 100 corresponds to the first movie, 010 to the second and 001 to the third one. The bitwise XOR of the three vectors reveals whether there is a collision: a necessary and sufficient condition for this is that the number of 1 bits is smaller than the number of participants (colliding 1s puts out each other). In order to do this with privacy preserved, everyone adds a random mask to the vectors which are then subtracted at the end.

The security level of the scheme depends on the number of seeders. If not all participants are seeders, all of the non-seeders' values can be claimed by the two neighboring participants, since they can simply calculate the difference. So this scheme is 2 out of n if $m < n$. If all of the participants are seeders, but we do not use the shuffling, we get 2 out of n again: in this case the neighbors can calculate the difference of the differences and get the secret. If we do use shuffling, we get a 4 out of n scheme. Below, when the $m = n$ case is considered, we always mean the

shuffled version, and the non-shuffled version if $m < n$.

For the runtime analysis, observe that in the seeder phase we have to generate one random number, perform two additions (add 1 or 0 to the random number and add to the pot) and send the pot to the next seeder. This is done in $max\{k, m\}$ rounds, so the time needed is: $max\{k, m\}(T(C)+2T(A)+T(R))$. The next phase is the value filling for non-seeders: $max\{k, (n-m)\}(T(C)+T(A))$. Finally removing seeds: $max\{k, m\}(T(C) + T(A))$. The overall complexity is: $max\{k, m\}(T(C) + 2T(A) + T(R)) + max\{k, (n-m)\}(T(C) + T(A)) + max\{k, m\}(T(C) + T(A)) = max\{k, m\}(2T(C) + 3T(A) + T(R)) + max\{k, (n-m)\}(T(C) + T(A))$.

2.3 Big Pot

We consider the special case where we only have one pot (unsigned integer), with k bits, initiated by the seeders (with random numbers). After seeding, every participant flips one bit of the pot corresponding to his or her secret. Finally the seeders remove their random numbers. If the number of one bits is not equal to the number of participants, we found a collision. In order to avoid the attack by the neighbors, it is also necessary to use n seeders.

The complexity of the algorithm is the following: $m(T(C, \lceil \frac{k}{W} \rceil)+2T(A, \lceil \frac{k}{W} \rceil)+ T(R, \lceil \frac{k}{W} \rceil)) + (n-m)(T(C, \lceil \frac{k}{W} \rceil) + T(A, \lceil \frac{k}{W} \rceil)) + m(T(C, \lceil \frac{k}{W} \rceil) + T(A, \lceil \frac{k}{W} \rceil)) = (m+n)T(C, \lceil \frac{k}{W} \rceil) + (2m+n)T(A, \lceil \frac{k}{W} \rceil) + mT(R)$.

2.4 Additive secret sharing based

In this section we consider schemes that are based on additive secret sharing. W.l.o.g, we assume secret values are from a finite field. The secret pieces of information are split into multiple fragments and shared in the following way: every participant holding secret x_i chooses 2 random numbers $x_{i,1}, x_{i,2} \in \mathbb{F}_{p^q}$ ($\mathbb{F}_{p^q}$ is a finite field with p^q element, where p is a prime, using the ordinary + operator) and then calculates $x_{i,3} = x_i - x_{i,1} - x_{i,2}$. Clearly $x_i = x_{i,1} + x_{i,2} + x_{i,3}$.

The problem statement (1) can be reformulated into an algebraic form (2) to better fit secret sharing.

$$f = sgn\left(\left|\prod_{i=1}^{n}\prod_{j=i+1}^{n}(x_i - x_j)\right|\right) \tag{2}$$

Clearly, the product vanishes if and only if there is a collision of values.

In the following assume that there are n participants and denote the ith participant's secret by $x_i = x_{i,1} + x_{i,2} + x_{i,3}$, $i = 1, \ldots, n$. Two of the three shares can be distributed, because without the third share it does not give any information for an adversary. In our approach, if q participants perform part of the protocol, we allow the ith participant to have access to shares $\{x_{j,k} \mid \forall j \in \{1..q\}, \forall k \in \{1,2,3\} : i \not\equiv k \ mod\ 3\}$.

Expanding the product in (2) gives an exponentially growing formula w.r.t. n, so we will relax privacy conditions and perform multiple collision-detection protocols

for smaller subsets of participants. We will consider the general collision detection protocol where collisions to be detected are given by a graph. For example, with people seated in a circle, we might only be interested in two *neighbors* having the same birthday, which corresponds to the collision-detection graph being a cycle.

If we cover all edges of the n-vertex complete graph by smaller collision-detection graphs (possibly redundantly), then we can detect all collisions, using several iterations on a more friendly version of (2).

We consider only simple finite and undirected graphs and will use standard graph-theoretical concepts (see e.g. [4] for graph concepts used). As usual, K_t denotes a complete graph with t vertices, $K_{t,u}$ denotes the complete bipartite graphs with t and u sized parts, P_t denotes a vertex disjoint path of length $t-1$, and S_t denotes the "star" graph with t edges ($K_{1,t-1}$). Below we focus on how the generalized version of the socialist millionaires' protocol can be performed on small collision-detection graphs.

2.4.1 $SMP(K_3)$

In the 3-participant case we want to find $sgn\left(|(x_1-x_2)(x_1-x_2)(x_2-x_3)|\right)$. In Table 1 we show which shares are made available to which participant in an encrypted way (one-to-one communication).

Table 1: Shares that one participant holds

1. participant		
$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
	$x_{2,2}$	$x_{2,3}$
	$x_{3,2}$	$x_{3,3}$

2. participant		
$x_{1,1}$		$x_{1,3}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
$x_{3,1}$		$x_{3,3}$

3. participant		
$x_{1,1}$	$x_{1,2}$	
$x_{2,1}$	$x_{2,2}$	
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Expanding the product, one finds that most terms can be computed by at least one participant individually. The part of the formula missing is (3).

$$2x_{12}x_{13}x_{21}-2x_{12}x_{13}x_{31}-2x_{12}x_{21}x_{23}+2x_{13}x_{31}x_{32}+2x_{21}x_{23}x_{32}-2x_{23}x_{31}x_{32} \quad (3)$$

With the help of a 4th participant, we can compute each of the summands because the necessary fragments can be sent to the helper without revealing any of the secrets. The fourth participant does not share a secret in this part.

Covering K_n by copies of K_3 graphs is not entirely trivial. The number of copies of K_3 needed is trivially between $\binom{n}{2}$ and $\binom{n}{2}/3$. The latter value is obtained by disjoint copies in the case of some special values of n using finite geometries. The overlapping decomposition a graph into the minimum number of complete subgraphs is NP-complete in general [11, 7]. There are polynomial time algorithms that creates cover by trees, $K_{1,k}$ or P_4 with overlap 2 [2]. In [17] it is proved that optimal covering is polynomial with S_k and P_k graphs. Covering a graph with

complete bipartite subgraphs, but not with a fixed size is discussed in [12]. The hardness of lane covering is discussed in [8]. Note that non-disjoint covers by small collision-detection graphs can leak information: if e.g. two participants detect a collision in two distinct 3-tuples with both of them involved in the collisions, the a posteriori probability of the two of them colliding increases largely.

Overall the collision detection protocol with $SMP(K_3)$ gives us an extra level of privacy compared to the pairwise socialist millionaires' protocol without adding to much computational overhead.

2.4.2 $SMP(P_3)$

Another approach computes only $(x_1 - x_2)(x_2 - x_3)$ for three participants, meaning we cover our complete graph with P_3 graphs. There is no need for a helper participant to do this type of sub protocol. The formulas for the participants can be seen in (4).

$$\begin{aligned} f1 =& x_{11}x_{23} - x_{11}x_{32} + x_{12}x_{23} - x_{12}x_{33} + \\ & x_{13}x_{22} + x_{13}x_{23} - x_{22}^2 - x_{13}x_{32} + x_{22}x_{32} + x_{23}x_{32} + x_{23}x_{33} \\ f2 =& x_{11}x_{22} - x_{11}x_{31} + x_{13}x_{21} - x_{13}x_{31} - \\ & x_{13}x_{33} - 2x_{21}x_{23} - x_{23}^2 + x_{21}x_{31} - 2x_{22}x_{23} + x_{22}x_{31} + x_{23}x_{31} \\ f3 =& x_{11}x_{21} - x_{11}x_{33} + x_{12}x_{21} + x_{12}x_{22} - \\ & x_{12}x_{31} - x_{12}x_{32} - x_{21}^2 - 2x_{21}x_{22} + x_{21}x_{32} + x_{21}x_{33} + x_{22}x_{33} \end{aligned} \tag{4}$$

Theorem 1 (Theorem B. from [17])**.** *Let p and q nonnegative integers, let n and k be positive integers such that $n \geq 4k$ and $k(p+q) = \binom{n}{2}$, and let one of the following conditions hold:*
(1) k is even and $p \geq \frac{k}{2}$,
(2) k is odd and $p \geq k$.
Then there exists a decomposition of K_n into p copies of P_{k+1} and q copies of S_{k+1}.

By Theorem 1, we can prove that we can decompose a complete subgraph with at least 4 vertices into P_3 graphs if $4 \mid n$ or $4 \mid (n-1)$. The theorem gives the number of covering graphs $p = \frac{n(n-1)}{4}$.

If $4 \mid n$, then every participant in one round generates two random number ($2T(R)$) subtracts two to achieve the secret fragmenting ($2T(A)$), sends two fragments ($4T(C)$) to the other participants (2-2 share to each), has 11 multiplications ($11T(M)$) and additions ($11T(A)$) and finally they share the f_i part of the solution to a leader in the group ($T(C)$). We have $\frac{n}{2}$ rounds, so we get $\frac{n}{2}\left(11T(M) + 13T(A) + 5T(C) + 2T(R)\right)$ for the overall running time.

2.5 $SMP(2K_2)$

A 4-participant approach that performs subprotocol based on $2K_2$ graphs (see Figure 1 (c)) can also solve the problem without a helper. The fourth participant's fragments can be seen in Table 2.

Table 2: Shares that one participant holds in 4 participant case

4. participant		
	$x_{1,2}$	$x_{1,3}$
	$x_{2,2}$	$x_{2,3}$
	$x_{3,2}$	$x_{3,3}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$

If we substitute the fragments and expand the $(x_1 - x_2)(x_3 - x_4)$ we get the participants formulas (5). Trivially a disjoint cover can be built up of the complete graph by two lines if $4 \mid n$ or $4 \mid n-1$. If $4 \mid n$, then in each round, no participant is idle.

$$\begin{aligned} f_1 =& x_{11}x_{32} - x_{11}x_{42} - x_{11}x_{43} + x_{12}x_{32} - x_{12}x_{42} \\ & - x_{22}x_{33} + x_{22}x_{42} - x_{23}x_{32} - x_{23}x_{33} \\ f_2 =& x_{13}x_{31} + x_{13}x_{33} - x_{21}x_{31} - x_{21}x_{33} + x_{21}x_{43} \\ & + x_{22}x_{41} + x_{22}x_{43} - x_{23}x_{31} + x_{23}x_{43} \\ f_3 =& x_{11}x_{31} + x_{11}x_{33} - x_{11}x_{41} + x_{12}x_{31} - x_{12}x_{41} \\ & - x_{21}x_{32} + x_{21}x_{41} + x_{21}x_{42} - x_{22}x_{31} \\ f_4 =& x_{12}x_{33} - x_{12}x_{43} + x_{13}x_{32} - x_{13}x_{41} - x_{13}x_{42} \\ & - x_{13}x_{43} + x_{23}x_{41} + x_{23}x_{42} - x_{22}x_{32} \end{aligned} \tag{5}$$

The complexity is as follows: every participant in one round generates the shares $(2T(R) + 2T(A))$, sends two fragments $(6T(C))$ for all of the other participants, and does 9 multiplications $(9T(M))$ and additions $(9T(A))$ and finally shares the f_i part of the solution with a leader in the group $(T(C))$. We have $\frac{n}{2}$ rounds, so we get $\frac{n}{2}(9T(M) + 11T(A) + 7T(C) + 2T(R))$.

2.6 Other collision-detection graphs

We also experimented with other collision-detection graphs. We expanded the formulas for different graphs and distributed the fragments by a randomized greedy algorithm. Figure 1 shows how many participants are needed for the different graphs used.

3 Comparison and conclusion

Some algorithms have some restrictions on the number of participants for which they can be applied. Leakage means some information that is unavoidably leaked in case of collisions. In Table 3, we compare the algorithms by the level of privacy,

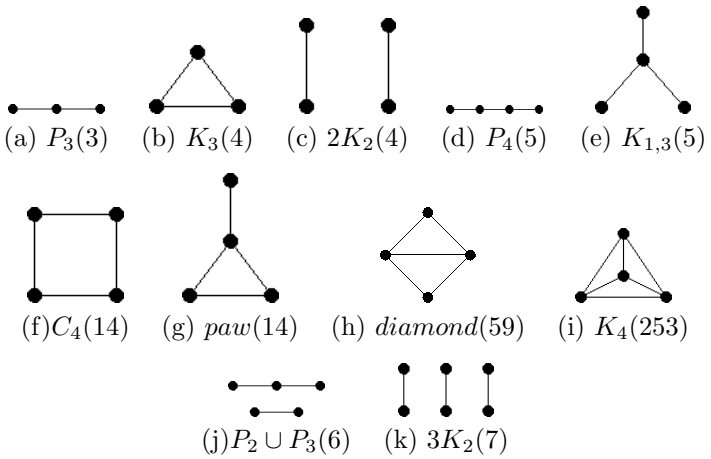

Figure 1: Graphs with number of necessary participants
Source: http://www.graphclasses.org/smallgraphs.html (reach: 2018-09-05)

the restrictions and show how many active adversaries in the system can claim any information of any other participant in the worst case. In Table 4 the runtimes can be seen.

Table 3: Adversary tolerance, most important information leakage and restrictions of the algorithms

Method name	Adversary	Leaked information	Restriction
Voting based	2	What is the duplicated value	
Pots ($m = 1$)	2	What is the duplicated value	
Pots ($m = n$)	4	What is the duplicated value	$n \geq 5$
Big pot ($m = 1$)	2	How many collisions exist	
Big pot ($m = n$)	4	How many collisions exist	$n \geq 5$
$SMP(P_3)$	2	Equality guess with $\frac{1}{2}$ probability	$4 \mid n \wedge n \geq 8$
$SMP(2K_2)$	2	Equality guess with $\frac{1}{2}$ probability	$4 \mid n \vee 4 \mid n-1$

Table 4: Estimated runtime of algorithms based on base functions (addition, multiplication, random number generation and communication)

Method name	Runtime
Voting based	$\left(q + \left\lfloor \frac{n}{2} \right\rfloor \frac{k}{q}\right)(3T(C) + 2T(A) + nT(A))$
Pots ($m = 1$)	$k(2T(C) + 3T(A) + T(R))+$ $max\{k, (n-1)\}(T(C) + T(A))$
Pots ($m = n$)	$max\{k, n\}(2T(C) + 3T(A) + T(R)) + k(T(C) + T(A))$
Big pot ($m = 1$)	$(n+1)T(C, \lceil \frac{k}{W} \rceil) + (n+2)T(A, \lceil \frac{k}{W} \rceil) + T(R)$
Big pot ($m = n$)	$2nT(C, \lceil \frac{k}{W} \rceil) + 3nT(A, \lceil \frac{k}{W} \rceil) + nT(R)$
$SMP(P_3)$	$\frac{n}{2}(11T(M) + 13T(A) + 5T(C) + 2T(R))$
$SMP(2K_2)$	$\frac{n}{2}(9T(M) + 11T(A) + 7T(C) + 2T(R))$

Let us estimate the runtime functions in the following way $T(M) = A \cdot T(A) = R \cdot T(R)$, $T(C) = C \cdot T(M)$ and let $T(A, r) = r \cdot T(A), T(R, r) = r \cdot T(R), T(C, r) = r \cdot T(C)$ and let $W = 64$ (the number of bits in one number). This is a reasonable approximation on modern architectures and software.

Table 5: Comparing runtimes of algorithms in $T(M)$ with multiple parametrizations

Parameters					
k	30	365	365	365	100
n	30	30	30	30	1000
C	5	5	20	2	5
A	1/3	1/3	1/3	1/3	1/3
R	1	1	1	1	1
Method estimations					
Voting based	1089	3798	10458	2466	156078
Pots ($m = 1$)	520	6327	22752	3042	6528
Pots ($m = n$)	520	6327	22752	3042	12533
Big pot ($m = 1$)	167	1000	3790	442	10680
Big pot ($m = n$)	360	2160	7560	1080	24000
$SMP(P_3)$	595	595	1720	370	19833
$SMP(2K_2)$	705	705	2280	390	23500

The $SMP(P_3)$ is worse than $SMP(2K_2)$ only if $1 + A > C$, which is a really unlikely case. Clearly the pitfall of the pot algorithms is the big k value.

When k and n are small, the big pot seems the most reasonable choice, but as k gets bigger, it becomes infeasible. The graph-based approaches have strong restrictions $4 \mid n \vee 4 \mid n-1$. It can be seen in Table 5 that the simple pots algorithm

becomes the best when n is large but k remains small.

In future work we plan to create a scheme based on multiple different graphs to avoid restrictions and achieve the best performance at the same time.

4 Acknowledgement

Péter Hudoba was supported by EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies — The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

Péter Burcsi has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

References

[1] Abramson, Morton and Moser, WOJ. More birthday surprises. *The American Mathematical Monthly*, 77(8):856–858, 1970.

[2] Alon, Noga, Caro, Yair, and Yuster, Raphael. Covering the edges of a graph by a prescribed tree with minimum overlap. *journal of combinatorial theory, Series B*, 71(2):144–161, 1997.

[3] Bárász, Mihály, Ligeti, Péter, Lója, Krisztina, Mérai, László, and Nagy, Dániel A. Another twist in the dining cryptographersprotocol. *Tatra Mountains Mathematical Publications*, 57(1):85–99, 2013.

[4] Bondy, John Adrian, Murty, Uppaluri Siva Ramachandra, et al. *Graph theory with applications*, volume 290. Citeseer, 1976.

[5] Boudot, Fabrice, Schoenmakers, Berry, and Traore, Jacques. A fair and efficient solution to the socialist millionaires problem. *Discrete Applied Mathematics*, 111(1-2):23–36, 2001.

[6] Chaum, David. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[7] Dor, Dorit and Tarsi, Michael. Graph decomposition is npc-a complete proof of holyer's conjecture. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 252–263. ACM, 1992.

[8] Ergun, Ozlem, Kuyzu, Gultekin, and Savelsbergh, Martin. The lane covering problem. *Manuscript*, 2003.

[9] Goldreich, Oded. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, New York, NY, USA, 2004.

[10] Hirt, Martin. *Multi Party Computation: Efficient Protocols, General Adversaries, and Voting*. Hartung-Gorre, 2001.

[11] Holyer, Ian. The np-completeness of some edge-partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

[12] Jukna, Stasys and Kulikov, Alexander S. On covering graphs by complete bipartite subgraphs. *Discrete Mathematics*, 309(10):3399–3403, 2009.

[13] Lin, Hsiao-Ying and Tzeng, Wen-Guey. An efficient solution to the millionaires problem based on homomorphic encryption. In *International Conference on Applied Cryptography and Network Security*, pages 456–466. Springer, 2005.

[14] Mathis, Frank H. A generalized birthday problem. *SIAM Review*, 33(2):265–270, 1991.

[15] Maurer, Ueli. Secure multi-party computation made simple. *Discrete Applied Mathematics*, 154(2):370–381, 2006.

[16] Pinkas, Benny. Fair secure two-party computation. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 87–105. Springer, 2003.

[17] Shyu, Tay-Woei. Decomposition of complete graphs into paths and stars. *Discrete Mathematics*, 310(15-16):2164–2169, 2010.

[18] Wagner, David. A generalized birthday problem. In *Annual International Cryptology Conference*, pages 288–304. Springer, 2002.

[19] Yao, Andrew C. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[20] Yao, Andrew Chi-Chih. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.