A Bridge Between Private Set Intersection Algorithms

Transforming MPSI into OT-MPSI

Maciej Kozik[1]
Supervisor: Dr. Zeki Erkin[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Maciej Kozik
Final project course: CSE3000 Research Project
Thesis committee: Dr. Zeki Erkin, Dr. Merve Gürel

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Financial criminals often manage to avoid being detected. Although banks can nowadays easily query and analyse the financial data they have access to, it is still an issue when multiple banks need to collaborate with each other to catch a criminal. Privacy laws prohibit them from simply sharing all their data with each other. Hence, methods for privately sharing their data are needed to ensure a smooth and secure collaboration. Computing a function between multiple parties, while keeping their inputs secure, is called a Secure Multiparty Computation (SMPC). One of its subtypes is Multiparty Private Set Intersection (MPSI). It allows for computing an intersection between multiple sets without revealing them. A special case of MPSI is Threshold-MPSI (T-MPSI), which returns items that are in at least T sets, while still keeping the actual sets hidden. Although these two protocols are closely correlated, there is no secure or efficient protocol to transform one into the other. Therefore, we focus our paper on making a step towards finding one. We design a transformation from MPSI to a slightly weaker variant of T-MPSI, which still keeps the information of who contributed an item to the intersection secure by utilising dummy Bloom Filters and Mental Poker. We focus our paper on the security aspect of the protocol, while keeping the efficiency bottleneck of the current state-of-the-art solution. Therefore, the experimental results of the runtime indicate that it is not yet a practical solution. However, it marks a step towards finding one.

1 Introduction

In the 1920s in the USA, the infamous criminal Al Capone used laundromats to hide illicit income [1]. He would funnel the profits from his criminal empire through a seemingly clean business of laundromats. The term "money laundering" originated from his actions and has since been used to describe the action of making illegal income look legitimate. It would seem that a century later, in the era of internet banking, where each bank can efficiently look up and analyse its transactional data, criminals would find it hard to commit financial crimes. However, this is not the case, as nowadays money laundering accounts for 2 to 5% of global GDP [2]. This is just one example of a financial crime; there are others, such as credit card fraud, scamming the elderly, and embezzlement. The scale of such activities prompts research into detecting and preventing them.

However, financial data associated with one crime may be distributed among different institutions. A single bank can only look into the data they have, limiting the detection capabilities. Hence, collaboration between financial institutions is necessary. This introduces privacy concerns, as banks cannot share client data, as imposed by the General Data Protection Regulation (GDPR) [3] in the European Union. Hence, methods for private computations over decentralised data are in high demand.

Evaluating a function privately using inputs from multiple parties is called a Secure Multiparty Computation (SMPC), first introduced by Yao [4] to solve the Millionaires' Problem. It concerned two people wanting to compare their wealth, without revealing the actual numbers. Since then, the field has seen many developments, including a framework that solves SMPC in general [5].

One of the subtypes of SMPC is the Multiparty Private Set Intersection (MPSI), first proposed by Freedman et al. [6]. It allows multiple parties holding confidential sets to compute their intersection, without revealing any other information about the input sets. It is of special interest in a financial setting, as it can be used to compare sets of suspicious accounts between banks without revealing confidential information. Since general SMPC

protocols are inefficient to solve MPSI [7], protocols specialised for this problem are needed. Recent developments focus on making MPSI suited for practical applications, mostly putting importance on efficiency [8, 7, 9, 10, 11].

However, for certain applications, the requirement for all sets to contain a certain item may be too strict. Take, for example, the case of 10 banks comparing their private sets of suspected clients. If all but one suspect a certain client, MPSI will not include them in the result, whereas it seems reasonable to do the opposite. For this purpose, one can use Threshold Multiparty Private Set Intersection (T-MPSI). It includes an item in the intersection if at least a threshold of private sets contain it, while still not revealing any other information about the confidential inputs. It was first introduced by Kissner and Song [12], although under a different name. Since then, only a few other T-MPSI algorithms were formulated, including but not limited to [7, 11, 13].

Since the range of T-MPSI algorithms is limited and they generally offer worse time complexities on paper [14, 15], efficiently and securely transforming an MPSI algorithm into T-MPSI is an interesting and practical research direction. This, however, has not been studied in detail, as the current state-of-the-art is the naive solution [13, 15], offering inefficient runtime and revealing partial information. In this paper, we focus on limiting the partial information that the parties can deduce when running multiple MPSI instances consecutively to form a slightly weaker variant of T-MPSI, which we call Over-Threshold-MPSI (OT-MPSI). It differs from T-MPSI in the fact that OT-MPSI also reveals how many sets each item that meets the threshold is in. The owners of each item in the intersection are still not revealed, and no party gets to know any information about other items. We operate in a security model that allows semi-honest adversaries, meaning they follow the protocol, but try to infer as much information as possible from the data they have to break the security of the protocol. Hence, the research question:

How to securely transform an MPSI protocol into an OT-MPSI protocol with an arbitrary threshold in the presence of semi-honest adversaries?

The main contribution of this paper is a method for converting an MPSI algorithm by Bay et al. [11] into OT-MPSI such that it limits the information each party can deduce about intermediate MPSI results. We achieve this by leveraging the Mental Poker protocol [16, 17, 18] and dummy Bloom Filters [19].

This paper is structured as follows. In Section 2, we explain the preliminaries and notation. Then, in Section 3, we give an overview of the related work. Our algorithm is presented in Section 4. We describe the experiments and the results in Section 5. We make a note about responsible research in Section 6. We devote Section 7 to the discussion of our findings. In Section 8, we conclude our paper and present future work.

2 Preliminaries and Notations

2.1 Notations

For the overview of notations used in this paper, please refer to table 1.

2.2 Homomorphic Encryption

We call an encryption scheme homomorphic if it supports operations on ciphertexts that correspond to operations on plaintexts. Definition 2.1 formalises this notion.

Table 1: Notations

t	Number of parties
T	Threshold for T-MPSI
l	Collusion threshold
k	Number of hash functions
n	Size of a set
m	Size of a Bloom Filter
P_i	i^{th} party
S_i	Set of party P_i
BF_i^0	Dummy Bloom Filter of party P_i
BF_i^1	Bloom Filter corresponding to S_i
v_b^n	b^{th} entry of the n^{th} subset vector
$Enc(M)$	Encryption of M
$Dec(M)$	Decryption of M
$Enc_K(M)$	Encryption of M by key K
$Dec_K(M)$	Decryption of M by key K
$Pr[A]$	Probability of event A
$\mathcal{C}$	Coalition of colluding parties
$\mathcal{S}$	Simulator

Definition 2.1. *(Homomorphic Encryption Scheme)*: An encryption scheme is homomorphic in operation $*$ if there exists an operation $\star$ such that for all plaintexts M_1, M_2, it holds that $Dec(Enc(M_1) \star Enc(M_2)) = M_1 * M_2$.

In this paper, we use the Additively Homomorphic Encryption, which supports homomorphic addition of encrypted plaintexts and multiplication by a plaintext scalar. An example of such a scheme is the Paillier Cryptosystem [20].

2.3 Mental Poker

First proposed by Shamir et al. [16], Mental Poker refers to playing Poker over the network. While playing in person allows all participants to visually verify the honesty of shuffling and dealing, in a distributed setting, this is not possible. Hence, cryptographic solutions are needed to ensure that no person can cheat.

The first protocol [16] leveraged Commutative Encryption, meaning that for any keys K_1, K_2 and any message M, it holds that $Enc_{K_1}(Enc_{K_2}(M)) = Enc_{K_2}(Enc_{K_1}(M))$. The protocol for two parties can be found in Figure 1.

One can see that at the end of the protocol in Figure 1, both Alice and Bob have five confidential cards, which can be decrypted by them, by the properties of Commutative Encryption. The protocol can be extended to more than two parties.

In this paper, we use Mental Poker to deal each of the $t-1$ parties one card. In our implementation, we use the protocol from [16] due to its simplicity. We note, however, that it leaks partial information [21], and that there exist more secure protocols [17, 18], but they make use of complex cryptographic primitives beyond the scope of this paper.

PARAMETERS: Symmetric commutative encryption keys K_A, K_B for Alice and Bob respectively

PROTOCOL:

(1) Alice takes the deck $X = \{x_1, \ldots, x_{52}\}$, and encrypts each card with their key to get $EX = \{Enc_{K_A}(x_1), \ldots, Enc_{K_A}(x_{52})\}$, shuffles EX and sends it to Bob

(2) Bob selects five random cards from the deck they received from Alice and sends them back to Alice.

(3) Bob chooses five other cards randomly and encrypts each with their own key to get $B = \{Enc_{K_B}(Enc_{K_A}(b_1)), \ldots, Enc_{K_B}(Enc_{K_A}(b_5))\}$ and sends them to Alice

(4) Alice decrypts each element in B with their key and sends back the result to Bob

Figure 1: Mental Poker between Alice and Bob

2.4 Bloom Filters

A Bloom Filter is a space-efficient set representation, proposed by Bloom [19]. It encodes a set of size n as an array A of size m, which initially has all entries equal to 0.

It uses k hash functions $h_1, \ldots, h_k$, where $h_i : \{0,1\}^* \to \{0, \ldots, m-1\}$. To insert an element x, one hashes it by all k hash functions to get $h_1(x), \ldots, h_k(x)$ and sets each $A[h_i(x)]$ to 1, for $i \in \{1, \ldots, k\}$. To check the membership of an element x, one hashes it by all k hash functions to get $h_1(x), \ldots, h_k(x)$ and checks if $\forall_{i \in \{1,\ldots,k\}} A[h_i(x)] = 1$. If it does not hold, x is not in the set; otherwise, the membership query returns that x is in the set.

Although false negatives are not possible in this representation, the space efficiency comes at a cost of a non-zero false positive rate. One can notice that it is possible for all array positions corresponding to some x to be set to 1, even though it was not added to the set. However, the probability of this happening can be mitigated by choosing the right m and k for the number of items n.

2.5 IND-CPA Security

Figure 2 presents a game between an adversary A and a challenger C. An encryption scheme is called IND-CPA (Indistinguishability under Chosen-Plaintext Attack) secure if A has a $\approx 50\%$ chance of winning this game.

Note that by design of the game in Figure 2, if an encryption scheme is deterministic, then it is not IND-CPA secure. The adversary could first send m_0 to C in the query phase to get $c_0 = Enc_K(m_0)$. Then they could send m_0 and some other message m_1 in the challenge phase and get back c. If $c = c_0$, they guess m_0, otherwise m_1. This means they can always win the game.

2.6 MPSI Protocol by Bay et al.

We base our work on the MPSI algorithm by Bay et al. [11]. It is secure against semi-honest adversaries. Moreover, it is secure against collusion - parties exchanging information to gain

PARAMETERS: An adversary A, challenger C who owns a secret key K to an encryption scheme

PROTOCOL:

QUERY PHASE:

(1) A sends a polynomially-bounded amount n of plaintexts to C, who encrypts them and sends back the ciphertexts $c_1, \ldots, c_n$ to A.

CHALLENGE PHASE:

(1) A chooses two messages m_0, m_1 of equal length, such that $m_0 \neq m_1$. Then sends them to C.

(2) C chooses a random bit i, encrypts m_i and sends $c = Enc_K(m_i)$ to A.

(3) A attempts to guess i based on c and $c_1, \ldots, c_n$, if they succeed, they win, otherwise they lose.

Figure 2: IND-CPA game

an advantage at breaking the security of the protocol. The largest size of a coalition of colluding parties that does not break the protocol is called the collusion threshold, which we denote by l. For the protocol from [11], $l = t - 1$. Since understanding its inner workings is crucial to understanding our solution and the motivation behind our choices, we dedicate this section to briefly explaining how MPSI from [11] works.

Figure 3 shows the overview of the MPSI algorithm. Party P_t is called the server, and all other parties are called clients. The protocol uses a Threshold Additively Homomorphic IND-CPA secure encryption scheme, meaning it requires a threshold amount of share decryption keys to decrypt the ciphertext. It also uses a mechanism called Share Decryption to Zero ($ShDec0$), which decrypts the ciphertext to 0 if the plaintext was 0, otherwise it results in a random value. Understanding the exact mechanism of $ShDec0$ is not needed for this paper.

It is crucial to note that since the clients send their Inverted Bloom Filters encrypted under an IND-CPA secure scheme, encryptions of equal bits are not necessarily equal due to the fact that it is a non-deterministic encryption. This makes sure that the server cannot infer any information about the relation between individual bits in each Bloom Filter.

3 Related Work

In this section, we present an overview of the related work. All protocols we mention are secure in the semi-honest model. Unless otherwise stated, they have a collusion threshold of $t - 1$.

MPSI has numerous implementations. The first protocol was proposed by Freedman et al. [6] in 2004. It uses homomorphically encrypted polynomials to compute the intersection and shares of zero to mask the private sets of both the leader and the other parties. However, it is not practical due to the time complexity of multiplying polynomials. Another early

PARAMETERS: t parties $P_1, \ldots, P_t$

INPUT: Party P_i has as input the set S_i, a share of the decryption key sh_i. All parties have the public encryption key pk.

PROTOCOL:

(1) The server P_t chooses k hash functions $\mathcal{H} = h_1, \ldots, h_k$ and sends them to all clients.

(2) Each client P_i, $i \in \{1, \ldots, t-1\}$, computes the Bloom Filter corresponding to set S_i based on the k hash functions - BF_i, flips all its bits to get the Inverted Bloom Filter IBF_i, and encrypts each bit separately with pk to get the Encrypted Inverted Bloom Filter $EIBF_i$. Each client P_i sends $EIBF_i$ to the server.

(3) The server, for each $y_j \in S_t$, computes its corresponding indices in the Bloom Filters $H_j = \{h(y_j) | h \in \mathcal{H}\}$. Then it homomorphically adds all entries corresponding to indices in H_j from all $EIBF$s. We denote the encrypted sum as s_j.

(4) The server participates with the clients in $ShDec0$ to see which s_j were encryptions of 0. For all that were, the server adds it to the output. Note that at this point, only the server has the output.

Figure 3: MPSI protocol from [11]

MPSI protocol is the one proposed by Kissner and Song [12], which also uses polynomials to represent the sets. They achieve security by leveraging uniformly random polynomials to hide the true set representation.

The first practical implementation was created by Kolesnikov et al. [8]. They introduced a new construction called an Oblivious Programmable Pseudo-Random Function (OPPRF), which returns random values for all inputs except those programmed to return a specific value. It is unknown to the party that queries the OPPRF whether they are given a pre-programmed value, and it is unknown to the party that programmed the OPPRF what it was queried on. [8] leveraged this new construction, combined with zero-sharing, to create an efficient and practical implementation of MPSI.

There were several improvements to the work of Kolesnikov et al. [8] over the years. Chandran et al. [7] removed the expensive zero-sharing phase and created a more efficient algorithm in a semi-honest setting with a collusion threshold of $\lfloor \frac{t-1}{2} \rfloor$ parties. Garimella et al. [9] improved the efficiency of OPPRFs and proposed a generalisation of the idea called an Oblivious Key-Value Store (OKVS).

The protocol we base our work on is from Bay et al. [11]. They created a practical implementation of MPSI based on Bloom Filters and Threshold Additively Homomorphic Encryption. For a detailed description of the protocol, please refer to Section 2.6. Their implementation excels in the case of many parties computing the intersection with smaller sets.

While MPSI received much attention, the research into T-MPSI is limited. The first proposed protocol of this functionality was by Kissner and Song [12]. They use polynomial roots combined with function derivatives. Several subsequent protocols followed [7, 13], with

the first practical and open-sourced implementation being the one by Bay et al. [11].

The research into transforming MPSI protocols into T-MPSI is even more limited. Mahdavi et al. [13] mention a naive solution to a variant of T-MPSI of weaker security, which reveals both the item counts and their owners in the result. The solution works by running MPSI on all subsets of parties of size T and then combining the intermediate results. They note that it is highly inefficient since it requires $\binom{t}{T}$ runs of MPSI for t parties and threshold T. Guan [15] also mentions this solution. To the extent of our knowledge, nobody has considered the security aspect of the naive solution to implement T-MPSI that reveals neither the item counts nor their owners.

4 Our MPSI to OT-MPSI$_{\text{individual}}$ transformation

4.1 Problem Formulation

We operate in a scenario where one party, denoted as the server P_t, with a set S_t, wants to see which of the items in S_t are in at least T sets among $P_1, \ldots, P_t$ parties. Other parties are called clients. In that sense, it is the setting of the so-called T-MPSI$_{\text{individual}}$ [22]. However, our protocol has slightly weaker security guarantees, as it also reveals how many sets each item that meets the threshold is in. We call it Over-Threshold-MPSI$_{\text{individual}}$ (OT-MPSI$_{\text{individual}}$). Therefore, the output is a list of entries of the form (e, x_e), where e is an item in the intersection and x_e is the corresponding count. We still, however, do not reveal who actually contributed each item to the threshold, unlike a very similar formulation from [13].

We concern ourselves with the semi-honest security model with the possibility of colluding parties. We denote the collusion threshold as l. Since our goal is to achieve $l = t - 1$, when we run intermediate MPSI instances, we choose the cryptographic parameters of each MPSI so as to ensure collusion resistance of $t - 1$.

When P_t wants to calculate their OT-MPSI$_{\text{individual}}$, they run N MPSI instances, for all $N = \binom{t-1}{T-1}$ distinct subsets of T parties that P_t is in, denoted as MPSI$_1$, ..., MPSI$_{\text{N}}$. Then at the end, P_t unions all intermediate results to get the final output.

Executing this procedure without proper care leaks information. Namely, for any MPSI$_{\text{n}}$, P_t knows with which subset of clients they computed the intersection with, meaning they can directly see, for each item in the result of the whole procedure, which parties had this item in their sets. To mask who participated in each MPSI, we use dummy Bloom Filters. To achieve security against collusion, we leverage Mental Poker.

4.2 Proposed Algorithm

We focus our work on adapting the MPSI algorithm by Bay et al. [11] in such a way that it can be run multiple times to implement OT-MPSI$_{\text{individual}}$, while masking who was the owner of each item in the result. Protocol in Figure 5 shows our transformation from the MPSI by Bay et al. to OT-MPSI$_{\text{individual}}$. For simplicity, we assume that the Bloom Filter size and the hash functions are fixed for all intermediate MPSI runs. This allows each client P_i to compute BF_i^0 and BF_i^1 only once.

Note that we use a Mental Poker oracle Q_a in our protocol; Figure 4 describes its functionality. In short, Q_a randomly deals identifiers in $\{1, \ldots, a\}$ such that each is given exactly once. Each client uses it to get to know their identifier without any other parties knowing it.

PARAMETERS: a, $a \in \mathbb{N}$

BEHAVIOUR: Waits for a query from P_i, $i \in \{1, \ldots, a\}$, returns $c_i \in \{1, \ldots, a\}$, subject to constraint $\forall_{i \neq j} c_i \neq c_j$

Figure 4: Functionality of a Mental Poker oracle Q_a

Functionality of a Mental Poker oracle Q_a can be implemented by playing Mental Poker [16] between a parties, where each party gets one card, and each card means a distinct identifier in $\{1, \ldots, a\}$.

PARAMETERS: t parties $P_1, \ldots, P_t$, Mental Poker oracle Q_{t-1}

INPUT: Party P_i has as input the set S_i, represented by BF_i^1, and a dummy Bloom Filter BF_i^0 with every entry equal to 1

PROTOCOL:

INITIALISATION:

(1) Each client P_i, $i \in \{1, \ldots t-1\}$ queries the Mental Poker oracle Q_{t-1} and gets $c_i \in \{1, \ldots, t-1\}$ in return.

RUNNING MPSIs:

(1) P_t generates all $N = \binom{t-1}{T-1}$ possible vectors v of the form $v \in \{0,1\}^{t-1}$, $\sum_{b=1}^{t-1} v_b = T-1$. We denote them as $v^1, \ldots, v^N$.

(2) For each $n \in \{1, \ldots, N\}$:

    (1) All parties engage in MPSI_n with the server being P_t.

    (2) Each client P_i, inputs $BF_i^{(v^n_{c_i})}$ to MPSI_n.

    (3) P_t gets the result I_n.

OUTPUT CALCULATION:

(1) P_t calculates $I = \bigcup_{n=1}^{N} I_n$.

(2) For each $e \in I$, P_t calculates x_e by solving the following equation: $|\{I_n | n \in \{1, \ldots, N\} \wedge e \in I_n\}| = \binom{x_e - 1}{T-1}$.

(3) P_t outputs $\{(e, x_e) | e \in I\}$.

Figure 5: Transformation from MPSI by Bay et al. to $\text{OT-MPSI}_\text{individual}$

The protocol in Figure 5 works as follows:

1. Each client P_i queries the Mental Poker oracle to get their c_i.
2. For each of the N possible subsets of clients of size $T-1$, the server sends the current subset vector v^n to all clients. v^n is just a binary string that has exactly $T-1$ ones and $t-T$ zeroes. Client P_i is in the n^{th} subset if $v^n_{c_i} = 1$, otherwise not. They respectively input either their actual Bloom Filter, or a dummy Bloom Filter with all entries equal to 1, to MPSI_n. P_t gets each intermediate MPSI result.
3. P_t outputs the union of all intermediate results along with the count for each element in this union.

4.3 Protocol Correctness

For a specific subset of parties $\mathcal{P}$, the parties that are not in $\mathcal{P}$ input a dummy Bloom Filter filled with all ones. This, by the underlying MPSI protocol [11], has no impact on the outcome, and the output for the MPSI is the intersection between the server and the clients that are currently in the subset. Since we cover all subsets of $T-1$ clients, the output of the final union is trivially $\text{OT-MPSI}_\text{individual}$ for P_t. Since we use Bloom Filters, there is a probability for a false positive to occur in the final intersection; however, it can be mitigated by choosing the right parameters for the Bloom Filters [11].

4.4 Security Analysis

We follow a standard way of proving security in the presence of semi-honest adversaries, as outlined in [23]. First, we need to define the notion of negligible functions in Definition 4.1 and computational indistinguishability in Definition 4.2, which is the simplified version from [11].

Definition 4.1. *(Negligible Function)*: A function $\mu(n)$ is called negligible if for every positive polynomial $p(n)$, there exists a sufficiently large k, such that:

$$\mu(k) < \frac{1}{p(k)}$$

Definition 4.2. *(Computational Indistinguishability)*: For two probability ensembles $X = \{X_k\}_{k\in\mathbb{N}}$, $Y = \{Y_k\}_{k\in\mathbb{N}}$, indexed by a security parameter k, we call them computationally indistinguishable ($X \overset{c}{\equiv} Y$), if for every probabilistic polynomial-time algorithm D, there exists a negligible function $\mu(n)$, such that for every $n \in \mathbb{N}$:

$$Pr[D(X_n) = 1] - Pr[D(Y_n) = 1] \leq \mu(n)$$

In Definition 4.3, we define what it means for a deterministic functionality to be secure in a semi-honest model.

Definition 4.3. *(Security of a Deterministic Functionality $\mathcal{F}$ in a Semi-Honest Model)*: Let $\mathcal{F}$ be a deterministic functionality modelling a t-party protocol Π. Denote $\mathcal{F}_i$ as the component of $\mathcal{F}$ that runs on party P_i. We denote the view of P_i during Π run on inputs $X = x_1, \ldots, x_t$, as $view_i^\Pi(X) = (i, x_i, r_i, \mathcal{M}_i, \mathcal{F}_i(X))$, where $\mathcal{M}_i = m_i^1, \ldots, m_i^d$ are the messages P_i receives during the execution and r_i is the content of their internal random

tape. We call Π secure in a semi-honest model if there exists a polynomial-time algorithm $\mathcal{S}$, called a simulator, such that for every $i \in \{1, \ldots, t\}$:

$$\{\mathcal{S}(i, x_i, \mathcal{F}_i(X))\}_{X \in (\{0,1\}^*)^t} \stackrel{c}{\equiv} \{view_i^{\Pi}(X)\}_{X \in (\{0,1\}^*)^t}$$

Since we are dealing with colluding parties, in Definition 4.3, instead of considering a single party i, we consider a coalition $\mathcal{C}$ that has a collective input $X_{\mathcal{C}}$, collective component $\mathcal{F}_{\mathcal{C}}$, and the union of all messages that they receive $\mathcal{M}_{\mathcal{C}}$.

We consider two cases:

1. P_t does not share information with any of the clients
2. P_t is a part of a colluding coalition $\mathcal{C}$ with another clients

In both cases, for simplicity, we assume that the server P_t generates the subset vectors v^n in a deterministic order.

In the first case, the view of any coalition of colluding parties $\mathcal{C}$, with $|\mathcal{C}| \leq t-1$, is their collective input $X_{\mathcal{C}} = \{x_i | P_i \in \mathcal{C}\}$, all messages they receive $\mathcal{M}_{\mathcal{C}}$, and no output of the protocol. Hence, we can construct a simulator $\mathcal{S}$ that does the following:

1. For the Mental Poker oracle, it generates random messages for each of the parties in $\mathcal{C}$, while still making sure they belong to a random permutation of all $c_i \in \{1, \ldots, t\}$.
2. Broadcasts the deterministic subset vectors v^n to the clients.
3. For each MPSI_n, $\mathcal{S}$ runs a simulator $\mathcal{S}_{MPSI}$ of the underlying MPSI implementation on inputs: $\mathcal{C}$'s inputs for MPSI_n.

We see that parties in $\mathcal{C}$ cannot distinguish between the real and simulated execution, which relies on the security of the underlying MPSI and the randomness of the Mental Poker. Therefore, for this case, the protocol is secure.

In the second case, the coalition $\mathcal{C}$, $|\mathcal{C}| \leq t-1$, has additionally the output $\mathcal{F}_t(X)$ and the messages the leader receives. We construct a simulator $\mathcal{S}$ that does the following:

1. For the Mental Poker oracle, it generates random messages for each of the parties in $\mathcal{C}$, while still making sure they belong to a random permutation of all $c_i \in \{1, \ldots, t\}$.
2. Broadcasts the deterministic subset vectors v^n to the clients.
3. Based on the final result $\{(e, x_e) | e \in I\}$ and the coalition's input sets $X_{\mathcal{C}}$, $\mathcal{S}$ randomly generates the intermediate results I_n such that the final result stays the same and the intermediate results are valid based on the current participants of $MPSI_n$.
4. For each $MPSI_n$, $\mathcal{S}$ runs a simulator $\mathcal{S}_{MPSI}$ of the underlying MPSI implementation on inputs: I_n and $\mathcal{C}$'s inputs for MPSI_n.

We see that parties in $\mathcal{C}$ cannot distinguish between the real and simulated execution. This case relies heavily on the fact that the server gets the IND-CPA-encrypted Bloom Filters from the clients. Hence, inputs from other parties can be simulated randomly as long as they keep the output unchanged. As in the first case, it also relies on the security

guarantees of Mental Poker and the security proof from [11].

Therefore, we conclude that our protocol in Figure 5 is secure against semi-honest adversaries with the collusion threshold $l = t - 1$.

4.5 Complexity Analysis

In the initialisation phase, the runtime depends on the underlying Mental Poker implementation. If we use a fast algorithm from Wei and Wang [18], it would add $O(Kt^2)$ runtime overhead per client, where K is the security parameter of the protocol. We omitted from the runtime the variables that represent the complexity of performing zero-knowledge proofs, multiplications and exponentiations. Then, the protocol requires $\binom{t-1}{T-1}$ runs of MPSI, each between all t parties. To union all intermediate results, if we assume a set representation based on hashing, and that the largest intermediate result has n elements, then the union has $O(\binom{t-1}{T-1}n)$ time complexity. To calculate the count x_e for each item e in the union I, we need to solve the equation $|\{I_n | n \in \{1, \ldots, N\} \wedge e \in I_n\}| = \binom{x_e-1}{T-1}$. For a fixed T and t, we can calculate $\binom{x_e-1}{T-1}$ for all possible values of x_e in $O(t)$ time. We can do it once at the beginning of the protocol and have a lookup table for all possible solutions to this equation. Then, if we store for each e, the amount of subsets it appeared in $|\{I_n | n \in \{1, \ldots, N\} \wedge e \in I_n\}|$, we can look up the solution to the equation from the precomputed values in constant time.

5 Implementation and Experimental Analysis

5.1 Experimental Setup and Results

We provide an implementation of our OT-MPSI$_{\text{individual}}$ in C++[1]. It reuses most of the code from [11]. Hence, it inherits all of its dependencies. Our implementation runs on one machine and spawns a thread for each client whenever they need to perform independent operations. However, most of the code still runs on a single thread. As for the Mental Poker implementation, we implement a protocol by [16], due to its simplicity.

We run both our OT-MPSI$_{\text{individual}}$ protocol and the naive implementation for $T = t-1$ and $T = t/2$. For $T = t-1$, we perform 20 runs for every combination of the parameters, and for $T = t/2$, we run 10 due to significantly higher runtime in this case. The benchmarking code was run on a 64-bit Ubuntu 24.04.4 LTS machine with an Intel Core Ultra 7 155H processor with 22 $\times$ 4.80GHz and 32GB RAM. As the underlying cryptographic parameters, we choose $t-1$ as a threshold for the Threshold Paillier Cryptosystem, and, similar to [11], $\kappa = 1024$. Parameters for Bloom Filters were chosen the same way as in [11]. We present our results for $T = t - 1$ in Figure 6 and for $T = t/2$ in Figure 7. To see the raw experimental data, please refer to Appendix A.

5.2 Result Analysis

Comparing the results for OT-MPSI$_{\text{individual}}$ in Figures 6 and 7, we see that the runtime depends heavily on the relation between the threshold for the intersection T and the

[1] `https://github.com/macinson/MPSI_to_T-MPSI`

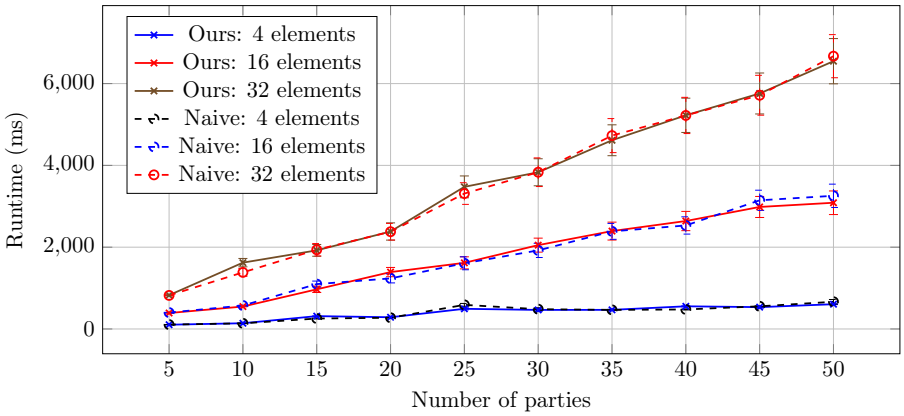

Figure 6: Runtime comparison between our OT-MPSI$_{\text{individual}}$ protocol and the naive protocol at $T = t - 1$, for different numbers of parties and set sizes. Datapoints are averaged over 20 runs. Error bars represent $\pm\sigma$.

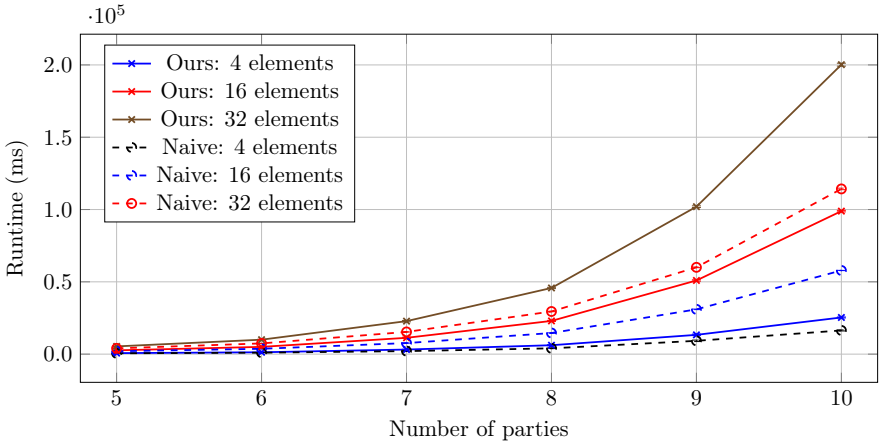

Figure 7: Runtime comparison between our OT-MPSI$_{\text{individual}}$ protocol and the naive protocol at $T = t/2$, for different numbers of parties and set sizes. Datapoints are averaged over 10 runs. Error bars represent $\pm\sigma$.

number of parties t. This is backed by the fact that for $T = t - 1$, it is required to do $t - 1$ runs of MPSI. However, for $T = t/2$, the number of MPSI runs is exponential in t. It can be shown that, as $d = |T - t/2|$ decreases, meaning that the value of T gets closer to $t/2$, our OT-MPSI$_{\text{individual}}$ converges into exponential runtime. Hence, the difference between the runtimes in Figure 6 and Figure 7. The results indicate that, in general, our OT-MPSI$_{\text{individual}}$ is not a practical solution in terms of efficiency.

Comparing the results for our OT-MPSI$_{\text{individual}}$ protocol and the naive one, we see that for $T = t - 1$, the overhead our security modifications generate is negligible. However, for $T = t/2$, as the number of parties grows, the runtime of our protocol is almost double the runtime of the naive one. This is explained by the fact that the underlying MPSI by Bay et al. scales linearly with the number of parties in terms of computational complexity. Since our OT-MPSI$_{\text{individual}}$ runs each intermediate MPSI between t parties, whereas the naive solution runs them between T parties, the security modifications of our protocol account for roughly $O(\frac{t}{T})$ times increase in the runtime complexity. Note that our OT-MPSI$_{\text{individual}}$ generates overhead for playing Mental Poker, but since we run it only once at the beginning, it is dominated asymptotically by the complexity of running all intermediate MPSIs.

6 Responsible Research

Our paper focuses on privacy-preserving technologies and preventing financial crime. We deem it highly unlikely that our protocol will be used maliciously by others. We do not use any sensitive data and do not conduct any user studies in our paper; therefore, these concerns are naturally inapplicable. In our experimental results, we ensure reproducibility by providing access to our code, which contains both the implementation of our protocol and our benchmarking, and by specifying the environment in which we run our experiments. We communicate our results with appropriate statistics and with enough detail, making it possible for others to evaluate the relevance of our findings. Apart from the visualisations, we also provide raw data in Appendix A.

7 Discussion

To our knowledge, the only other solution to the problem of transforming MPSI into T-MPSI is the naive solution, as mentioned by [13, 15]; therefore, we compare our findings with it.
In terms of security guarantees, our OT-MPSI$_{\text{individual}}$ protocol hides the owners of the items in the final intersection. The naive solution solves the variant of T-MPSI adopted by Mahdavi et al. [13], meaning it reveals both the item counts and their owners in the result. Therefore, our solution is an improvement in terms of security over the naive one.
When it comes to computational complexity, the naive solution requires $\binom{t-1}{T-1}$ runs of MPSI, each between T parties. Note that in both [13, 15] it is said that it requires $\binom{t}{T}$ runs, but it is implied that it is supposed to solve T-MPSI$_{\text{all}}$ variant [22], whereas we focused on T-MPSI$_{\text{individual}}$. Our solution requires $\binom{t-1}{T-1}$ runs of MPSI as well, but each between t parties. Since the underlying MPSI has linear computational complexity in the number of parties [11], we hide the participants at a cost $O(\frac{t}{T})$ times overhead for each MPSI run. Additionally, we add the overhead of playing Mental Poker.
Using dummy data structures and Mental Poker can be utilised in the same way to implement OT-MPSI$_{\text{individual}}$ from a similar MPSI protocol by Bay et al. [24] that uses Bit Vectors

instead of Bloom Filters. A Bit Vector represents a set as an array A of size $|U|$, where U is the universe that all items in the parties' sets belong to. We assume the items are ordered in U. The i^{th} item is in the set if and only if $A[i] = 1$. This protocol also operates in the client-server topology. Each party can construct a dummy Bit Vector that has all entries equal to 1, and proceed in the same way as in our protocol in Figure 5. The security analysis holds since the protocol from [24] also uses an IND-CPA secure encryption scheme. Since the protocol from [24] scales linearly with the number of parties, the complexity analysis is also analogous.
A clear limitation of our current analysis is the fact that we did not consider the network overhead in our experiments. In real-world scenarios, that would be a crucial factor to consider. An analysis of the network bandwidth would hence be needed to complete our results.
Another limitation is that our implementation uses a simple Mental Poker protocol [16]. It was very straightforward to implement; however, it is shown that it leaks partial information [21]. The motivation behind using it is that most out-of-the-box Mental Poker implementations we encountered were either not suited for research purposes or were not open-source. To implement our protocol in practice, a secure Mental Poker protocol [17, 18] should be used.

8 Conclusions and Future Work

Although MPSI and T-MPSI are closely related, the only solution to the problem is the naive one, which is neither efficient nor secure. In our paper, we introduce an OT-MPSI$_{\text{individual}}$ protocol that successfully improves the security of the naive solution. It allows running multiple MPSI instances subsequently, without revealing the participants of each instance. It does so with the collusion threshold of $l = t - 1$. Improved security comes at a cost of increased runtime, as each intermediate MPSI is executed between all parties. Although the security goal is achieved, our OT-MPSI$_{\text{individual}}$ still remains an impractical protocol. An experimental analysis shows that the runtime of our protocol depends exponentially on the number of parties in the worst case, making it perform worse than existing T-MPSI implementations [15]. Despite its shortcomings, we believe it is an important step towards making a secure and efficient transformation from MPSI to T-MPSI.

A natural research direction to follow would be to investigate how to make the protocol more efficient. It can be shown by a simple argument that, by just using MPSI, it is not possible to avoid running it for all subsets of parties of size T. Hence, research into combining it with other SMPC primitives may provide a solution. Before the efficiency problem is solved, this approach will remain impractical.
Another research direction would be to focus on making the protocol hide the count for each item in the result. Most T-MPSI formulations do not reveal it [7, 11, 24], as it can be treated as a security risk. Hence, to finish the security aspect of the transformation, it would be desirable to hide this data. This would require making sure that intermediate results are not directly available to the server.
Also, we base our protocol on a concrete MPSI implementation. Our security and complexity analyses stem from the properties of the underlying MPSI protocol. Future research should focus on formulating a general transformation from any MPSI protocol to T-MPSI.

References

[1] B. Unger, "Money laundering regulation: from Al Capone to Al Qaeda," in *Research Handbook on Money Laundering*, B. Unger and D. van der Linde, Eds. Cheltenham, U.K.: Edward Elgar Publishing, 2013, pp. 19–32. [Online]. Available: https://doi.org/10.4337/9780857934000.00009

[2] United Nations Office on Drugs and Crime, "Money laundering," [Online]. Available: https://www.unodc.org/unodc/en/money-laundering/overview.html. [Accessed Apr. 21, 2026].

[3] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L 119, pp. 1–88, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[4] A. C. Yao, "Protocols for secure computations (extended abstract)," in *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, 1982, pp. 160–164. [Online]. Available: https://doi.org/10.1109/SFCS.1982.38

[5] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: a system for secure multi-party computation," in *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, P. Ning, P. F. Syverson, and S. Jha, Eds. ACM, 2008, pp. 257–266. [Online]. Available: https://doi.org/10.1145/1455770.1455804

[6] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, ser. Lecture Notes in Computer Science, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 1–19. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_1

[7] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty PSI and extensions to circuit/quorum PSI," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 1182–1204. [Online]. Available: https://doi.org/10.1145/3460120.3484591

[8] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 1257–1272. [Online]. Available: https://doi.org/10.1145/3133956.3134065

[9] G. Garimella, B. Pinkas, M. Rosulek, N. Trieu, and A. Yanai, "Oblivious key-value stores and amplification for private set intersection," in *Advances in Cryptology -*

CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II, ser. Lecture Notes in Computer Science, T. Malkin and C. Peikert, Eds. Springer, 2021, pp. 395–425. [Online]. Available: https://doi.org/10.1007/978-3-030-84245-1_14

[10] O. Nevo, N. Trieu, and A. Yanai, "Simple, fast malicious multiparty private set intersection," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 1151–1165. [Online]. Available: https://doi.org/10.1145/3460120.3484772

[11] A. Bay, Z. Erkin, J. Hoepman, S. Samardjiska, and J. Vos, "Practical multi-party private set intersection protocols," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1–15, 2022. [Online]. Available: https://doi.org/10.1109/TIFS.2021.3118879

[12] L. Kissner and D. X. Song, "Privacy-preserving set operations," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, V. Shoup, Ed. Springer, 2005, pp. 241–257. [Online]. Available: https://doi.org/10.1007/11535218_15

[13] R. A. Mahdavi, T. Humphries, B. Kacsmar, S. Krastnikov, N. Lukas, J. A. Premkumar, M. Shafieinejad, S. Oya, F. Kerschbaum, and E. Blass, "Practical over-threshold multi-party private set intersection," in *ACSAC '20: Annual Computer Security Applications Conference, Virtual Event / Austin, TX, USA, 7-11 December, 2020*. ACM, 2020, pp. 772–783. [Online]. Available: https://doi.org/10.1145/3427228.3427267

[14] J. Vos, M. Conti, and Z. Erkin, "Sok: Collusion-resistant multi-party private set intersections in the semi-honest model," in *IEEE Symposium on Security and Privacy, SP 2024, San Francisco, CA, USA, May 19-23, 2024*. IEEE, 2024, pp. 465–483. [Online]. Available: https://doi.org/10.1109/SP54263.2024.00079

[15] C. Guan, "A comparative study of threshold multiparty private set intersection protocols," Master's thesis, Delft University of Technology, Delft, The Netherlands, 2024. [Online]. Available: https://resolver.tudelft.nl/uuid:425aff9c-a92c-4272-8674-33630fdf062a

[16] A. Shamir, R. L. Rivest, and L. M. Adleman, "Mental poker," in *The Mathematical Gardner*, D. A. Klarner, Ed. Boston, MA: Springer, 1981, pp. 37–43. [Online]. Available: https://doi.org/10.1007/978-1-4684-6686-7_5

[17] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information," in *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, Eds. ACM, 1982, pp. 365–377. [Online]. Available: https://doi.org/10.1145/800070.802212

[18] T. Wei and L. Wang, "A fast mental poker protocol," *J. Math. Cryptol.*, vol. 6, no. 1, pp. 39–68, 2012. [Online]. Available: https://doi.org/10.1515/jmc-2012-0004

[19] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970. [Online]. Available: https://doi.org/10.1145/362686.362692

[20] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Springer, 1999, pp. 223–238. [Online]. Available: https://doi.org/10.1007/3-540-48910-X_16

[21] D. Coppersmith, "Cheating at mental poker," in *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, ser. Lecture Notes in Computer Science, H. C. Williams, Ed. Springer, 1985, pp. 104–107. [Online]. Available: https://doi.org/10.1007/3-540-39799-X_10

[22] C. Guan, J. van Assen, and Z. Erkin, "Collective threshold multiparty private set intersection protocols for cyber threat intelligence," in *IEEE International Workshop on Information Forensics and Security, WIFS 2024, Rome, Italy, December 2-5, 2024*. IEEE, 2024, pp. 1–6. [Online]. Available: https://doi.org/10.1109/WIFS61860.2024.10810671

[23] Y. Lindell, "How to simulate it - A tutorial on the simulation proof technique," in *Tutorials on the Foundations of Cryptography*, Y. Lindell, Ed. Springer International Publishing, 2017, pp. 277–346. [Online]. Available: https://doi.org/10.1007/978-3-319-57048-8_6

[24] A. Bay, Z. Erkin, M. Alishahi, and J. Vos, "Multi-party private set intersection protocols for practical applications," in *Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021, July 6-8, 2021*, S. D. C. di Vimercati and P. Samarati, Eds. SCITEPRESS, 2021, pp. 515–522. [Online]. Available: https://doi.org/10.5220/0010547605150522

A Raw Data from the Experiments in Section 5

Table 2: Mean runtime comparison for $T = t - 1$ for our OT-MPSI$_{\text{individual}}$ protocol for different numbers of parties t and set sizes n. Runtime in milliseconds. Means are computed over 20 runs. We report $\pm\sigma$ for each mean.

Number of parties t	$n = 4$	$n = 16$	$n = 32$
5	104.25 ± 0.95	390.15 ± 5.15	826.05 ± 10.46
10	140.15 ± 1.52	548.30 ± 6.35	1,621.85 ± 102.38
15	314.20 ± 5.21	969.20 ± 77.27	1,922.30 ± 143.91
20	286.60 ± 7.83	1,391.05 ± 113.42	2,383.45 ± 210.61
25	493.15 ± 25.43	1,616.85 ± 147.71	3,475.20 ± 266.65
30	465.70 ± 41.31	2,049.90 ± 171.18	3,831.60 ± 330.40
35	467.55 ± 14.51	2,393.50 ± 218.73	4,614.35 ± 377.54
40	554.50 ± 11.95	2,639.55 ± 235.43	5,224.35 ± 419.14
45	533.80 ± 23.94	2,983.40 ± 257.12	5,758.25 ± 500.75
50	605.50 ± 9.66	3,086.25 ± 288.52	6,547.35 ± 553.18

Table 3: Mean runtime comparison for $T = t/2$ for our OT-MPSI$_{\text{individual}}$ protocol for different numbers of parties t and set sizes n. Runtime in milliseconds. Means are computed over 10 runs. We report $\pm\sigma$ for each mean.

Number of parties t	$n = 4$	$n = 16$	$n = 32$
5	695.20 ± 2.77	2,620.00 ± 13.16	5,351.90 ± 38.86
6	1,285.10 ± 9.45	5,015.10 ± 143.23	9,991.30 ± 253.23
7	3,169.30 ± 173.45	11,276.60 ± 410.69	22,814.30 ± 471.84
8	6,158.50 ± 338.91	22,967.40 ± 363.65	45,753.70 ± 434.14
9	13,327.80 ± 524.90	50,934.30 ± 471.62	101,968.00 ± 583.28
10	25,347.30 ± 335.38	98,898.80 ± 530.86	200,173.00 ± 1,016.75

Table 4: Mean runtime comparison for $T = t - 1$ for a naive OT-MPSI$_{\text{individual}}$ protocol for different numbers of parties t and set sizes n. Runtime in milliseconds. Means are computed over 20 runs. We report $\pm\sigma$ for each mean.

Number of parties t	$n = 4$	$n = 16$	$n = 32$
5	106.80 ± 1.08	400.80 ± 5.83	820.55 ± 10.58
10	137.85 ± 1.20	573.65 ± 5.11	1,383.40 ± 95.66
15	255.15 ± 10.95	1,097.10 ± 74.01	1,937.10 ± 151.68
20	274.20 ± 3.62	1,236.85 ± 110.28	2,377.90 ± 203.85
25	589.45 ± 34.94	1,606.35 ± 158.65	3,311.00 ± 264.65
30	483.80 ± 33.41	1,921.35 ± 173.68	3,832.90 ± 350.08
35	466.35 ± 6.12	2,386.45 ± 194.33	4,728.25 ± 416.13
40	477.20 ± 6.34	2,531.10 ± 211.05	5,221.10 ± 442.69
45	554.25 ± 9.52	3,145.95 ± 247.16	5,715.05 ± 487.48
50	665.10 ± 55.73	3,256.15 ± 285.06	6,670.05 ± 527.11

Table 5: Mean runtime comparison for $T = t/2$ for a naive OT-MPSI$_{\text{individual}}$ protocol for different numbers of parties t and set sizes n. Runtime in milliseconds. Means are computed over 10 runs. We report $\pm\sigma$ for each mean.

Number of parties t	$n = 4$	$n = 16$	$n = 32$
5	560.50 ± 2.73	1,992.70 ± 6.82	3,997.20 ± 12.16
6	1,005.90 ± 3.39	3,691.00 ± 14.48	7,409.40 ± 35.16
7	2,073.30 ± 8.95	7,639.60 ± 19.16	15,350.50 ± 38.33
8	4,011.30 ± 121.72	14,589.20 ± 117.99	29,480.50 ± 139.32
9	9,200.00 ± 504.31	30,962.70 ± 178.75	60,023.20 ± 214.16
10	16,379.20 ± 353.31	57,850.40 ± 81.12	114,253.00 ± 526.32