Mugla Journal of Science and Technology

NEW THRESHOLD PRIVATE SET INTERSECTION PROTOCOLS

ASLI BAY*, Department of Computer Engineering, Antalya Bilim University, Türkiye, asli.bay@antalya.edu.tr
(https://orcid.org/ 0000-0002-3820-1778)

Received: 07.11.2023, Accepted: 04.04.2024
*Corresponding author
Research Article
DOI: 10.22531/muglajsci.1387499

Abstract

With the rising amount of digital technologies that we use on a daily basis, it is more important than ever to handle and process private data securely. Research and academic communities are becoming increasingly interested in multi-party computation, with a focus on the field of Private Set Intersection (PSI). In this regard, this work introduces a novel technique that successfully converts the Cid-Davidson Private Set Intersection protocol into a Threshold Private Set Intersection. It achieves this conversion by introducing two new protocols, TPSI-1 and TPSI-2, and utilizing two previously developed methodologies while the Reed-Solomon codes and the Shamir-secret sharing scheme are the foundations of TPSI-1, whereas Secure Comparison Protocols serve as the foundation for TPSI-2. Specifically, our suggested protocols perform better asymptotically than previous threshold PSI protocols because they have a fixed number of rounds and linear communication and computation complexity that increase with data set size. This study adds to the continuous effort to strengthen the security and effectiveness of private data calculations, highlighting how safe data processing is changing in an era where digital technologies are ingrained in every aspect of our lives.

Keywords: Homomorphic Encryption, Multi-party Computation, Private Set Intersection, Secure Comparison Protocols, Secret Sharing

YENİ EŞİKLİ ÖZEL KÜME KESİŞİM PROTOKOLLERİ

Özet

Günlük olarak kullandığımız dijital teknolojilerin miktarının artmasıyla birlikte, özel verilerin güvenli bir şekilde ele alınması ve işlenmesi her zamankinden daha önemlidir. Araştırma ve akademik topluluklar, Özel Küme Kesişimi (PSI) alanına odaklanarak çok partili hesaplamaya giderek daha fazla ilgi duymaktadır. Bu bağlamda, bu çalışma Cid-Davidson Özel Küme Kesişimi protokolünü başarılı bir şekilde Eşik Özel Küme Kesişimi'ne dönüştüren yeni bir teknik sunmaktadır. Bu dönüşümü, TPSI-1 ve TPSI-2 olmak üzere iki yeni protokol sunarak ve daha önce geliştirilmiş olan iki metodolojiyi kullanarak gerçekleştirmektedir, Reed-Solomon kodları ve Shamir-gizli paylaşım şeması TPSI-1'in temellerini oluştururken, Güvenli Karşılaştırma Protokolleri TPSI-2'nin temelini oluşturmaktadır. Özellikle, önerdiğimiz protokoller, sabit sayıda tura ve veri kümesi boyutuyla artan doğrusal iletişim ve hesaplama karmaşıklığına sahip oldukları için asimptotik olarak önceki eşik PSI protokollerinden daha iyi performans göstermektedir. Bu çalışma, özel veri hesaplamalarının güvenliğini ve etkinliğini güçlendirmeye yönelik sürekli çabalara katkıda bulunmakta ve dijital teknolojilerin hayatımızın her alanına girdiği bir çağda güvenli veri işlemenin nasıl değiştiğini vurgulamaktadır.

Anahtar Kelimeler: Homomorfik Şifreleme, Çok Taraflı Hesaplama, Özel Küme Kesişimi, Güvenli Karşılaştırma Protokolleri, Gizli Paylaşım

Cite

Bay, A., (2024). "New Threshold Private Set Intersection Protocols", Mugla Journal of Science and Technology, 10(1), 51-60.

1. Introduction

Recent years have seen a lot of interest in private set intersection (PSI) protocoIs. In PSI protocols, two parties, denoted as P_1 and P_2, holding their private data sets S_1 and S_2 and compute their intersection, $S_1 \cap S_2$ without disclosing any additional information.

Due to significant advancements in digital technology, several applications have begun employing PSI protocols. For instance, for advertising efficiency, companies like Facebook utilize PSI protocols to compare customer merchant lists and adviser lists [1]. Another application is password check-up, which uses PSIs to ascertain whether the password has been compromised. In this way, users can compare their credentials with millions of entries in breached databases without revealing any part of their passwords.

In some scenarios, the intersection of two data sets can only be made public if its size is larger than or equal to a predetermined threshold value. This is called a threshold private set intersection (TPSI) in which $S_1 \cap S_2$ is privately computed if $|S_1 \cap S_2| \geq t$. For example, two bikers might want to share their routes if their routes contain identical segments, as in a real-world scenario

known as route discovery [2, 3]. They compare their routes, and if there are enough shared route segments, they disclose their routes to each other. Another situation is the usage of online dating services, where users hope to make acquaintances by disclosing personal information on social media websites [4]. A friend selection is better done methodically to make the relationship more realistic. In other words, friendship is feasible when there are enough shared interests between two people.

1.1 Related Work

Yao [5] offers the first secure multiparty computation (MPC) protocol to address the question of which of two millionaires is wealthier. Since then, MPC and in particular, private set intersection (PSI) has gained popularity among researchers. There are thousands of works that offer various PSI solutions. To the best of the authors' knowledge, PSI protocols can be categorized as follows:

- *Oblivious Polynomial Evaluation (OPE) based PSI:* In OPE problem, the sender's input is a polynomial P and the receiver's input is x. The receiver can take the value of $P(x)$ for any value of x without getting any other information about P or revealing any information about x to the sender. OPE was first proposed in PSIs by Freedman et al. [6] where data sets were defined by polynomials. Then, Kissner and Song [7] propose another solution which is also based on an additive homomorphic Public-key Encryption scheme. After, Camenish and Zaverucha [8] propose another OPE-based PSI that requires inputs to be approved by a third party. Hazay and Nissim then improve Freedman et al. protocol [6], however, both protocols only let one party compute the intersection (one-way). Contrary to [6], [7,8] are two-way PSIs which means that two parties learn the result of the intersection at the end. About their complexities, the aforementioned protocols do not have linear complexities.

- *Oblivious Pseudorandom Function (OPRF)* based *PSI:* A two-party OPRF-based PSI where the sender has a private key k and the receiver has a private input x evaluates a pseudorandom function (PRF) $f_k(x)$ privately. In this context, Hazay and Lindell [9] first propose a PSI protocol that only provides one-way computations of the intersection. Later Jarecki and Liu [10] extend the work of Hazay and Lindell with additive homomorphic public-key encryption and provide PSI in the malicious setting. Both constructions mentioned above are one-way and have linear complexity. Differently, the work by Debnath and Dutta [11] is a two-way PSI protocol that also having linear complexity.
- *Bloom filter-based PSI:* Bloom filters are probabilistic structures that compactly represent data sets using bits, often 0s and 1s. They are efficient to use, especially with bigger data sets. In [12] Bloom filters are first used and combined by AND operator. However, their method is not secure due to the privacy leakage of private set elements. Later, Kerchbaum [13] proposes a Bloom-Filter-based PSI solution that makes use of Goldwasser-Micali public-key encryption scheme [14]. Similarly, Dong et al. [15] combine (garbled) Bloom filters with oblivious transfer computation. Later, the protocol by Cid- Davidson that we consider in this work is also based on Bloom filters and an additive homomorphic public-key encryption scheme and has linear complexities. Other PSIs based on Bloom filters are in [16–18].

- *Bit-set data representation-based PSI:* in this representation, the sets are chosen from a fixed and ordered domain and represented with a fixed vector consisting of 0's and 1's. That is, the length of this vector is the same as the length of the domain. Ruan et al. [19] firstly make use of bit-set representation and additive homomorphic encryption to propose PSI and PSI-like protocols. Their protocols are more efficient than existing ones when the private data sets are small like with $\leq 2^{12}$ elements.

- *Others*: The intersection of two data sets can be found by using circuits to compare them. If a naive way is used, we require $\mathcal{O}(n^2)$ time complexity for comparison of the data sets. To reduce this complexity, some different techniques are proposed: a merge-sort network [20] requiring $\mathcal{O}(nlogn)$ comparisons. In [21], they combine the Cuckoo hashing with circuits to further reduce computational complexity. In [22], Pinkas et al. propose a new type of circuit-based protocol for computing PSI protocols that requires almost linear comparisons. Recently, Ruan-Mao [23] release the PSI protocol using point-value polynomial representation. This way, they avoid using encryption to secure data that makes their protocol very efficient.

Threshold Private Set Intersections (TPSI): Threshold PSI functionality differs from PSI in that a TPSI gives the intersection if the intersection's size is larger than or

Table 1. TPSIs in a semi-honest environment.

Protocol	Communication (bits)	Computation (operations)
[2]	$\mathcal{O}(\lambda n)$	$\mathcal{O}(n^2)$
[29]	$\mathcal{O}(\lambda n log(k+1))$	$\omega(log\lambda)\mathcal{O}(n)$
[25]	$\mathcal{O}(t)$	$\mathcal{O}((n-t)^4)$
[27]	$\mathcal{O}(\lambda nk)$	$\mathcal{O}(nlogn)$
TPSI-1 (Section 3.1)	$\mathcal{O}(log\|X\|kn)$	$\mathcal{O}(kn)$
TPSI-2 (Section 3.2)	$\mathcal{O}(max(log\|X\|kn, log\|X\|knlogn)$	$\mathcal{O}(max(nk, nlogn))$

equal to a specific positive integer known as a *threshold t.* To clarify more, the intersection $S_1 \cap S_2$ of S_1 and S_2 is released if $|S_1 \cap S_2| \geq t$. We direct the reader to Fig. 3 for a more comprehensive definition.

There are a few TPSI protocols, and the majority of them compute the intersection's cardinality first before deciding if it is larger than or equal to the threshold number [2, 4, 6, 22, 24]. Namely, Freedman et al. [6] use private matching to compute set cardinality for outputting the intersection if $|S_1 \cap S_2| \geq t$. Pinkas et al. [22] present a PSI-CAT protocol that outputs "1" if the intersection's cardinality exceeds a threshold t. Extending this to TPSI involves a minor update to their PSI-CAT circuit to include a key release condition based on intersection size, and utilizing this key for symmetric encryption of the final message in any linear-complexity PSI protocol. In this way, they can outperform the TPSI protocol in [2] which has quadratic computation complexity. They securely compute privacy-preserving ride-sharing functionality, however, their TPSI works for small data sets as the secret reconstruction requires quadratic time complexity.

As our protocols are two-party and their security is assumed in the semi-honest setting, we only compare our protocols with that of two-party TPSI in the semi-honest setting. Zhao and Chow [25] propose two different types of TPSIs such that the intersection is released when $|S_1 \cap S_2| \geq t$ or when $|S_1 \cap S_2| \leq t$, where S_1 and S_2 are private data sets of two parties. The advantage of implementing the two types of TPSI protocols is obvious when we consider the online dating platform. On this platform, when a user wants to match with other users of desired properties, $|S_1 \cap S_2| \geq t$ helps, and when a user wants to match with others who do not have some undesired properties, such as smoking or drinking alcohol, $|S_1 \cap S_2| \leq t$ works. Ghosh and Nilges [26] propose threshold multi-party PSI which is used in the malicious setting. Zhang et al. [27] propose a threshold scheme from the DCW protocol [15]. To achieve the threshold and hence to reduce the time complexity of reconstructing the secret, they make use of the Reed-Solomon coding algorithm. Their protocol supports larger data sets compared to previous TPSIs. Although their protocol's communication complexity is linear in the size of data sets n, its computational complexity is logarithmic in terms of n. Lastly, it is important to highlight that there are two generic Circuit-based PSI protocols referenced as [28] and [29], which possess the flexibility to be seamlessly transformed into TPSIs.

1.2. Contribution

This study makes valuable contributions to the fields of threshold private set intersection (TPSI) and secure data computing. We have extended the capabilities of the Cid-Davidson protocol by utilizing the techniques presented by Bay et al. [30] and Zhang et al. [27]. This has led to the development of two new secure TPSI protocols. These innovations not only expand the toolkit for safe data intersection, but also provide practitioners looking for secure and private data computing methods with useful options. The security proofs are based on simulations for both protocols, confirming their reliability in situations with semi-honest adversaries. Moreover, the comprehensive asymptotic complexity analysis shows that these procedures have remarkable computational efficiency. To summarize:

- We apply two previously known methods of Bay et al. [27] and Zhang et al. [30] to the Cid-Davidson protocol. Therefore, in this study, we present two variants of secure TPSIs based on the Cid-Davidson protocol.
- For both of our protocols, we provide their security analysis by providing a formal simulation-based security proof in the semi-honest adversary model.
- We present an asymptotic complexity analysis of our protocols. Compared to the previous designs depicted in the first three protocols in Table 1, our protocol is much faster in computation. Compared to the protocol of Zhang et al., our TPSI has a better computational cost when $k < logn$. However, we cannot analyze that much data in our protocols, as usually k is larger than $logn$, to be more precise, the complexity of our protocols and theirs are very similar.

The remainder of the paper is structured as follows: in Section 2, we provide preliminaries and notations. Our first novel TPSI protocol (TPSI-1) based on secret-sharing and the Reed-Solomon Codes is presented in Section 3.1. Later in Section 3.2, we propose our second novel TPSI protocol (TPSI-2) based on the Secure Comparison protocols. Finally, Section 4 is where we wrap up our work.

2. Preliminaries and Notations

Table 2 contains a list of the notations used throughout the text.

Table 2. Notations.

P_1	a client
P_2	a server
S_i	private data set of i-th party, $i \in \{1,2\}$
n_i	size of S_i, $i \in \{1,2\}$
n	$n = max\{n_1, n_2\}$
PSI	private set intersection
TPSI	threshold private set intersection
λ	A security parameter
S	intersection of S_1 and S_2 , $S = S_1 \cap S_2$
h_i	$h_i: \{0,1\}^* \rightarrow \{0,1\}^m, 1 \leq i \leq k$
pk, sk	a pair of public and private key
$seed_{P_i}$	a seed for database S_i, for $i \in \{1,2\}$
$seed$	a seed for PRNG, $seed = seed_{P_1} \oplus seed_{P_2}$
sh_j	j-th secret share of a secret s among $2n - t$ elements

2.1. Bloom Filters

A Bloom Filter is first proposed in [30] which is a way of doing efficient data representation. The formal definition of a Bloom Filter can be as follows:

Bloom Filter: A Bloom Filter encodes a data set S of maximum size n into a m bit string as $\mathbf{BF}[0], \dots, \mathbf{BF}[j], \dots, \mathbf{BF}[m-1]$. The elements of S are represented in a Bloom Filter with the help of k randomly chosen hash functions $(h_1, h_2, \dots, h_k)$, where $h_i: \{0,1\}^* \rightarrow [0,1, \dots, m-1]$. To insert every element of S into a Bloom Filter, first all bins (equivalently, indices) of the Bloom Filter is filled with 0, then for each element $x \in S$, we update the indices $h_1(x), h_2(x), \dots h_k(x)$ to 1. If the bin already stores 1, then it will remain as 1. The details can be found in [32,33].

In the Cid-Davidson PSI protocol [34], they use the inverted and encrypted Bloom filters as follows:

Definition 1. *If* **BF** *is a Bloom filter representing the data set S, then the inverted Bloom filter of* **BF** *is* **IBF**, *where* $\mathbf{IBF}[j] = \mathbf{BF}[j] + 1 \mod 2$.

Definition 2. *If* **BF** *is a Bloom filter representing the data set S, then the encrypted Bloom filter is defined as* $\mathbf{EBF}[j] = E(\mathbf{BF}[j])$.

2.2. Shamir Secret Sharing Scheme and the Reed-Solomon Codes

The creation of contemporary cryptographic protocols can benefit greatly from the usage of secret-sharing systems. A secret s is divided into n shares in a secret-sharing scheme so that the secret can be reconstructed if enough shares, more than a threshold number t, are joined. One of the very useful schemes belongs to [35] called *(t,n)*-Shamir Secret Sharing Scheme (SSS). The Lagrange interpolation Theorem serves as the foundation for SSS, where the threshold value is t and $t \leq n$. The ideal functionality $\mathcal{F}_{SSS}$ for SSS is shown in Fig. 1.

Reed-Solomon codes [36] are a family of error-correcting codes that are widely used in digital communications and storage, as they admit efficient encoding/decoding algorithms. The Reed-Solomon decoding algorithms can be used to find the shared secret of SSS when there exist errors in some of the shares. The decoding algorithm based on Fourier Transforms proposed in [37] will be considered in our protocol.

Parameters: Two integers are n and t such that $n < t$ and F is the finite field. Let $ind_1, ind_2, \dots, ind_n$ be fixed and distinct points from F.

Secret Sharing: An input s from F is shared with a random polynomial f of degree t such that $f(0) = s$ and the shares of s is $O = \{f(ind_1) = sh_1, \dots, f(ind_n) = sh_n\}$.

Secret Reconstruction: For a subset of $R \subset O$ of size $t+1$, by Lagrange interpolation formula, the polynomial f is reconstructed as f' such that $f'(ind_j) = sh'_j$ and $f'(0) = s'$ where $sh'_i \in R$

Figure 1. Ideal Functionality $\mathcal{F}_{SSS}$ for SSS.

2.3. Secure Comparison Protocols

A secure comparison protocol (SCP) is a secure way of comparing two private numbers in such a way that there are two parties, P_1 and P_2, and their respective private integers are x and y, respectively, and both parties want to find out which of their integers is the largest while keeping its value a secret from the other. Yao [38] first describes this issue in the literature, known as the millionaires' problem.

Veugen et al. [39] propose using different SCPs in the setting mentioned above where the compared numbers are encrypted and held by only one party. One of their protocols is composed of three phases: (1) they first transform their encrypted inputs x and y into two privately held inputs c and r, (2) then they employ an SCP which outputs the secret shares of the comparison result of these privately shared c and r, (3) they then use another transformation to find the encrypted result of the comparison x and y. Note that in [39], for the second step they utilize three different protocols based on homomorphic encryption (Pailler PKE) or linear secret sharing. For our TPSI-2, we choose the NO protocol [40] as it has constant round complexity.

The NO Secure Comparison Protocol, having $\mathcal{O}(l)$ secure multiplications and a constant number of rounds which is 7, is in charge of the protocol's overall complexity. Note that each secure shared multiplication has three local multiplications for two parties and three ciphertexts are communicated. Also, in shared multiplication for the precomputation phase, one has to make two encryptions and one decryption. Therefore, one execution of the protocol needs the communication of $\mathcal{O}(l)$ ciphertexts and the same number of encryptions as time complexity, where l is the bit length of x and y. Due to the page limitation, we skip the details of the protocol which can be found in [39].

2.4. Private Set Intersection (PSI) and Threshold PSI (TPSI)

In PSIs, there are two parties P_1 and P_2 having respective private data sets S_1 and S_2 wanting to compute the intersection $S_1 \cap S_2$ without disclosing any additional elements. On the other hand, TPSI provides a feature that allows the intersection to be calculated by the server when the intersection size meets $|S_1 \cap S_2| \geq t$, where t is the threshold parameter. For these protocols, their functionality is given in Fig. 2 and Fig. 3, respectively.

Parameters: The server P_2 and the client P_1 have two secret data sets $S_1 = \{x_1, x_2, \dots, x_{n_1}\}$ and $S_2 = \{y_1, y_2, \dots, y_{n_2}\}$ of the sizes n_1 and n_2, respectively, λ is the security parameter.
Input: Wait the inputs from both the server P_2 and the client P_1.
Computation: Ideal functionality $\mathcal{F}_{\text{PSI}}$ computes $S_1 \cap S_2$.
Output: $\mathcal{F}_{\text{PSI}}$ sends the result to the server.

Figure 2. Ideal Functionality $\mathcal{F}_{\text{PSI}}$ for PSI.

Parameters: The server P_2 and the client P_1 have two secret data sets $S_1 = \{x_1, x_2, \dots, x_{n_1}\}$ and $S_2 = \{y_1, y_2, \dots, y_{n_2}\}$ of the sizes n_1 and n_2, respectively, λ is the security parameter, t is the threshold number.
Input: Wait the inputs from both the server P_2 and the client P_1 and also wait for the threshold t.
Computation: Ideal functionality $\mathcal{F}_{\text{TPSI}}$ computes $S_1 \cap S_2$ and $|S_1 \cap S_2|$.
Output: $\mathcal{F}_{\text{TPSI}}$ sends the result to the server if $|S_1 \cap S_2| \geq t$. Otherwise the server obtains nothing.

Figure 3. Ideal Functionality $\mathcal{F}_{\text{TPSI}}$ for TPSI.

2.5. Security Setting

The security of the threshold version of the Cid-Davidson PSI protocol that is proposed in this work is again assumed in the *semi-honest security* model. The parties adhere to the protocol in this security model honestly and do not stray from it. Additionally, we consider the corrupted party as *a static adversary* where the number of corrupted parties is fixed before the protocol starts.

2.6. Additive Homomorphic Encryption

A public key encryption system is defined as being additively homomorphic if $E_{pk}(M_1 + M_2)$ can be efficiently computed using only two ciphertexts, $c_1 = E(M_1)$ and $c_2 = E(M_2)$, without having access to the secret key. That is,

$$D_{sk}\left(E_{pk}(M_1) +_H E_{pk}(M_2)\right) = M_1 + M_2,$$
$$and\ D_{sk}\left(\alpha E_{pk}(M_1)\right) = \alpha M_1,$$

where α is an arbitrary scalar value.

Finally, a ReRand function, which enables us to rerandomize the ciphertext by having only the public key, is a valuable approach that we employ in this study for an additively homomorphic scheme.

2.7. The Cid-Davidson PSI protocol

The Cid-Davidson PSI protocol [34] is a Bloom Filter-based PSI protocol using additively homomorphic PKI. Parties namely P_1 and P_2 agreed on k hash functions $\{h_1, \dots, h_k\}$. P_1 has a (pk, sk) key pair for the homomorphic PKI and let pk be available to P_2. They jointly compute the intersection of their private sets S_1 and S_2 without any information leakage except the intersection and the size of the sets which are given to the parties in advance. The protocol steps are enumerated as follows.

1. P_1 computes the corresponding Bloom filter $\mathbf{BF_1}$ as a representation of his data set S_1 and invert them to obtain $\mathbf{IBF_1}$.
2. P_1 encrypts $\mathbf{IBF_1}$ and gets $\mathbf{EIBF_1}$. He then sends $\mathbf{EIBF_1}$ to P_2.
3. P_2 computes k hash values of each element $y_i \in S_2$, and for each i, obtains$\{C_1^{(j)}, \dots, C_k^{(j)}\}$, where $C_d^{(j)} = \mathbf{EIBF_1}[h_d(y_j)]$ for all $j \in \{1, \dots, n_2\}$.
4. P_2 computes $c_j = C_1^{(j)} +_H \cdots +_H C_k^{(j)}$ and randomizes it by $r_j \cdot c_j$, where $r_j \in {}_{\text{R}}\mathbb{Z}_{\text{N}}$.
5. Then, P_2 computes $\tilde{p}_j = ReRand((r_j \cdot c_j) +_H E_{pk}(y_j))$ and $\tilde{c_j} = \text{ReRand}(c_j)$ and sends $(\tilde{p}_j, \tilde{c_j}), j \in \{1, \dots, m\}$ to P_1.
6. P_1 first checks whether $D_{sk}(\tilde{c_j}) = 0$. If yes, he adds $y_j = D_{sk}(\tilde{p}_j)$ to the intersection S.

To make the protocol visualized, the depiction of the protocol is given in Fig. S1.

3. The Proposed TPSI Protocols

This section includes our proposed threshold private set intersection protocols namely TPSI-1 and TPSI-2.

3.1 TPSI-1 by Secret-Sharing and Reed-Solomon Codes

TPSI stipulates that the intersection can only be obtained by the server (or client) if its size is larger than or equal to a threshold value *t*. (see Fig. 3). To achieve threshold intersection from the Cid-Davidson protocol, we make use of both the Reed-Solomon Codes and the Shamir-Secret Sharing Scheme. A similar method is also used in [27], where they transfer the PSI protocol of DCW [15] into TPSI. Although the idea is similar, its application is different due to the differences between the DCW and the Cid-Davidson protocols. Due to page restrictions, we recommend users visit [27] for the details of their protocol.

Our protocol relies on a secret sharing mechanism to determine the relationship between the intersection of two datasets and a threshold value, denoted as *t*. This process guarantees that there are enough valid shares to reconstruct the secret, denoted as s. Thus, the client can calculate the polynomial of the secret sharing scheme as long as the intersection contains a sufficient number of elements. Access to the intersection is granted only

when the necessary criteria are met. Additionally, our approach incorporates the Reed-Solomon decoding algorithm to reconstruct the secret, bypassing the need to compute every potential combination of shares.

By adapting the Cid-Davidson PSI protocol into a TPSI protocol, we will demonstrate how we design TPSI-1 in the manner that is described below.

Input: The client P_1 has a pair of public-key and a secret-key (pk, sk) of an additively homomorphic PKI and the server P_2 is only given to pk. The size of the private sets has to be the same size which is $|S_1| = |S_2| = n$.

Initialization: A set of hash functions $\{h_1, h_2, \dots, h_k\} \in \mathcal{H}$ is randomly chosen and sent to the client P_1 by the server P_2. By using the functionality of SSS, the server generates $sh_j's$ where $f(ind_j) = sh_j$ for $j \in \{1, \dots, 2n - t\}$.

Dummy Variable Generation: To produce the same $d = n - t$ dummy variables, P_1 and P_2 individually generate two random seeds $seed_{P_1}$ and $seed_{P_2}$ of length λ, respectively, namely $seed_{P_1}$, $seed_{P_2} \in \{0,1\}^{\lambda}$, respectively. They then exchange and combine them as $seed = seed_{P_1} \oplus seed_{P_2}$ to agree on the same seed for the random number generator. Afterward, P_1 and P_2 individually compute $d = n - t$ number of (same) dummy elements and add them to their data sets S_1 and S_2 to obtain updated data sets S_1' and S_2'. Here, $S_1' = \{x_1, x_2, \dots, x_n, x_{n+1}, \dots, x_{2n-t}\}$ and $S_2' = \{y_1, y_2, \dots, y_n, y_{n+1}, \dots, y_{2n-t}\}$.

Local EIBF generation:

The client P_1

- represents his data sets S_1' as a Bloom filter $\mathbf{BF_1}$ and computes $\mathbf{IBF_1}$.
- computes the encrypted inverted Bloom filter $\mathbf{EIBF_1}$ and sends it to P_2.

Set Intersection computation:

- The server P_2 computes k hash values of each element $y_i \in S_2$ and generates $\{C_1^j, \dots, C_k^j\}$, where $C_d^j = \mathbf{EIBF_1}[h_d(y_j)]$ for all $j \in \{1, \dots, 2n - t\}$ and $d \in \{0, \dots, k\}$.
- P_2 computes $C_j = C_1^j +_H \cdots +_H C_k^j$ and randomizes it by $r_j \cdot C_j$, where $r_j \in {}_R\mathbb{Z}_N$.
- P_2 calculates g_j as $ReRand(r_j \cdot C_j) +_H E(sh_j || ind_j)$. He applies the first λ-bit of $sh_j || ind_j$ to sh_j, where the last λ-bit corresponds to ind_j.
- P_2 sends g_j to P_1, where $1 \le j \le 2n - t$.
- P_1 decrypts g_j's and obtains a secret s' by the Reed-Solomon Decoding Algorithm [35]. He then computes and sends $E(s')$ to P_2.
- P_2 first encrypts his secret s as $E(s)$ and then homomorphically computes $E(s' - s)$, which can be done by finding multiplicative inverse $E(s)^{-1}$ of $E(s)$. He then computes

$p_j = ReRand((r_j \cdot C_j) +_H E(y_j) +_H E\left(r^{(j,1)} \cdot (s' - s)\right))$ and $C_j' = ReRand((C_j) +_H E\left(r^{(j,2)} \cdot (s' - s)\right))$.

- For each j, P_1 first determines whether $D(C_j') = 0$. If so, he extends the intersection S by adding $y_j = D(p_j)$.

To ensure effective error correction with Reed-Solomon coding, dummy elements must differ from private data. For security reasons, the size of dummy elements matches that of private data. Given the large domain of private data sets compared to the number of symbols (n), the probability of collisions between dummy and private elements is small. Consider a domain size of 2^{128}. In this scenario, the probability of collision can be expressed as $n(n - t)/2^{128}$. For a single dummy element, the likelihood of collision with a set of n private elements is $n/2^{128}$. When considering $n - t$ dummy elements, the overall probability becomes $n(n - t)/2^{128}$. Refer to Fig. S2 for a visual depiction of the protocol.

Correctness: The protocol works correctly because it fully adopts the Cid-Davidson PSI protocol and relies on the correct use of the Reed-Solomon code. Indeed, the secret s' is constructed correctly that is $s = s'$ if and only if y_i is in the intersection of S_1 and S_2 and $|S_1 \cap S_2| \ge t$. Otherwise, the client will see nothing.

3.1.1 Security Proof

On the basis of the assumption that there exists a IND-CPA-secure [41] additively homomorphic PKE scheme π, the security of the protocol is going to be demonstrated. We consider the corrupted protocol participant as the adversary who is curious but honest. Before starting the proof, we provide some notations: let the input of the client be $Inp_1 = (S_1, |S_2|, pk, sk)$, and the server be $Inp_2 = \left(S_2, |S_1|, pk, \{sh_j || ind_j\}_{j=1}^{2n-t}\right)$. The output of the client P_1 is $S = S_1 \cap S_2$, and of the server is nothing($\emptyset$). We will consider two scenarios: (1) P_2 is honest, and P_1 is corrupted, (2) P_2 is corrupted, and P_1 is honest.

In the first scenario, the simulator $\mathcal{S}$ is given Inp_1 and the output of the protocol $S = S_1 \cap S_2$ and he needs to simulate the honest server towards the corrupted client. The simulator $\mathcal{S}$ starts with generating a simulated $(\tilde{g}_1, \dots \tilde{g}_{2n-t})$ view which is indistinguishable from the real $(g_1, \dots, g_{2n-t})$ one. $\mathcal{S}$ first chooses a random input for the honest server that complies with what the corrupted party knows the public hash functions $h_1, \dots, h_k$, and the size of the private set of the honest server. The simulated output has to agree with the output of P_1. Therefore, the simulator starts by choosing set $\tilde{S}_2$, such that $S_1 \cap \tilde{S}_2 = S$,. He then chooses $\widetilde{seed}$ and $\tilde{s}$ for producing inputs and outputs of SSS. $\mathcal{S}$ produces dummy elements and random input elements for $\tilde{S}_2'$. Then $\mathcal{S}$ follows the protocol and produces $(\tilde{g}_1, \dots \tilde{g}_{2n-t})$ from the EIBF of the corrupted client and random shares of SSS. He then follows the protocol and generates simulated tuple $\{(\tilde{p}, \tilde{C}_j')\}_{j=1}^{n_2}$Assume that there is an adversary Adv who can distinguish between $(\tilde{g}_1, \dots \tilde{g}_{2n-t})$ and $(g_1, \dots, g_{2n-t})$ and $\{(\tilde{p}, \tilde{C}_j')\}_{j=1}^{n_2}$ and $\{(p_j, C_j')\}_{j=1}^{n_2}$ with non-negligible probability. Hence, each of $g_i, \tilde{g}_i, (\tilde{p}, \tilde{C}_j')$ *and* $(p_j C_j')$ look like fresh encryptions of IND-CPA secure scheme Π due to the fact that they are

rerandomized homorphic sums. Therefore, one can easily built an adversary Adv_{Π} from Adv breaks the IND-CPA-security of Π.

In the second scenario where the server P_2 is corrupted and the client P_1 is honest, the simulator $\mathcal{S}$ is given to the input of the server Inp_2 and no output (as the server produces has no output.) The simulator $\mathcal{S}$ generates a random input set for the honest client P_1. Similarly, he chooses a random $\widetilde{seed}$ to generate and S_1' and S_2', and selects a random share $\tilde{s}'$ for P_1. He generates simulated $\widetilde{\mathbf{EIBF_1}}$ by encrypting it with pk. He then follows the protocol and generates the simulated view $E_{pk}(\tilde{s}')$. Similarly, to the first case, distinguishing $\widetilde{\mathbf{EIBF_1}}$ and $E_{pk}(\tilde{s}')$ from the real view helps to break the IND-CPA-security of Π.

3.1.2. Complexity Analysis

Communicational Complexity: In TPSI-1 protocol, there is a constant number of rounds. In the set-up phase, each party sends their seed $seed_{P_i}$ of size λ-bit to the other party. During the execution of the protocol data sent is dependent on the size of the Bloom Filter which is m.

The communication complexity of the execution of the protocol can be computed as follows: in the first round, P_1 sends the encrypted Bloom filter to P_2. Note that the encrypted Bloom filter has the size of ciphertexts. We know that for an optimal Bloom filter size, m has to be at least $knloge$ bit-long. Therefore, to do a fair comparison with other protocols, the complexity can be defined in terms of the size of the data set n, hence we say P_1 sends $\mathcal{O}(kn)$ciphertexts to the server P_2, where k is a number of hash functions. In the second and the last rounds, the server sends $2n-t$ and $2(2n-t)$ ciphertexts to the client, respectively. In the third round, the client sends a ciphertext to the server. Hence, while the server's communicational complexity is $\mathcal{O}(n)$ciphertexts, the client's one dominates the server's and is $\mathcal{O}(kn)$ ciphertexts. Note that to write in bits, we multiply the number of ciphertext with $\log|X|$.

Computational Complexity: The computational complexity for the client P_1 is in the first round is $\mathcal{O}(kn)$ encryptions and hash computations. In the third round he first decrypts $2n-t$ ciphertexts. Then he applies the Reed-Solomon decoding algorithm for $2n-t$ codewords which requires $\mathcal{O}(nlogn)$ multiplications. In the final part he does $\mathcal{O}(n)$ decryptions. Therefore, the dominated complexity of the client $\mathcal{O}(kn)$ encryptions. On the server-side, P_2 computes k hash evaluations to retrieve the values from the encrypted Bloom filter which needs $\mathcal{O}(kn)$ hash computations. P_2 then makes $\mathcal{O}(nk)$ homomorphic additions. In round 3, he then computes, $\mathcal{O}(n)$homomorphic additions and one inverse operation. Hence the computational complexity of the server is $\mathcal{O}(kn)$ homomorphic additions.

Note that in this work we consider the concrete instantiation from the Paillier or Elgamal schemes, where multiplication in $\mathcal{Z}_N$ is the homomorphic addition.

3.2. TPSI-2 by Secure Comparison Protocols

By calling a Secure Comparison protocol (SCP) [40] as a sub-protocol, we have upgraded the Cid-Davidson PSI protocol in TPSI-2 protocol for the threshold feature. In this scenario, we employ an SCP (Secure Computation Protocol) to evaluate the presence of an adequate quantity of items at the intersection. Essentially, the SCP runs first in parallel for each element within S_1, determining its intersection status. Utilizing the homomorphic property of encryption, the sum is computed without disclosing any specific data, and then compared by the SCP against a predetermined threshold value. Subsequently, the encrypted result of the SCP acts as a mask over the intersection, only revealing it if the count of intersecting elements meets or exceeds the threshold.

For Multi-party TPSI of the Cid- Davidson Protocol, [30] also makes use of a related concept. The functionality of the TPSI-2 protocol is going to be clarified as follows. Prior to adding the following steps, the same steps $1 \sim 4$ from the original Cid-Davidson Protocol (see Section 2.7) are first carried out. Afterwards,

- P_2 initializes simultaneous n_2 SCPs to check the decryption of C_j's is 0 (smaller than 1) or not, for every y_j. That is, P_2 gets the results $E(\alpha_j)$'s from SCPs. In this case, $E(\alpha_j)$ is the encryption of 1 if the decrypted value of $\tilde{c}_j = \text{ReRand}(C_j)$ is smaller than 1 and will be the encryption of 0 if the decrypted value of $\tilde{c}_j$ is bigger than or equal to 1.
- P_2 computes $E(\alpha) = \text{ReRand}(E(\alpha_1)+_H \cdots +_H E(\alpha_{n_2}))$. Note that according to the SCP, the output of the protocol will only be obtained by P_2.
- P_2 runs once more a SCP with P_1 and checks $E(\alpha)$ to a fresh encryption $E(t)$. In this case, the output $E(\beta)$ is obtained where β is 1 if $\alpha < t$, or it is 0 if $\alpha \geq t$.
- P_2 computes $\tilde{p}_j = \text{ReRand}((r_j \cdot C_j)+_H E(y_j)+_H (r_j' \cdot E(\beta)))$ and $\widetilde{TC}_j = ReRand(C_j+_H(r_j' \cdot E(\beta))$ and sends $(\tilde{p}_j, \widetilde{TC}_j), j \in \{1, \dots, n_2\}$ to P_1.
- Then P_1 decrypts $(\tilde{p}_j, \widetilde{TC}_j)$, and decides the intersection by first checking whether $D(sk, \widetilde{TC}_j) = 0$. If yes, he adds $y_j = D(sk, \tilde{p}_j)$ to the intersection S $j \in \{1, \dots, n_2\}$. Otherwise, P_1 outputs $\perp$.

Please refer to Fig. S3 for an illustration of the protocol.

Correctness: TPSI-2 perfectly outputs that y_i is in the intersection when the total number of intersection elements α satisfies $\alpha \geq t$. When this is satisfied, β becomes 0. The rest continues as the Cid-Davidson PSI. In the other case, the intersection will be empty.

3.2.1. Security Proof

The security of the TPSI-2 protocol will be shown under the presumption that there is an additively homomorphic PKE scheme Π that is IND-CPA-secure and a secure SCP in the semi-honest setting. Similar to the

proof of TPSI-1, we offer the following notations in advance: Let the server's input be $Inp_2 = (S_2, |S_1|, pk)$ and the client's input be $Inp_1 = (S_1, |S_2|, pk, sk)$, the output of the server P_2 is nothing ($\emptyset$), whereas the output of the client P_1 is $S = S_1 \cap S_2$. We think about two scenarios: (1) P_2 is honest, and P_1 is corrupted, (2) P_2 is corrupted, and P_1 is honest.

In the first scenario, the simulator $\mathcal{S}$ is required to simulate the honest server in relation to the corrupted client. He is supplied Inp1 and the result of the protocol $S = S_1 \cap S_2$. The corrupted client receives simulated $\tilde{c}^j$ as input in the SCP protocol, which the simulator must first build. Therefore, the honest server's input is first chosen at random by $\mathcal{S}$ in order to satisfy the corrupted party's knowledge of the public hash functions $h_1, \dots, h_k$ and the size of the honest server's private set. Also, the simulated output and the output of P_1 must coincide. The simulator then begins by selecting set S, $\tilde{S}_2$, where $S_1 \cap \tilde{S}_2 = S$. Then $\mathcal{S}$ adheres to the protocol and generates $\tilde{c}^j$ from EIBF. Then $\mathcal{S}$ simulates the conduct of the honest parties throughout the first execution of the SCP procedure. Essentially, there is one comparison against a brand-new encryption of $E(1)$ using an instantaneous SCP for each $\tilde{c}^j$. Here, $\mathcal{S}$ participates in the execution of the SCP protocol by simulating the role of the honest parties. Then, he combines the result of SCPs and participates once again in the SCP to compare the $E(\alpha)$ with $E(t)$. He then follows the protocol to compute the simulated $\tilde{p}^j$ and $\widetilde{TC}^j$ for all j. Finally, he decrypts the results by P_1's sk to obtain S. Everything that the corrupted parties receive from the simulator is rerandomized, making it impossible to tell it apart from new encryption. As a result, the underlying IND-CPA security can be compromised by an opponent who can tell the difference between the real run and the simulation run.

The simulator is provided the input of the server Ind_2 and no output in the second scenario, where the server P_2 is corrupted and the client P_1 is honest (as the server produces has no output). As in the security proof of TPSI-1, for the honest client P1, the simulator creates a random input set $\tilde{S}_1$ for S_1. He generates simulated $\widetilde{EIBF}_1$ by encrypting it with pk. In a manner similar to the first part, distinguishing $EIBF_1$ from the real view aids in compromising Π's IND-CPA-security.

3.2.2. Complexity Analysis

Communication Complexity: The number of rounds in the second TPSI is also constant. In the first round, P_1 sends the encrypted Bloom filter to P_2 of size $\mathcal{O}(kn)$ ciphertexts, where $(n = max(n_1, n_2))$. There are n_2 (equivalently n) parallel SCPs in Round 2, and one SCP in Round 3. The communicational complexity of SCPs will be $\mathcal{O}(nl)$ ciphertexts. Here, as l is the bit-length of threshold value t which has to be less than n, one can safely replace l with $\log_2 n$ as $t < n$. In the last round, $\mathcal{O}(n)$ ciphertexts are sent to P_1. Therefore, communication complexity of the protocol is $\mathcal{O}(\max(kn, nlog_2 n))$ ciphertexts.

Computational Complexity: In the first round P_1 makes k hash values of his elements to build his Bloom filter then encrypts it. Hence, this round requires $\mathcal{O}(\text{kn})$ hash computations and encryptions to build and compute his encrypted Bloom filter. Then, in the second round, P_2 computes k hash evaluations to retrieve the values from the encrypted Bloom filter which needs $\mathcal{O}(kn)$ hash computations. P_2 then makes $\mathcal{O}(kn)$ homomorphic additions. In rounds 3 and 4, there are $n + 1$ executions of SCP which needs $\mathcal{O}(nlog_2 n)$ encryptions. In the final round, there is the decryption of $\mathcal{O}(n)$ ciphertexts. Hence total computational complexity is $\mathcal{O}(\max(kn, nlog_2 n))$ encryptions.

4. Conclusions and Future Works

This work has advanced the field of threshold private set intersection (TPSI) protocols, which has attracted a lot of interest due to the growing need for secure applications. The research has incorporated a threshold feature into the Cid-Davidson PSI protocol, resulting in the development of two new protocols: TPSI-1 and TPSI-2. TPSI-2 utilizes Secure Comparison Protocols, while TPSI-1 is founded on Shamir-Secret Sharing and Reed-Solomon Coding. These protocols stand out due to their performance, which exceed those that are currently in the literature. They achieve both computing and communication complexity that scales linearly with the size of the data sets, and they notably display a constant number of rounds, which greatly improves their practical utility in safe data intersection applications.

Additionally, the Cid-Davidson protocol is successfully enhanced with threshold functionality by the study, motivating further research into the viability of comparable improvements for other effective PSI procedures. Hence, our future study aims to expand the set of secure data intersection protocols by including threshold functionality when appropriate.

5. References

[1] "The Disconcerting Details: How Facebook Teams Up With Data Brokers to Show You Targeted Ads." https://www.eff.org/deeplinks/2013/04/disconcerting-details-how-facebook-teams-data-brokers-show-you-targeted-ads, 2013. Accessed: 2021-11-25.

[2] Hallgren, P., Orlandi, C. and Sabelfeld, A., "Privatepool: Privacy-preserving ridesharing," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 276–291, 2017.

[3] Sherif, A. B. T., Rabieh, K., Mahmoud, . M. E. A. and Liang, X., "Privacy-preserving ride sharing scheme for autonomous vehicles in big data era," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 611–618, 2017.

[4] Zhao, Y. and. Chow, S. S. M, "Can you find the one for me? privacy-preserving matchmaking via threshold psi." Cryptology ePrint Archive, Report 2018/184, 2018. https://ia.cr/2018/184.

[5] Yao, A. C.-C. , "Protocols for secure computations (extended abstract)," in *FOCS*, pp. 160–164, IEEE Computer Society, 1982.

[6] Freedman, M. J. , Nissim, K. and Pinkas, B. , "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(C.Cachin and J. Camenisch, eds.), vol. 3027 of Lecture Notes in Computer Science, pp. 1–19, Springer, 2004.

[7] Kissner, L. and Song, D. X., "Privacy-preserving set operations," in Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings (V. Shoup, ed.), vol. 3621 of Lecture Notes in Computer Science, pp. 241–257, Springer, 2005.

[8] Camenisch, J., and Zaverucha, G. M., "Private intersection of certified sets," *in Financial Cryptography and Data Security* (R. Dingledine and P. Golle, eds.), (Berlin, Heidelberg), pp. 108–127, Springer Berlin Heidelberg, 2009.

[9] Hazay, C., and Lindell, Y., "Efficient protocols for set intersection and pattern matching with Security against malicious and covert adversaries," in *Theory of Cryptography* (R. Canetti, ed.),(Berlin, Heidelberg), pp. 155–175, Springer Berlin Heidelberg, 2008.

[10] Jarecki, S. and Liu, X., "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," in *Theory of Cryptography* (O. Reingold, ed.), (Berlin, Heidelberg), pp. 577–594, Springer Berlin Heidelberg, 2009.

[11] Debnath, S. K. and Dutta, R.,"Towards fair mutual private set intersection with linear complexity," Security and Communication Networks, vol. 9, no. 11, pp. 1589–1612, 2016.

[12] Burkhart, M. and Fontas, X. D. , "Fast private set operations with sepia," 2012.

[13] Kerschbaum, F., "Outsourced private set intersection using homomorphic encryption," in *7th ACM Symposium on Information, Compuer* and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012 (H. Y. Youm and Y. Won, eds.), pp. 85–86, ACM, 2012.

[14] Goldwasser S. and Micali, S.,"Probabilistic encryption*," Journal of Computer and System Sciences* vol. 28, no. 2, pp. 270–299, 1984.

[15] Dong, C., Chen, L. and Wen, Z. , "When private set intersection meets big data: an efficient and scalable protocol," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013* (A. Sadeghi, V. D. Gligor, and M. Yung, eds.), pp. 789–800, ACM, 2013.

[16] Kiss, A., Liu, J. Schneider, T., Asokan, N. and Pinkas, B., "Private set intersection for unequal set sizes with mobile applications," *Proceedings on Privacy Enhancing Technologies,* vol. 2017, no. 4, pp. 177–197, 2017.

[17] Debnath, S. K. and. Dutta, R ,"Efficient private set intersection cardinality in the presence of ma licious adversaries," in *Provable Security* (M.-H. Au and A. Miyaji, eds.), (Cham), pp. 326–339, Springer International Publishing, 2015.

[18]. Zhang, X, Zhu, H., Chen, M., Sun, M. , Liao, X. and Hu, L., "Outsourcing set intersection computation based on bloom filter for privacy preservation in multimedia processing," *Secur. Commun. Networks*, Hindawi, vol., 2018.

[19] Ruan, O., Wang, Z., Mi, J. and Zhang, M., "New approach to set representation and practical private set-intersection protocols," *IEEE Access*, vol. 7, pp. 64897–64906, 2019.

[20] Huang, Y.,Evans, D. and Katz, J. "Private set intersection: Are garbled circuits better than custom protocols?," in *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012,* The Internet Society, 2012.

[21] Pinkas, B. , Schneider, T., Segev, G. and Zohner, M., "Phasing: Private set intersection using permutation-based hashing," in *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*. (J. Jung and T. Holz, eds.), pp. 515–530, USENIX Association, 2015.

[22] Pinkas, B. , Schneider, T. , Weinert, C. and Wieder, U. , "Efficient circuit-based psi via cuckoo hashing," in *Advances in Cryptology – EUROCRYPT 2018* (J. B. Nielsen and V. Rijmen, eds.), (Cham), pp. 125–157, Springer International Publishing, 2018.

[23] Ruan, O. and Mao, H., "Efficient private set intersection using point-value polynomial representation," S*ecurity and Communication Networks*, vol. 2020, pp. 8890677:1–8890677:12, 2020.

[24] Ghosh, S. and Nilges, T., "An algebraic approach to maliciously secure private set intersection," vol. 11478, pp. 154–185, 2019.

[25] Zhao, Y. and Chow, S. S. M. ,"Are you the one to share? secret transfer with access structure," *Proc. Priv. Enhancing Technol*., vol. 2017, no. 1, pp. 149–169, 2017.

[26] Ghosh, S. And Nilges, T. ,"An algebraic approach to maliciously secure private set intersection." Cryptology ePrint Archive, Report 2017/1064, 2017.

[27] Zhang, E. , Chang, J. and Li, Y. ,"Efficient threshold private set intersection," *IEEE Access*, vol.9, pp. 6560–6570, 2021.

[28] Chandran, N., Gupta, D., and Shah, Akash, "Circuit-PSI with Linear Complexity via Relaxed Batch OPPRF", 22nd Privacy Enhancing Technologies Symposium (PETS 2022), 2022.

[29] Karakoç, F., Küpçü, A., "Enabling Two-Party Secure Computation on Set Intersection" IACR Cryptol. ePrint Arch. 2023: 609, 2023.

[30] Bay, A., Erkin, Z., Hoepman, J., Samardjiska, S. and Vos, J., "Practical multi-party private set intersection protocols," *IEEE Trans. Inf. Forensics Secur.*, vol. 17, pp. 1–15, 2022.

[31] Zhao, Y., and Chow, S. S., "Can you find the one for me?," *in Proceedings of the 2018 Workshop on Privacy in the Electronic Society, WPES'18, (New York, NY, USA)*, p. 54–65, Association for Computing Machinery, 2018.

[32] Bloom, B. H., "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, pp. 422–426, 1970.

[33] Bose, P. , Guo, H. , Kranakis, E. , Maheshwari, A. , Morin, P., Morrison, J., Smid, M. H. M. and. Tang, Y. "On the false-positive rate of bloom filters," *Inf. Process. Lett.*, vol. 108, no. 4, pp. 210–213, 2008.

[34] Davidson, A. and Cid, C., "An efficient toolkit for computing private set operations," in Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II (J. Pieprzyk and S. Suriadi, eds.), vol. 10343 of Lecture Notes in Computer Science, pp. 261–278, Springer, 2017.

[35] Shamir, A., "How to share a secret.," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[36] Reed, I. S. and Solomon, " G., Polynomial codes over certain finite fields," vol. 8, no. 2, pp. 300–304, 1960.

[37] Gao, S. , "A New Algorithm for Decoding Reed-Solomon Codes", pp. 55–68. Boston, MA: Springer, US, 2003.

[38] Yao, A. C., "Protocols for secure computations (extended abstract)," in *23rd Annual Symposium on Foundations of Computer Science*, Chicago, Illinois, USA, 3-5 November 1982, pp. 160–164, IEEE Computer Society, 1982.

[39] Veugen, T., Blom, F., Hoogh, S. J. A. de and Erkin, Z., "Secure comparison protocols in the semi-honest model," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 7, pp. 1217–1228, 2015.

[40] Garay, J. A., Schoenmakers, B. and Villegas, J. "Practical and secure solutions for integer comparison," in Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings (T. Okamoto and X. Wang, eds.), vol. 4450 of Lecture Notes in Computer Science, pp. 330–342, Springer, 2007.

[41] Bellare, M., Desai, A. , Lokipii, E. and Rogaway, P. , "A concrete security treatment of symmetric encryption," in *Proceedings of the 38th Annual Symposium on Foundations of Computer Science, FOCS '97*, (USA), p. 394, IEEE Computer Society, 1997.