Collaborative Detection of Malicious Clients for Financial Institutions using Multi-Party Computation

Trustworthy Financial Crime Analytics

Lauren de Hoop[1]
Supervisor(s): Dr. Zeki Erkin[1]**, Dr. Kubilay Atasu**[1]**, Lourens Touwen**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

Abstract

Financial institutions have a large responsibility when it comes to detecting and preventing financial crime. However, dedicated tools to aid in financial crime detection have more demand than supply. The combination of regulatory restrictions with regards to sharing client information between financial institutions and a lack of dedicated tools for financial crime detection results in a flawed system that allows criminals to evade detection and easily continue their activities by moving between institutions. This paper answers the question: *How can privacy-preserving data sharing methods enable collaborative detection of malicious clients among financial institutions?* Multi-Party Private Set Intersection (MPSI) allows multiple parties to intersect their respective datasets, without revealing any data to the other parties that are not in the intersection. A special case of MPSI is Threshold Multi-Party Private Set Intersection (T-MPSI), where given a threshold T, an item is only included if T or more parties hold that item. This paper implements a new version, Flagged Threshold Private Set Intersection (FT-MPSI), that adds a label to each item, where the label indicates if the client has been flagged as malicious - accused or convicted of financial crime. To be included in the intersection, the item must now also be identified by at least one party as malicious. The final result of the intersection is revealed to the computing party and can be shared with the parties holding the original items while no other information is leaked. The runtime performance of the FT-MPSI protocol is compared to that of the T-MPSI protocol. FT-MPSI is slower by a constant factor of approximately 2, compared to T-MPSI, it scales linearly to the number of parties and size of the sets of the input. FT-MPSI is a practical solution for financial institutions to use in financial crime detection.

1 Introduction

Whether it is the underworld laundering drug money [1] or foreign entities using charities to finance terrorism [2], it has become increasingly easier for criminals to abuse financial systems to launder their funds and scam innocent people out of their hard-earned money. The demand for better detection systems for these types of financially motivated criminals is highly sought after. Simultaneously, regulations on data protection and privacy have gained significant attention in recent years, resulting in moral, political, and regulatory conflicts [3]. The combination of a lack of dedicated tools for financial crime detection and regulatory restrictions with regards to sharing client information between financial institutions results in a flawed system that allows criminals to evade detection and easily continue their activities by moving between institutions. Even when criminal activities have been detected or clients have been convicted of financial crimes, they still manage to open new accounts with different financial institutions despite their history [4].

The lack of oversight that individual financial institutions have on the activities of their clients throughout the financial system makes it very challenging to combat financial crime in a cross-institutional landscape. Clients of a financial institution are protected under the General Data Protection Regulation (GDPR)[1], making it illegal for financial institutions to directly communicate their clients' data with other financial institutions without a substantiated legal clause. This limitation prevents financial institutions from directly communicating with each other about the activities of their clients, regardless of legality.

[1]GDPR

There exists however a protocol, The Protocol Incident Warning System for Financial Institutions (Pifi) [5], that allows financial institutions to share information with each other about suspicious activities or clients, through a centrally accessible database, see Section 3.1. A Financial Institution can search a prospective or active client for a match in the database, and retrieve more information if such a match is found. When no match is found in the database, this does not prove a client's innocence, as further client due diligence should always be conducted. The main drawback of this solution is its high cost, as it relies entirely on manual labour, making it highly inefficient. In addition, client due diligence is typically updated only once a year, which does not allow for proactive risk evaluation.

In an effort to improve the detection capabilities of the financial sector, researchers have looked at ways to detect financial crime without sharing sensitive information [6]–[11]. These protocols are specifically interested in detecting money laundering patterns between financial institutions using existing algorithms or machine learning and identifying clients who are involved in these laundering schemes. One way of ensuring data privacy utilized by some of these protocols is Secure Multi-Party Computation (MPC)[12], which enables multiple parties to compute a function without revealing their private input.

One special case of MPC is Private Set Intersection (PSI)[13], a protocol that allows multiple parties to intersect their data and only reveal the items they have in common, but not their full private datasets. PSI can be used to solve many different problems [14]–[20]. Different versions of this protocol exist, each with a distinct purpose and output [21]. As far as can be determined, there has been no attempt to utilize PSI to compare the clients of different financial institutions for overlapping suspicious account holders or current account holders who have been identified by other financial institutions as fraudulent or involved in financial crimes.

The existing methods for communicating between financial institutions lack efficiency and the ability to proactively assess potential risks. This paper aims to investigate the usability of privacy-preserving techniques for secure communication between financial institutions to assist in financial crime detection. This leads to the main research question:

How can privacy-preserving data sharing methods enable collaborative detection of malicious clients among financial institutions?

In this paper, a new method is introduced in which an adaptation of an existing PSI protocol [22]. The protocol calculates the intersection of clients from different financial institutions, where at least one financial institution has identified the client as malicious. The existing protocol is a Threshold Multi-Party Private Set Intersection (T-MPSI), where given a threshold T, an item is only included if T parties hold that item. The new version, Flagged Threshold Private Set Intersection (FT-MPSI), adds a label to each item, where the label indicates if the client has been flagged as malicious, i.e. accused or convicted of financial crime. To be included in the intersection, the item must now also be identified by at least one party as malicious. The final result of the intersection is revealed to the computing party, and can be shared with the parties holding the original items while no other information is leaked.

The contribution includes an implementation of the FT-MPSI protocol in C++ with an average run-time of 6 minutes with 50 parties all having 512 clients. This protocol is secure in the semi-honest model, meaning that no party learns anything beyond their own private input and the final set intersection.

The rest of this paper is structured as follows: Chapter 2 introduces the preliminary knowledge used throughout the paper. Chapter 3 discusses existing implementations of similar research. Chapter 4 introduces the implementation for which the results are in Chapter 5. Chapter 6 includes responsible research considerations, and finally Chapter 7 forms the discussion, limitations, and future work.

2 Preliminaries

2.1 Notation

Throughout the paper, the following notation will be used:

Table 2.1: Notations used throughout the paper

Symbol	**Description**
T	Threshold for intersection
t	Number of parties
n	Set size
$\mathcal{H}$	Hash function(s) used in Bloom filters
k	Number of hash functions
BF	Bloom filter
EBF	Encrypted Bloom filter
I_T	Result of flagged threshold private set intersection
$\mathcal{U}$	Universe of client identifiers
P_i	i-th party
P_t	Designated server party
X_i	Private input set of party P_i
F_i	Flagged subset of X_i
λ	Security parameter
ℓ	Threshold for decryption in threshold PKE
$\mathsf{Enc}(M)$	Encryption of message M using Paillier threshold PKE
$\mathsf{Dec}(C)$	Decryption of ciphertext C
$\mathsf{ReRand}(C)$	Rerandomization of ciphertext C

2.2 Secret Sharing

Secret sharing was first introduced in 1979 by Adi Shamir [23] and George Blakley [24], independently from each other, as a reliable key management system. The general idea is the following: say that you have a secret S, and you want to share this with your friends by giving each of them a part of the secret $S_1, S_2, ..., S_n$ for a total of m friends. These individual pieces will reveal nothing about the original secret S, however if enough of them are combined, say k pieces, then the secret can be recovered.

Shamir's Secret Sharing [23] is based on polynomial interpolation: given a set of coordinates,

$$S = \{(x_i, y_i) \mid x_i \in X,\ y_i \in Y\},\ \text{where } X = \{x_1, \ldots, x_m\} \text{ with } x_i \neq x_j \text{ for } i \neq j,\ \text{and } Y = \{y_1, \ldots, y_m\}.$$

There exists a unique polynomial $q(x)$ of degree $k-1$ such that $q(x_i) = y_i$ for all $x_i \in X$.

Given a secret S, generate a random $k-1$ degree polynomial

$$q(x) = a_0 + a_1x + a_2x^2 + ... + a_{k-1}x^{k-1}\ \text{, where } a_0 = S$$

and derive our partial secrets using:

$$S_1 = q(1), S_2 = q(2), ..., S_k = q(k).$$

Using polynomial interpolation, it is possible to reconstruct the polynomial if you have at least k points of the polynomial. Finally, evaluate $q(0)$ to get S. If you have less than k points, it is impossible to infer the correct polynomial from the given data points.

2.3 Bloom filter

Bloom filters were first introduced as a space-efficient probabilistic data structure by Bloom [25]. A Bloom filter uses hashing functions to effectively reduce the size of the input data as well as reduce the computation of set intersections. However, as it is still a probabilistic estimation, there is a possibility of getting a false positive, where it might say that an item is in the set, when it is not. The Bloom filter will never produce a false negative, so if an item is in the set this can always be confirmed. The probability of a false positive can be lowered by increasing the size of the Bloom filter or increasing the number of hash functions.

Using a Bloom filter to compute a set intersection works as follows: Given a set of k random hash functions $\mathcal{H} \in \{h_1(x), ..h_k(x)\}$ and two parties, P_1 and P_2 with sets S_1 and S_2, respectively. A Bloom filter starts as an array of n bits, all set to zero.

1. Both parties produce a bloom filter:
   (a) For item $s_i \in S_{\{1,2\}}$ the party calculates the output of each hash function in $\mathcal{H}$.
   (b) hash function $h_j(s_i)$ produces an index in the array. The bit at the index is set to one.
   (c) This process is repeated for each item in the set. Producing a Bloom filter.
2. The parties now exchange their respective Bloom filters.
3. Each party can now produce the set intersection as follows:
   (a) For each item $s_i \in S_{\{1,2\}}$ the party again calculates the output of each hash function in $\mathcal{H}$.
   (b) Using this input it checks whether each bit in the Bloom filter produced by the has functions is set to one.
   (c) If all bits are one, then the item is in the intersection, otherwise it is not.

Because the parties exchanged the Bloom filters, both parties learn the intersection at the same time. This protocol assumes no alterations are made to the Bloom filters in communications and all parties followed the protocol correctly. This protocol can be extended for an arbitrary amount of parties.

2.4 Homomorphic Encryption

Homomorphic encryption (HE) is a form of encryption that allows computations to be made directly on encrypted data, without having to decrypt it first. When a result is decrypted it will produce the same answer as the computation on the plaintext data would have given. This property makes it possible to do calculations on private data while only revealing the result, similar to secret sharing. Multiple Homomorphic encryption types exit, namely: partially, somewhat, levelled fully and fully homomorphic encryption.

Partially homomorphic encryption only supports a single type of operations, e.g. addition or multiplication. *Somewhat homomorphic encryption* allows for different kinds of computations but only one computation can be performed at a time, e.g. you can perform both addition and multiplication but only one operation per encryption. *Levelled fully homomorphic encryption* can handle multiple types of computations, where they can also be performed consecutively up to a certain threshold, e.g. you can perform up to 6 multiplications and 10 additions, but more would cause the result to be invalid. Lastly, *fully homomorphic encryption* allows an infinite number of computations to be performed on the encrypted data. With this increasing functionality comes the trade-off of increasing complexity and computational power. Fully homomorphic implementations allows for a lot of flexibility in the calculations, but it is more complex to maintain and will significantly slow down the calculations.

An example of an additive homomorphic encryption schema is Paillier's cryptosystem [26]. The cryptosystem allows two encrypted values, $Enc(m_1)$ and $Enc(m_2)$, to be added to each other producing $Enc(m_1 + m_2)$, as well as a multiplication of an encrypted value by some constant producing $Enc(cm_1)$ for any c.

2.5 Threshold Private Set Intersection

Multi-Party Computation (MPC) schemes enable multiple parties to compute a function without revealing their private inputs. The first instance of a multi-party computation problem is known as Yao's Millionaire's problem, based on a paper by Andrew Yao published in 1982 [27]. In this paper he proposes a problem where two millionaires wish to determine who is richer without revealing how much money they have. The solution is a two party case of multi-party computation, this solution was then extended to the multi-party setting by Oded Goldreish in 1987 [12]. Multi-Party Computations can be based on several different types of algorithms, one of them being Shamir's Secret Sharing Algorithm introduced in Section 2.2.

A Private Set Intersection (PSI) allows two parties to intersect their respective datasets, without revealing any data to the other party that is not in the intersection. PSI is a special case of MPC. Up until recently PSI has been mostly explored in the theoretical domain, however, recently some practical applications have been published [14], [17]–[22], [28]. From the introduction of MPC it took another 20 years before PSI, first introduced as Private Matching, was first formalised in 2004 by Freedman *et al.* [13]. Their paper introduced a protocol for two parties with private datasets to compute the intersection of their data without leaking any additional data, and a strawman protocol for the multi-party case.

There are several variants of PSI that can be found in literature, of which an extensive review has been done by Morales *et al.* [21]. One of these variants is the Threshold Private Set Intersection (TM-PSI), this variation has two different interpretations in literature. One is the cardinality of the intersection set, if the cardinality is above a given threshold it is revealed, otherwise it is not. The second is, given a threshold T, an item is only included in the intersection if it is in T or more parties. This last interpretation is what will be used in this paper. It should also be noted that this will usually refer to a multi-party scenario, as this interpretation has no added value in a two-party scenario. The underlying techniques used to implement TM-PSI can differ, [16] used oblivious transfer and homomorphic encryption, while [22] used Bloom filters and homomorphic key encryption schemes.

The proposed T-MPSI algorithms by [22] works as follows:
Given t parties P_i sets S_i for $i \in \{1, ..., t\}$ and k random hash functions $(h_1, .., h_k)$. There is one party that will work as the server, denoted by P_t, which will be responsible for the calculations. The rest of the parties, $\{P_1, ..., P_{t-1}\}$, are the clients.

1. Each party P_i computes the Bloom filter BF_i of their private set S_i and then encrypts the BF_i to get EBF_i, which is sent to the server.
2. For each item $y_j \in S_t$ of the server:
    (a) Create the hash for y_j, then for each client, get the bits in the same hash positions as y_j and sum all of them together.
    (b) This sum will reveal how many clients hold the item. The server can then use a secure comparison method to check if the sum is above the threshold. y_j is included in the intersection if the sum is above the threshold.
3. The server will then have the threshold intersection and can share this information if requested.

For the formal definition, the reader is referred to [22].

3 Related Work

3.1 Communication between Financial Institutions

In 2008 the Anti-Money Laundering and Anti-Terrorist Financing Act (Wet ter voorkoming van witwassen en financieren van terrorisme - Wwft)[29] was enacted in The Netherlands. In 2018 and 2020 this protocol was altered to adapt to the EU's Fourth Anti-Money Laundering Directive. The Wwft operates a risk-based approach, meaning financial and other institutions can largely choose what risks they are willing to accept relative to a risk rating they assign to each of their clients. As a result of their chosen risks, they should have proper mitigating procedures in place to then prevent Money Laundering and/or Terrorism Financing as well as notifying officials of suspicious activities. To facilitate this, Dutch financial institutions, government institutions, and some independent institutions have created the following two protocols for the purpose of communicating in accordance with the Wwft and GDPR.

Bank Data Retrieval Portal (VB) [30], [31]: Authorised institutions, including Dutch Investigative Services such as the FIOD and the Tax Authorities, can request client data through the protocol. All Dutch banks are subscribed to this system and will receive such requests. If a bank provides services to the specified client, it returns the requested data to the central portal. This protocol automates the obligatory legal proceedings for obtaining client information.

Protocol Incident Warning System for Financial Institutions (Pifi) [5], [30]: The Protocol Incident Warning System for Financial Institutions (Pifi) is a collaborative warning system accessible to participating financial institutions and insurance companies. These participants are responsible for adding clients who have violated laws related to fraud or the Wwft, and they may also report employees who have abused their position. Before onboarding a new client or employee, institutions can check the individual's details in the Pifi system. If a match is found, it includes an incident report, based on which the institution can decide whether to proceed with the onboarding process. However, the absence of a match does not guarantee the individual's integrity, as additional due diligence is always required.

For these protocols to work, one must rely on the integrity of the participating institutions and the assumption that they will share the relevant data in full and in a timely manner. Financial institutions have a legal obligation to respond to a data retrieval request from the VB according to the Wwft. The Pifi operates on the promise that participating parties provide relevant data, there is no imposed sanction if the institution does not share their information. Furthermore, as the participating institutions of Pifi are responsible for their own client due diligence, it is up to these institutions how often they check their clients against the protocol's database. Each check has to be done manually through a request to the database, making it a very labour-intensive tool for frequent use. Assuming most financial institutions adhere to a yearly update of the client due diligence, assuming the Pifi database is only checked during the client due diligence, it could take up to a year before an institution discovers their client is operating maliciously.

In general, the VB and Pifi protocols are essential in helping financial institutions to prevent and identify financial crimes and criminals, however there is room for improvement in both efficiency and reliability.

3.2 Financial Crime Detection using Secure Multi-Party Computation

Secure Multi-Party Computation has been improved over and over again since its first appearance in 1982, it has been proven to be useful to solve a host of different problems, [32] shows that there are several open-source libraries, such as FRESCO [33] and Sharemind [34], which implement different MPC protocols that can be used in practical implementations. MPC has been proven to be useful in machine learning, federated learning, securing databases, and many more. For a comprehensive overview of all its implementations, the reader is referred to [32].

For financial crime detection using MPC, the literature is less extensive. For a long time, the runtime performance and computational complexity of MPC algorithms was too large to be used in practical implementations. As mentioned above, more work has been done to improve the performance of MPC protocols, making it a feasible solution for use in practical applications [8]–[10], [35]. All of these implementations work on the basis of a transactional graph, where the nodes represent accounts and their users, and directed edges are used to represent transactions between accounts.

[35] implements a secure multi-party computation of the PageRank algorithm. This algorithm enables financial institutions to detect accounts with a lot of traffic, which, together with other detection methods, can enable banks to discriminate between fraudulent and non-fraudulent accounts. [8] takes a very different approach by implementing a graph neural network (GNN), where different parties can independently train the GNN on their data without revealing private information to the other parties. This GNN can then be used to detect fraudulent users. [9] Implements a similar approach by also introducing a GNN based on MPC for AML detection, as well as a Graph Feature Processor with a strong predictive power of 99% accuracy on an AML dataset. Lastly, [10] also uses a graph-based algorithm, where each account is given a risk score. These risk scores are then propagated several times through the graph, making it possible for the algorithm to detect unusual activity in a transactional graph.

As for PSI, existing papers have implied that the techniques could be used for financial crime detection. As far as can be determined, no research has been published on the use of PSI to aid in the detection of financial crime.

4 Malicious Client Detection across Financial Institutions

4.1 Flagged Threshold Private Set Intersection (FT-MPSI)

The implementation presented here is an adaptation of the Threshold Multi-Party Private Set Intersection (T-MPSI) protocol introduced by [22], as described in Section 2.5. This adaptation, called Flagged Threshold Multi-Party Private Set Intersection (FT-MPSI), is designed to enable financial institutions to collaboratively detect potentially malicious clients while preserving data privacy. Specifically, given several private input sets, the protocol computes the intersection of items that appear in at least T private sets, with the additional requirement that at least one of those sets has identified the item as malicious. Formally, the protocol computes:

$$I_T = \{x \in \mathcal{U} \mid |\{i \mid x \in X_i\}| \geq T \land \exists j \in [1, n] : x \in F_j\},$$

where $\mathcal{U}$ is the universe of client identifiers, X_i denotes the input set of party i, F_j is the flagged subset of X_j, T is the threshold, and n is the number of parties. The key differences in FT-MPSI compared to T-MPSI are twofold: first, each item in the input sets is associated with a flag that is set individually by each party rather than uniformly across all instances of that item; second, the server plays an active role in the protocol and is included in the threshold count.

Before presenting the steps of the protocol, there are two concepts that need to be introduced. The first is Additively Homomorphic Threshold Public Key Encryption (PKE), which allows the parties to collaboratively work on the encrypted data without revealing their inputs. All parties have the same public key, which can be used to encrypt any message. As the protocol uses Additive Homomorphic Encryption, specifically the Paillier Homomorphic Encryption [26], computations as mentioned in Section 2.4 can be performed. The private key is distributed among all parties and ℓ parties are needed to decrypt any message, this is the threshold. Secondly, Multi-Party Secure Comparison Protocol (SCP), compares two items a and b, ensuring a and b are not exposed during the computation. The SCP protocol used in the implementation is Kerschbaum *et al.* [36].

Following the steps of [22], this is the implementation of FT-MPSI:

Given: A set of t parties $\mathcal{P} = \{P_1, P_2, \ldots, P_t\}$. Each party P_i holds a private input set $X_i \subseteq \mathcal{U}$ of size n, where $\mathcal{U}$ is the universe of identifiers. Each element $x \in X_i$ is associated with a flag $f_i(x) \in \{\texttt{true}, \texttt{false}\}$ indicating whether P_i has flagged x as suspicious or malicious. Define $F_i = \{x \in X_i \mid f_i(x) = \texttt{true}\}$ as the flagged subset. A threshold parameter $T \in [1, t]$ specifies the minimum number of parties that must hold an item for it to be considered in the output.

Initialization: The last party, P_t, designated as the *server*, selects a set of k public hash functions $\{h_1, \ldots, h_k\}$ from a hash function family $\mathcal{H}$ and sends them to all other parties $P_1, \ldots, P_{t-1}$.

Encrypted Bloom Filter (EBF) Generation: Each party P_i for $i \in [1, t-1]$ performs the following:

1. Constructs a Bloom Filter BF_i for X_i using hash functions $\{h_1, \ldots, h_k\}$.
2. Constructs a Flagged Bloom Filter FBF_i for F_i using the same hash functions.
3. Encrypts each bit of both filters using the Threshold Paillier PKE scheme with shared public key pk:
$$\text{EBF}_i[j] = E_{pk}(\text{BF}_i[j]), \quad \text{FEBF}_i[j] = E_{pk}(\text{FBF}_i[j]) \tag{1}$$
4. Send EBF_i and FEBF_i to the server P_t.

Set Intersection Generation (by P_t): Let X_t be the server's own input set. For each element $y_j \in X_t$, the server does the following:

1. Compute all k hash functions $\{h_1(y_j), \ldots, h_k(y_j)\}$.
2. For each party P_i, extract ciphertexts $\{C_1^{i,j}, ..., C_k^{i,j}\}$ where
$$C_d^{i,j} = EBF_i[h_d(y_i)] \tag{2}$$
and $j \in \{1, ..., n\}$.

3. Do the same for the FEBF producing $\{FC_1^{i,j}, ..., FC_k^{i,j}\}$ where
$$FC_d^{i,j} = FEBF_i[h_d(y_i)] \tag{3}$$
and $j \in \{1, ..., n\}$

4. For each C_j^i and FC_j^i compute a fresh encryption of k, $E_{pk}(k)$.

5. Now run $t \cdot n$ SCPs in parallel, with any ℓ parties P_i to compare C_j^i to $E_{pk}(k)$. Each SCP then outputs:
$$E_{pk}(\alpha_{i,j}) = \begin{cases} E_{pk}(1) & \text{if } y_i \in X_i \\ E_{pk}(0) & \text{otherwise.} \end{cases} \tag{4}$$

6. Run SCP another $t \cdot n$ times in parallel, with any ℓ parties P_i to compare FC_j^i to $E_{pk}(pk, k)$. Each SCP again outputs:
$$E_{pk}(\phi_{i,j}) = \begin{cases} E_{pk}(1) & \text{if } y_i \in F_i \\ E_{pk}(0) & \text{otherwise.} \end{cases} \tag{5}$$

7. Compute,
$$E_{pk}(\alpha_j) = ReRand\left(\sum_{i=1}^{t-1} E_{pk}(\alpha_{i,j})\right), \tag{6}$$
and
$$E_{pk}(\phi_j) = ReRand\left(\sum_{i=1}^{t-1} E_{pk}(\phi_{i,j})\right). \tag{7}$$
α_j and ϕ_j represent the number of parties that hold y_j and the number of parties that have flagged y_j, respectively.

8. Compare $E_{pk}(\alpha_j)$ to $E_{pk}(T-1)$, using the SCP protocol with ℓ parties. This will produce:
$$E_{pk}(\beta_j) = \begin{cases} E_{pk}(1) & \text{if } \alpha_j \geq T-1 \\ E_{pk}(0) & \text{otherwise.} \end{cases} \tag{8}$$

9. Compare $E_{pk}(\phi_j)$ to $E_{pk}(1)$, again using the SCP protocol with ℓ parties and check whether $y_j \in F_t$, i.e. the server has flagged the item. Giving:
$$E_{pk}(\delta_j) = \begin{cases} E_{pk}(1) & \text{if } \phi_j \geq 1 \vee y_i \in F_t \\ E_{pk}(0) & \text{otherwise.} \end{cases} \tag{9}$$

10. For each $j \in \{1, ..., n\}$ ask ℓ parties to perform a joint decryption of all $E_{pk}(\alpha_j)$ and $E_{pk}(\phi_j)$.

11. For each $y_j \in X_t$ add to I_T if $D_{pk}(\alpha_j) = 1 \wedge D_{pk}(\phi_j) = 1$, otherwise discard.

Output: The server outputs I_T.

4.2 Protocol Correctness

The FT-MSPI implements the same protocol as the T-MSPI, of which the correctness is proven in [22]. FT-MSPI implements an additional Bloom filter with lesser elements than the original set, this Bloom filter is created using the same protocol and also encrypted using the same Homomorphic Encryption PKE protocols. The server executes the SCP for both Bloom filters using the exact same protocol. This alteration does not break the correctness of the T-MPSI protocol, as this is an addition, and the original protocol is still executed in parallel. As the added steps adhere to the original implementation of the protocol and follow the exact encryption and re-randomization steps, the FT-MPSI protocol is also as correct as T-MPSI and secure in the semi-honest model. As Bloom Filters can produce false positives, the protocol does not guarantee complete correctness. However, an increase in the number of hash functions used to create the Bloom Filter can make the false positive negligible, see Section 2.3, which creates a negligible impact on the correctness of the protocol.

4.3 Implementation

The original T-MPSI was taken from [22][2]. This implementation is in C++, depends on GMP[37] and NTL[38] libraries, and MurmurHash3[39]. The new protocol FT-MSPI is an adaptation of the T-MPSI protocol with a second set for flagged items which is run through the protocol parallel to the original set, as described in Section 4.1. Both protocols were updated to run in parallel. The practical implementation can be found on GitHub[3].

Both the original T-MPSI and the new FT-MPSI were evaluated by their run-time performance by taking the average of 10 runs of the protocol using all combinations of different number of parties (t), different set sizes (n). Each protocol was run with a threshold of two, a threshold for homomorphic public key encryption of $t/2$, and a security parameter of 1024, common in public key encryption. This benchmark implementation was also adapted from the original benchmark implementation of [22]. All benchmarks were executed on a 64-bit Linux system with an AMD Ryzen Threadripper 7970X CPU at 32 $\times$ 1.5 - 4.0 GHz and 270 GB RAM.

5 Analysis

5.1 Security Analysis

The solution [22] has been proven to be secure in the semi-honest model. In the semi-honest model, a subset of parties is considered corrupt and may communicate with each other to gain more information than what is revealed, but they still adhere the the protocol correctly. FT-MPSI adds a flag to each item in the private sets of the parties. This flag is never communicated, only the encrypted representation of all flagged items from a party is shared with the server. As it has already been proven that such a representation of items is secure in the semi-honest model, FT-MPSI is also secure in the semi-honest model.

Table 5.1: Communication and Computation Complexities of T-MPSI and FT-MPSI

	Communication		Computation	
Protocol	**Client**	**Server**	**Client**	**Server**
T-MPSI	$\mathcal{O}(\max(\lambda, t)n)$	$\mathcal{O}(nt\ell)$	$\mathcal{O}(\max(\lambda, t)n)$	$\mathcal{O}(nt)$
FT-MPSI	$\mathcal{O}(\max(\lambda, t)n)$	$\mathcal{O}(nt\ell)$	$\mathcal{O}(\max(\lambda, t)n)$	$\mathcal{O}(nt)$

5.2 Complexity Analysis

The communication and computational complexities of both protocols differ between the client and the server, as they have different roles within the protocol. For T-MPSI the complexities stated have been proven by [22]. The communication complexity of T-MPSI for the client is $\mathcal{O}(\max(\lambda, t)n)$, where λ is the statistical security parameter used in the homomorphic public key encryption, t is the threshold, and n is the input set size per party. The main source of this cost is the Secure Comparison Protocol (SCP), which is executed once in the protocol and requires the parties to communicate to the server for t rounds. The server has a communication complexity of $\mathcal{O}(nt\ell)$, where ℓ is the threshold for the homomorphic PKE. The communication complexity of the server is also due to the computation of the SCP.

FT-MPSI extends T-MPSI by executing the SCP twice, once for threshold PSI and once for flagged PSI. While this technically doubles the communication overhead, the asymptotic complexity remains the same, as constant factors are omitted. Thus, the communication complexity of FT-MPSI is also $\mathcal{O}(\max(\lambda, t)n)$ for the clients and $\mathcal{O}(nt\ell)$ for the server.

The computational complexity of T-MPSI for the server is $\mathcal{O}(nt)$ is due to the homomorphic encryption needed to perform all the rounds of the SCP from the client side. The computational complexity of the clients is dominated by either the creation of their Bloom Filters, which has a complexity of $\mathcal{O}(\lambda n)$, of the SCP protocol as well, which has a complexity of $\mathcal{O}(nt)$. This leads to the clients having a computational complexity of $\mathcal{O}(\max(\lambda, t)n)$.

In FT-MPSI, each party constructs two encrypted Bloom filters, one for the threshold PSI and another for the flagged PSI-resulting in a computational complexity of $\mathcal{O}(\lambda n)$. The server, performs the SCP twice for each party, yielding a computational complexity of $\mathcal{O}(\max(\lambda, t)n)$. As a result, the overall computational complexity remains $\mathcal{O}(\lambda n)$ for the parties, and $\mathcal{O}(\max(\lambda, t)n)$ for the server.

5.3 Performance Analysis

Table 5.3 reports the mean run-time and standard deviation, in seconds, of the FT-MPSI protocol for varying numbers of parties and set sizes. Results are shown for party counts ranging from 5 to 50 and set sizes from 4 to 512. Each entry represents the average over ten runs, with standard deviations included. Table 5.2 presents the corresponding run-time measurements for the T-MPSI protocol under the same conditions. Both protocols were run with a security parameter $\lambda = 1024$ and $\ell = t/2$. Furthermore, Figure 5.3 creates a linear

[2]Source code available at: `https://github.com/jellevos/threshold-multiparty-psi`
[3]Source code can be found on: `https://github.com/Lorn1020/threshold-multiparty-psi`

representation of all runtimes of both the T-MSPI and FT-MPSI protocols. Each colour corresponds to a set size, where the dotted lines represent the FT-MPSI run-times and the solid line the T-MPSI run-times.

Table 5.2: T-MPSI: Run-time performance mean and standard deviation in seconds by number of parties and set size averaged over 10 runs

	$\mathbf{n} = 2^2$	$\mathbf{n} = 2^4$	$\mathbf{n} = 2^6$	$\mathbf{n} = 2^8$	$\mathbf{n} = 2^9$
t = 5	0.26 ± 0.02	0.63 ± 0.03	2.35 ± 0.02	9.39 ± 0.05	18.86 ± 0.09
t = 15	0.51 ± 0.04	1.55 ± 0.03	5.68 ± 0.04	22.82 ± 0.26	45.40 ± 0.42
t = 25	0.72 ± 0.03	2.37 ± 0.03	8.93 ± 0.05	36.01 ± 0.36	$71.89b \pm 0.61$
t = 35	1.14 ± 0.01	4.11 ± 0.04	15.63 ± 0.07	62.30 ± 0.30	124.18 ± 0.80
t = 40	1.31 ± 0.01	4.70 ± 0.03	18.05 ± 0.15	72.82 ± 0.38	144.71 ± 0.59
t = 50	1.77 ± 0.03	6.34 ± 0.03	24.54 ± 0.08	97.84 ± 0.38	195.63 ± 1.98

Table 5.3: FT-MPSI: Run-time performance mean and standard deviation in seconds by number of parties and set size averaged over 10 runs

	$\mathbf{n} = 2^2$	$\mathbf{n} = 2^4$	$\mathbf{n} = 2^6$	$\mathbf{n} = 2^8$	$\mathbf{n} = 2^9$
t = 5	0.37 ± 0.02	0.87 ± 0.03	3.22 ± 0.03	12.76 ± 0.05	25.51 ± 0.12
t = 15	0.68 ± 0.05	2.13 ± 0.12	7.94 ± 0.08	32.03 ± 0.32	63.05 ± 0.54
t = 25	1.22 ± 0.03	4.03 ± 0.07	15.44 ± 0.18	62.38 ± 0.78	122.90 ± 0.67
t = 35	1.99 ± 0.02	6.92 ± 0.09	25.79 ± 0.16	101.10 ± 0.64	201.81 ± 1.18
t = 40	2.49 ± 0.04	8.68 ± 0.10	32.80 ± 0.38	128.15 ± 0.54	255.76 ± 1.18
t = 50	3.34 ± 0.04	12.03 ± 0.11	46.24 ± 0.29	184.37 ± 1.49	365.67 ± 2.05

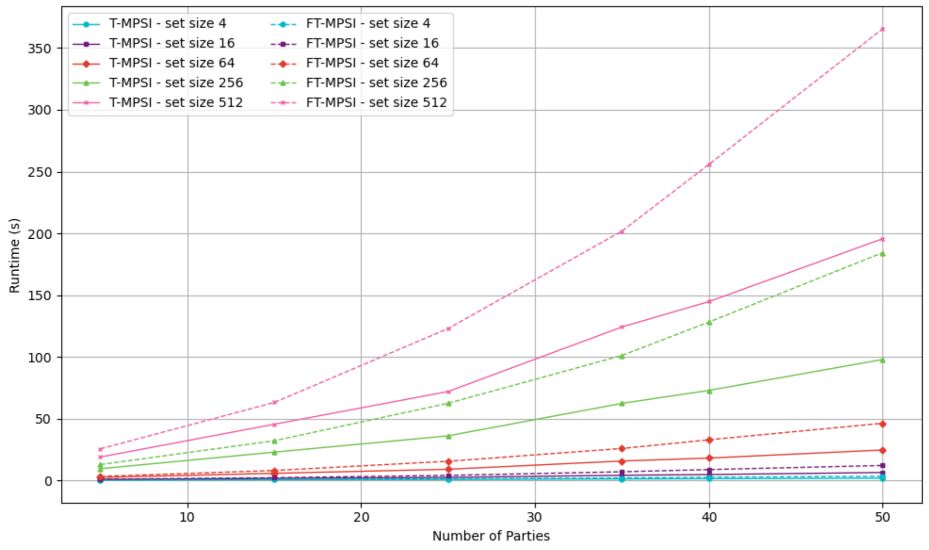

Figure 5.1: Run-time comparison of the T-MPSI and FT-MPSI protocols averaged over 10 runs

6 Responsible Research

The datasets used for the experiments are all synthetic datasets specifically created for this research. Even though the synthetic datasets do not contain any sensitive information, the ultimate use case of the algorithm would be based on the collection of personally identifiable information. However, the whole purpose of the algorithm used is to obscure this personal data in such a way that it is not revealed to any party not already in possession of it. If potential future users follow the protocol in the correct manner, this will safeguard their private information, this does also include the assumptions made prior to running the algorithm. The code for the final implementation of the PSI protocol has also been made available on GitHub[4].

7 Conclusion

The results indicate that the run-time of FT-MPSI increases with both the number of participating parties and the set size. This matches the expected $\mathcal{O}(\max(\lambda, t)n)$ complexity, where n represents the set size. For a fixed number of parties, the run-time grows approximately linearly with the set size, while increasing the number of parties leads to higher run-time due to additional protocol interactions. In all configurations, the observed standard deviations are small compared to the mean values, suggesting consistent performance across repeated runs. Comparing FT-MPSI with T-MPSI, the latter consistently achieves lower run-times, taking roughly half the time. This is consistent with the complexity prediction, as FT-MPSI produces two Bloom Filters instead of one, and performs the set intersection for both filters accumulating their results in the final intersection. Despite this additional

[4]Source code can be found on: `https://github.com/Lorn1020/threshold-multiparty-psi`

overhead, FT-MPSI shows similar scaling and proves to be practical in its implementation.

Several assumptions have been made during the course of this thesis which influenced the result of the experiments. The implemented code is all run on the same machine and, while it emulates the parties to be independent from each other, they are not run separately, so no communication delay is present in the runtime results. Furthermore, this implementation assumes that all parties are honest about their input and do not flag innocent clients to gather information from other parties that it is not authorised to access.

The implementation of T-MPSI uses Kerschbaum *et al.* Secure Comparison Protocol [36], more efficient SCP protocols have been implemented since Kerschbaum, such as Lu *et al.* [40] who created a constant round SCP, which could significantly speed up the computation of the T-MPSI and FT-MPSI protocol. Furthermore, in the FT-MPSI protocol, the flagged comparisons protocol could be replaced with a Private Set Union, such as in [41], to possibly speed up computation. Future work should also focus on implementing the protocol for a distributed setting, where parties can operate on separate servers. This also allows communication between parties to be included in the run-time evaluation of the protocol. The protocol should also be tested against Wwft [42] and GDPR legislation to evaluate its usability by financial institutions as a detection tool.

FT-MSPI was created to address the limitations of existing privacy-preserving PSI protocols. While T-MPSI [22] allows Financial Institutions to identify common clients based on a threshold of parties, it lacks the ability to distinguish between innocent and malicious clients. Existing protocols such as Pifi [5] rely heavily on manual work and does not allow for proactive risk evaluation. FT-MSPI addresses these concerns by introducing a flagging mechanism that allows financial institutions to flag their clients as suspicious or malicious and ensures only clients that are held my multiple institutions and have been identified as malicious by at least one of them are in the final output. This allows the financial institutions to collaboratively identify malicious clients while adhering to the strict data protection regulations. FT-MPSI offers a practical, automated, and privacy-preserving method as an alternative to the current state of the art in collaborative financial crime detection.

References

[1] "Global investigation exposes alleged billion-dollar Russian money-laundering network," *The Guardian*, Dec. 2024, ISSN: 0261-3077. [Online]. Available: https://www.theguardian.com/business/2024/dec/04/global-investigation-exposes-alleged-billion-dollar-russian-money-laundering-network (visited on 05/30/2025).

[2] "Funding for Terror-Linked NGOs through US-Registered Charities and Financial Service Providers," *NGO Monitor*, Mar. 2025. [Online]. Available: https://ngo-monitor.org/reports/funding-for-terror-linked-ngos-through-us-registered-charities-and-financial-service-providers (visited on 05/30/2025).

[3] R. Pascoe, "Banks should work together on money laundering, minister says," *DutchNews.nl*, Jan. 2025. [Online]. Available: https://www.dutchnews.nl/2025/01/banks-should-work-together-on-money-laundering-minister-says/ (visited on 05/30/2025).

[4] J. Kollewe, "Starling Bank fined £29m for 'shockingly lax' financial crime controls," *The Guardian*, Oct. 2024, ISSN: 0261-3077. [Online]. Available: https://www.theguardian.com/business/2024/oct/02/starling-bank-fined-29m-for-shockingly-lax-financial-controls (visited on 05/30/2025).

[5] *Protocol Incidenten-waarschuwingssysteem Financiële Instellingen (Pifi)*. [Online]. Available: https://www.nvb.nl/publicaties/protocollen-regelingen-richtlijnen/protocol-incidenten-waarschuwingssysteem-financiele-instellingen-pifi/ (visited on 05/30/2025).

[6] I. Molloy, S. Chari, U. Finkler, *et al.*, "Graph Analytics for Real-Time Scoring of Cross-Channel Transactional Fraud," in *Financial Cryptography and Data Security*, J. Grossklags and B. Preneel, Eds., vol. 9603, Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 22–40, ISBN: 978-3-662-54969-8 978-3-662-54970-4. (visited on 04/30/2025).

[7] A. Sangers, M. Van Heesch, T. Attema, *et al.*, "Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection," in *Financial Cryptography and Data Security*, I. Goldberg and T. Moore, Eds., vol. 11598, Cham: Springer International Publishing, 2019, pp. 605–623. DOI: 10.1007/978-3-030-32101-7_35. (visited on 04/28/2025).

[8] X. Liu, X. Fan, R. Ma, *et al.*, "Collaborative Fraud Detection on Large Scale Graph Using Secure Multi-Party Computation," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, Boise ID USA: ACM, Oct. 2024, pp. 1473–1482, ISBN: 979-8-4007-0436-9. DOI: 10.1145/3627673.3679863. [Online]. Available: https://dl.acm.org/doi/10.1145/3627673.3679863 (visited on 04/29/2025).

[9] F. Effendi and A. Chattopadhyay, "Privacy-Preserving Graph-Based Machine Learning with Fully Homomorphic Encryption for Collaborative Anti-money Laundering," in *Security, Privacy, and Applied Cryptography Engineering*, vol. 15351, Cham: Springer, 2024, pp. 80–105. DOI: 10.1007/978-3-031-80408-3_6. (visited on 05/16/2025).

[10] M. B. van Egmond, V. Dunning, S. van den Berg, *et al.*, "Privacy-Preserving Anti-money Laundering Using Secure Multi-party Computation," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 14745, Cham: Springer, 2025, pp. 331–349. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-78679-2_18 (visited on 05/16/2025).

[11] "Enhancing Anti-money Laundering Efforts with Network-based Algorithms," in *Complex Networks & Their Applications XIII*, Cham: Springer, 2025, pp. 115–124. [Online]. Available: `https://link.springer.com/10.1007/978-3-031-82431-9_10` (visited on 05/01/2025).

[12] O. Goldreich, S. Micali, and A. Wigderson, "How to play ANY mental game," in *Proceedings of the nineteenth annual ACM conference on Theory of computing - STOC '87*, New York, New York, United States: ACM Press, 1987, pp. 218–229. DOI: `10.1145/28395.28420`. (visited on 04/30/2025).

[13] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," in *Advances in Cryptology - EUROCRYPT 2004*, T. Kanade, J. Kittler, J. M. Kleinberg, *et al.*, Eds., vol. 3027, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 1–19. [Online]. Available: `http://link.springer.com/10.1007/978-3-540-24676-3_1` (visited on 04/30/2025).

[14] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov, "BotGrep: Finding P2P bots with structured graph analysis," in *Proceedings of the 19th USENIX conference on Security*, ser. USENIX Security'10, Washington, DC, USA: USENIX Association, Aug. 2010, p. 7. [Online]. Available: `http://www.usenix.org/events/sec10/tech/full%5C_papers/Nagaraja.pdf` (visited on 04/30/2025).

[15] 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*, ser. CCS '17, New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 1257–1272. DOI: `10.1145/3133956.3134065`. (visited on 05/02/2025).

[16] R. A. Mahdavi, T. Humphries, B. Kacsmar, *et al.*, "Practical Over-Threshold Multi-Party Private Set Intersection," in *Proceedings of the 36th Annual Computer Security Applications Conference*, ser. ACSAC '20, New York, NY, USA: Association for Computing Machinery, Dec. 2020, pp. 772–783. DOI: `10.1145/3427228.3427267`. (visited on 05/23/2025).

[17] A. Bay, Z. Erkin, M. Alishahi, and J. Vos, "Multi-party private set intersection protocols for practical applications," *Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021*, Proceedings of the 18th International Conference on Security and Cryptography, SECRYPT 2021, S. D. C. di Vimercati and P. Samarati, Eds., pp. 515–522, 2021. DOI: `10.5220/0010547605150522`. (visited on 05/15/2025).

[18] Y. Li, D. Ghosh, P. Gupta, S. Mehrotra, N. Panwar, and S. Sharma, "PRISM: Private Verifiable Set Computation over Multi-Owner Outsourced Databases," in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD '21, New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 1116–1128. DOI: `10.1145/3448016.3452839`. (visited on 04/30/2025).

[19] Y.-C. Chen and K.-C. Huang, "JEDI: Joint and Effective Privacy Preserving Outsourced Set Intersection and Data Integration Protocols," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4504–4514, 2023. DOI: `10.1109/TIFS.2023.3295941`. (visited on 05/23/2025).

[20] C. Guan, J. van Assen, and Z. Erkin, "Collective Threshold Multiparty Private Set Intersection Protocols for Cyber Threat Intelligence," in *2024 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec. 2024, pp. 1–6. DOI: 10.1109/WIFS61860.2024.10810671. [Online]. Available: https://ieeexplore.ieee.org/document/10810671 (visited on 05/15/2025).

[21] D. Morales, I. Agudo, and J. Lopez, "Private set intersection: A systematic literature review," *Computer Science Review*, vol. 49, p. 100 567, 2023. DOI: 10.1016/j.cosrev.2023.100567. (visited on 04/28/2025).

[22] A. Bay, Z. Erkin, J.-H. Hoepman, S. Samardjiska, and J. Vos, "Practical Multi-Party Private Set Intersection Protocols," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1–15, 2022, ISSN: 1556-6021. DOI: 10.1109/TIFS.2021.3118879. (visited on 05/15/2025).

[23] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. DOI: 10.1145/359168.359176. (visited on 05/02/2025).

[24] G. R. BLAKLEY, "Safeguarding cryptographic keys," in *1979 International Workshop on Managing Requirements Knowledge (MARK)*, Jun. 1979, pp. 313–318. DOI: 10.1109/MARK.1979.8817296. (visited on 05/02/2025).

[25] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. DOI: 10.1145/362686.362692. (visited on 04/30/2025).

[26] 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., vol. 1592, Springer, 1999, pp. 223–238. [Online]. Available: http://link.springer.com/10.1007/3-540-48910-X_16 (visited on 06/02/2025).

[27] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, Chicago, IL, USA: IEEE, Nov. 1982, pp. 160–164. DOI: 10.1109/SFCS.1982.38. (visited on 04/30/2025).

[28] X. Qian, L. Wei, J. Zhang, and L. Zhang, "Malicious-Secure Threshold Multi-Party Private Set Intersection for Anonymous Electronic Voting," *Cryptography*, vol. 9, no. 2, p. 23, 2025. DOI: 10.3390/cryptography9020023. (visited on 05/23/2025).

[29] M. v. B. Z. e. Koninkrijksrelaties, *Wet ter voorkoming van witwassen en financieren van terrorisme*. [Online]. Available: https://wetten.overheid.nl/BWBR0024282/2025-03-01 (visited on 05/15/2025).

[30] *Data delen tegen criminaliteit*. [Online]. Available: https://www.nvb.nl/themas/digitaal-van-dienst/uw-persoonsgegevens/data-delen-tegen-criminaliteit/ (visited on 05/08/2025).

[31] M. v. J. e. Veiligheid, *Verwijzingsportaal Bankgegevens (VB) - Producten- en dienstencatalogus - Justitiële Informatiedienst*, webpagina, Apr. 2021. [Online]. Available: https://www.justid.nl/producten-en-dienstencatalogus/digitaal-uitwisselen/routeren-informatie/verwijzingsportaal-bankgegevens-vb (visited on 05/08/2025).

[32] D. Feng and K. Yang, "Concretely efficient secure multi-party computation protocols: Survey and more," *Security and Safety*, vol. 1, p. 2 021 001, 2022. DOI: 10.1051/sands/2021001. (visited on 06/02/2025).

[33] Alexandra Institute, *FRESCO - A FRamework for Efficient Secure Computation*, Feb. 2025. [Online]. Available: https://github.com/aicis/fresco.

[34] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A Framework for Fast Privacy-Preserving Computations," in *Computer Security - ESORICS 2008*, S. Jajodia and J. Lopez, Eds., vol. 5283, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 192–206. [Online]. Available: http://link.springer.com/10.1007/978-3-540-88313-5_13 (visited on 06/02/2025).

[35] A. Sangers, M. Van Heesch, T. Attema, *et al.*, "Secure Multiparty PageRank Algorithm for Collaborative Fraud Detection," en, in *Financial Cryptography and Data Security*, I. Goldberg and T. Moore, Eds., vol. 11598, Cham: Springer International Publishing, 2019, pp. 605–623, ISBN: 978-3-030-32100-0 978-3-030-32101-7. DOI: 10.1007/978-3-030-32101-7_35. [Online]. Available: http://link.springer.com/10.1007/978-3-030-32101-7_35 (visited on 04/28/2025).

[36] F. Kerschbaum, D. Biswas, and S. de Hoogh, "Performance Comparison of Secure Comparison Protocols," in *2009 20th International Workshop on Database and Expert Systems Application*, Aug. 2009, pp. 133–136. DOI: 10.1109/DEXA.2009.37. (visited on 06/10/2025).

[37] F. S Foundation, *The GNU MP Bignum Library*. [Online]. Available: https://gmplib.org/ (visited on 06/09/2025).

[38] V. Shoup, *NTL: A Library for doing Number Theory*. [Online]. Available: https://libntl.org/ (visited on 06/09/2025).

[39] *MurmurHash3*. [Online]. Available: https://github.com/aappleby/smhasher/wiki/MurmurHash3 (visited on 06/09/2025).

[40] T. Lu, X. Kang, B. Zhang, *et al.*, "Efficient 2PC for Constant Round Secure Equality Testing and Comparison," 2024. [Online]. Available: https://eprint.iacr.org/2024/949 (visited on 06/10/2025).

[41] J. Vos, M. Conti, and Z. Erkin, "Fast Multi-party Private Set Operations in the Star Topology from Secure ANDs and ORs," *IACR Cryptol. ePrint Arch.*, p. 721, 2022. [Online]. Available: https://eprint.iacr.org/2022/721 (visited on 06/19/2025).

[42] *Introduction Wwft*. [Online]. Available: https://www.dnb.nl/en/sector-information/open-book-supervision/laws-and-eu-regulations/anti-money-laundering-and-anti-terrorist-financing-act/introduction-wwft/ (visited on 05/15/2025).