On the Insecurity of Bloom Filter-Based Private Set Intersections

Jelle Vos*
J.V.Vos@tudelft.nl

Jorrit van Assen*
J.S.VanAssen@tudelft.nl

Tjitske Koster
T.O.Koster@tudelft.nl

Evangelia Anna Markatou
E.A.Markatou@tudelft.nl

Zekeriya Erkin
Z.Erkin@tudelft.nl

November 22, 2024

Abstract

Private set intersections are cryptographic protocols that compute the intersection of multiple parties' private sets without revealing elements that are not in the intersection. These protocols become less efficient when the number of parties grows, or the size of the sets increases. For this reason, many protocols are based on Bloom filters, which speed up the protocol by approximating the intersections, introducing false positives with a small but non-negligible probability. These false positives are caused by hash collisions in the hash functions that parties use to encode their sets as Bloom filters. In this work, we show that these false positives are more than an inaccuracy: an adversary in the augmented semi-honest model can use them to learn information about elements that are not in the intersection. First, we show that existing security proofs for Bloom filter-based private set intersections are flawed. Second, we show that even in the most optimistic setting, Bloom filter-based private set intersections cannot securely realize an approximate private set intersection unless the parameters are so large that false positives only occur with negligible probability. Third, we propose a practical attack that allows a party to learn if an element is contained in a victim's private set, showing that the problem with Bloom filters is not just theoretical. We conclude that the efficiency gain of using Bloom filters as an approximation in existing protocols vanishes when accounting for this security problem. We propose three mitigations besides choosing larger parameters: One can use oblivious pseudo-random functions instead of hash functions to reduce the success rate of our attack significantly, or replace them with password-based key derivation functions to significantly slow down attackers. A third option is to let a third party authorize the input sets before proceeding with the protocol.

*These authors contributed equally to this work.

1 Introduction

Private set intersection protocols (PSI) and their multi-party equivalent are protocols for computing the intersection between n parties' private sets, without revealing any other information about those private sets. These protocols enable information sharing in situations where revealing data would be undesirable, like in financial transactions, or where information sharing must be limited, like in threat intelligence or no-fly lists. More formally, a private set intersection protocol is a protocol between n parties $\mathcal{P}_i$ for $i = 1, \ldots, n$. Each party has a private set $X_i \subseteq \mathcal{U}$ of at most k elements. One party that we refer to as the leader (denoted $\mathcal{P}_1$) obtains the intersection $X_1 \cap \cdots \cap X_n$ as the protocol's output. All other attributes of the private sets must remain hidden.

Approximate PSI schemes allow a trade-off between the computational and communicational cost of a protocol and the accuracy of the resulting intersection. A common method for constructing efficient approximate PSI protocols is to use Bloom filters. Bloom filter-based PSI protocols let parties first encode their sets as Bloom filters $\hat{X}_i \leftarrow \mathsf{Encode}(X_i)$ using h hash functions H_j for $j = 1, \ldots, h$. These filters start out as an indexed set of m Boolean bins that are all set to 0. Each party $\mathcal{P}_i$ uses the filter's hash functions to map each of their elements in set X_i to h of the bins, setting them to 1. One can compute a Bloom filter representing the intersection by combining the Bloom filters using an element-wise logical `AND` operation. The `AND` operation must be performed on the Bloom filters, which must remain private, using a secure computation technique such as homomorphic encryption.

The approximation inherent to Bloom filters is caused by the possibility of hash collisions: a hash of any two distinct elements may map to the same bin (i.e. $H_i(x) = H_j(x')$ where $x \neq x'$). As such, such a Bloom filter-based protocol will never wrongfully exclude elements from the intersection (i.e. there are no false negatives), but the result may include false positives with some probability. Specifically, each negative element in the leader's set may wrongfully appear in the intersection with probability at most p. This makes Bloom filter-based MPSI protocols suitable for use cases, in which false positives may be permissible with a small but non-negligible probability.

Previous work [1, 2] has shown that the Bloom filter representing the intersection might leak information if it is revealed. This is because bins in the intersection may be set to 1, even if the same bins are set to 0 when the Bloom filter is obtained by directly encoding the intersection, $\mathsf{Encode}(X_1 \cap \cdots \cap X_n)$. Instead of revealing the combined Bloom filter $\hat{X}_\cap = \hat{X}_1 \wedge \cdots \wedge \hat{X}_n$, private set intersection protocols use secure computation techniques to query the filter on every element in the leader's set and only reveal the result:

$$\hat{X}_\cap[H_1(x)] \wedge \cdots \wedge \hat{X}_\cap[H_h(x)] \text{ for } x \in X_1 \,. \tag{1}$$

One might think that this constitutes a secure Bloom filter-based private set intersection protocol, as the leader would not be able to distinguish between false positives and actual elements in the intersection, preventing it from exploiting $\hat{X}_\cap$ to learn anything about the private sets. However, this assumes that the leader has no auxiliary knowledge about the private sets. Rindal & Rosulek [3] already showed that Bloom filters lead to problems in security proofs in the malicious setting, and other recent work by Liu et al. [4] identifies problems with Bloom filter-based private set unions.

In this work, we show that the above approach is not sufficient to achieve secure MPSI: the approximate nature of Bloom filters does, in fact, allow the leader to learn information about the private sets that it could not from the exact intersection. What is more, previous works do not take this approximation into account in their security proofs. For example,

several works prove security with respect to the ideal functionality of an exact PSI to model the security of Bloom filter-based protocols, as opposed to the ideal functionality of an approximate PSI. This gap caused by approximation can only be closed if p is negligible, but this, in turn, causes parameters to grow significantly. In this work, we show that these parameters are so large that they undo the performance benefit of choosing an approximate protocol in the first place.

One might think that the gap can be easily closed by proving security with respect to an ideal functionality for approximate set intersections. Unfortunately, we show that Bloom filter-based PSI protocols are fundamentally flawed in this regard. Wo do so by exploiting the fact that the actual false positive rate is not constant; it depends on the elements in the private sets. The result is that the existence of a false positive in the final intersection may reveal information about any of the private sets. Consider the following minimal example (albeit slightly contrived), which demonstrates that the false positive rate of a Bloom filter does not only affect the correctness but also the security of the protocol. In other words, even a perfectly secure Bloom filter-based PSI leaks information about the input sets with non-negligible probability.

Example 1. *Let us analyze the situation where $h = 1$, $n = 2$, and $k = 1$. Assume that we have two distinct party-specific universes, $\mathcal{U}_1 = \{\mathtt{a}\}$ and $\mathcal{U}_2 = \{\mathtt{b}\}$, and we use a Bloom filter-based PSI protocol in which the two parties input sets X_1 and X_2. Then, in the ideal world corresponding to ε_{fp}-approximate PSI, the leader would get a non-empty intersection with probability ε_{fp}, regardless of X_2. However, in the real world, the output is non-empty with probability $\frac{1}{m}$ if, and only if, $X_1 = \{\mathtt{a}\}$ and $X_2 = \{\mathtt{b}\}$, but is always empty otherwise. I.e. the leader learns the other party's set with probability $\frac{1}{m}$.*

The attacks described in this work are all in the augmented semi-honest model; where parties can freely choose their inputs, after which they do not deviate from the protocol. This is an augmentation of the semi-honest model, in which parties that are not corrupted do not deviate from their predetermined inputs to the protocol. It has been shown that this property of the semi-honest model leads to counter-intuitive situations in which a protocol that is secure in the malicious model cannot be proven to be secure in the semi-honest model [5].

As a result of the attacks we propose, Bloom filter-based private set intersection protocols that use a non-negligible false positive probability will become slower. The easiest mitigation is to lower the false positive probability, slowing down the protocol due to the use of larger Bloom filters. Alternative solutions would include using oblivious pseudo-random functions [6] such that the hash functions can remain secret, or switching to hash functions that are very expensive to compute, such as password-based key derivation functions like PBKDF2 [7].

The paper is organized as follows. We proceed with a description of Bloom filters in Section 2. After that, in Section 3, we define the security model for multi-party private set intersections that we use, including definitions for approximate MPSI. Next, we present an abstraction of Bloom filter-based MPSI protocols in Section 4. We present our main results in Sections 5 and 6, in which we put forward our theoretical analysis and our practical attack. We finish by discussing mitigations in Section 7, and we conclude in Section 8.

2 Bloom Filters

Bloom filters, first introduced by Bloom [8], are a probabilistic data structure designed to perform set membership queries efficiently. Bloom filters exploit the uniformity of h hash functions to each map an element $x \in \mathcal{U}$ to one of m bins. Each bin contains one bit that all start at 0. To encode an element x, we hash x with h hash functions H_i `for` $i \in \{1, 2, \ldots, h\}$, which select bins of the Bloom filters to be set to 1. We can encode a private set X by encoding each element $x \in X$ individually. We define the notation $\hat{X} \leftarrow \mathsf{encode}(X)$ to denote the Bloom filter encoding of set $X \subseteq \mathcal{U}$.

One can test whether an element y is contained in the Bloom filter $\hat{X}$ by computing all bins corresponding to y and checking if indeed all bits are 1.False negatives cannot occur, but when all of the bins of element y are set to 1, it is not sure that the element was indeed encoded in the Bloom filter, or that the bins were set to 1 by encoding other elements. I.e. false positives *can* occur. We use p to denote an upper bound on the probability of a Bloom filter encoding k distinct elements returning a false positive. The probability p depends on the number k of elements inserted, the size of the Bloom filter m and the number h of hash functions used. An upper bound was derived by Goel and Gupta [9].

$$p \leq \left(1 - e^{-\frac{h(N+0.5)}{m-1}}\right)^h \ . \tag{2}$$

Given a desired false positive probability ε_{fp} and maximum set size k, we can calculate the corresponding required number of hash functions h and the minimal number of bins m_{opt} as follows:

$$h = -\log_2(\varepsilon_{\text{fp}}) \ , \tag{3}$$

$$m_{opt} \geq \frac{-h\,(k+0.5)}{\ln\left(1 - \sqrt[h]{p}\right)} + 1 \ . \tag{4}$$

The protocols discussed in this work use Bloom filters to compute intersections. One way in which Bloom filters are convenient for this purpose, is that different Bloom filters can be combined to generate a filter representing the intersection. Specifically, two Bloom filters $\hat{X}_1$ and $\hat{X}_2$ can be combined using a bin-wise `AND` operation to generate a Bloom filter $\hat{X}_\cap$ representing the intersection. Bloom filters do not allow for efficient extraction of the original elements, however, they do allow for efficient testing of the inclusion of a specified value $x \in \mathcal{U}$, so one can extract the intersection by querying the elements from one of the original sets.

3 Definition of MPSI security

The results in our work contradict the security proofs in previous work [10, 11]: we show that Bloom filter-based private set intersections with non-negligible false positive probabilities cannot securely realize private set intersections, whereas previous work contains security proofs for the opposite. In this section, we first discuss how the security definitions and analyses of previous work and show how these are flawed. One of these flaws is that the security proofs attempt to show that Bloom filter-based PSI realizes *exact* PSI, but this is clearly not true when false positives occur. In the second part of this section, we propose new definitions for *approximate* PSI. In Section 5, we show that Bloom filter-based PSI also does not realize these weaker functionalities.

3.1 Definitions & flaws in existing security proofs

We now inspect the security proofs of Bloom filter-based PSI protocols more closely. We specifically focus on works that use the approximation of a Bloom filter to speed up the protocol. In other words; works that set the false positive probability p to be non-negligible.

In their proof (Theorem 4.2), Debnath et al. [10] assume that the output is the exact intersection, claiming that the protocol only fails with the false positive probability p. However, in reality, the security proof does not hold the moment that any false positive occurs. This can happen with probability $1 - (1 - p)^k$. Bay et al. [11] also assume that the output is the exact intersection, so the proof does not hold with probability this probability, but their proof is slightly different. They simulate the inputs of uncorrupted parties by choosing random input sets that conform to the intersection. Given that the combined Bloom filter highly depends on the input sets, this potentially skews the advantage even more.

Other works, like that by Vos et al. [12] only prove that the aggregation is secure, so the security proof does not extend to the final computation of the intersection. The same goes for the work by Miyaji et al. [13], which only considers security for the protocol before decryption.

To summarize, all these proofs either use the MPSI functionality, thereby failing to consider false positives, or the proofs are incomplete (because they do not consider Bloom filters). There is an option for remedying these proofs, namely including the approximate behaviour of the underlying protocol and showing that false positives occur with a negligible probability, e.g. $p \leq 2^{-40}$. However, for the addressed schemes, this results in a significant decrease in performance. There are already other works that take this approach, such as the work by Ben Efraim et al. [14]. If false positives practically never occur, then the protocol behaves as an exact intersection.

3.2 An exact ideal functionality

To treat two-party private set intersections and multi-party private set intersections in general, we define an exact ideal functionality $\mathcal{F}_{\text{MPSI}}$ for MPSI that roughly follows the universal composability model, see Figure 1. In this functionality, $\mathcal{S}$ is essentially an external adversary that controls the communication channels. In this work, it is sufficient to think of the $\mathcal{S}$ as an external influence that decides when the protocol finishes.

3.3 An approximate ideal functionality

As mentioned before, the exact ideal functionality is unsuitable for proving the security of Bloom filter-based MPSI when the false positive probability is not negligible. After all, any false positive would allow a distinguisher to tell it apart from the exact MPSI ideal functionality. Instead, we define an approximate MPSI ideal functionality $\mathcal{F}_{\text{aMPSI}}$ that returns an intersection based on the leader's set with a constant probability of false positives ε_{fp} and false negatives ε_{fn}, see Figure 2. In the rest of our paper, $\varepsilon_{\text{fn}} = 0$. We refer to an approximate MPSI protocol with false positive probability ε_{fp} as ε_{fp}-approximate.

3.4 A weaker ideal functionality

In Section 5, we show that Bloom filters with a non-negligible false positive probability also cannot securely realize $\mathcal{F}_{\text{aMPSI}}$. One might argue that the only reason why Bloom filters are not approximate MPSIs is that their false positive probability varies, but that this variance

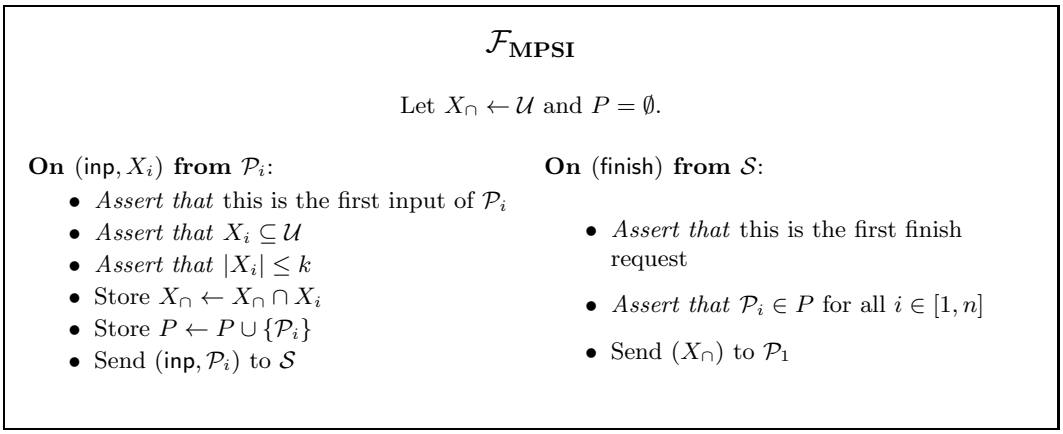

Figure 1: The ideal functionality $\mathcal{F}_{\text{MPSI}}$.

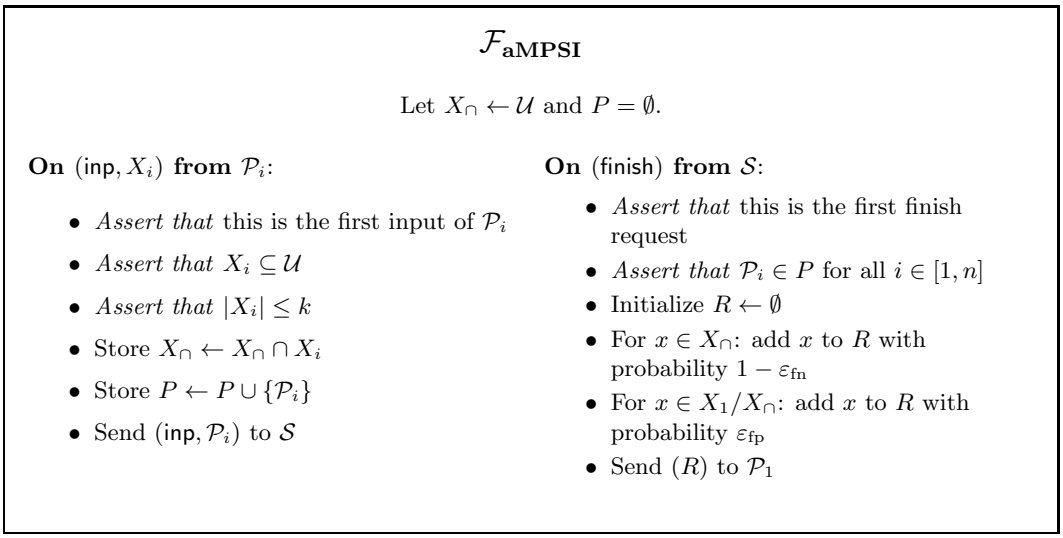

Figure 2: The ideal functionality $\mathcal{F}_{\text{aMPSI}}$.

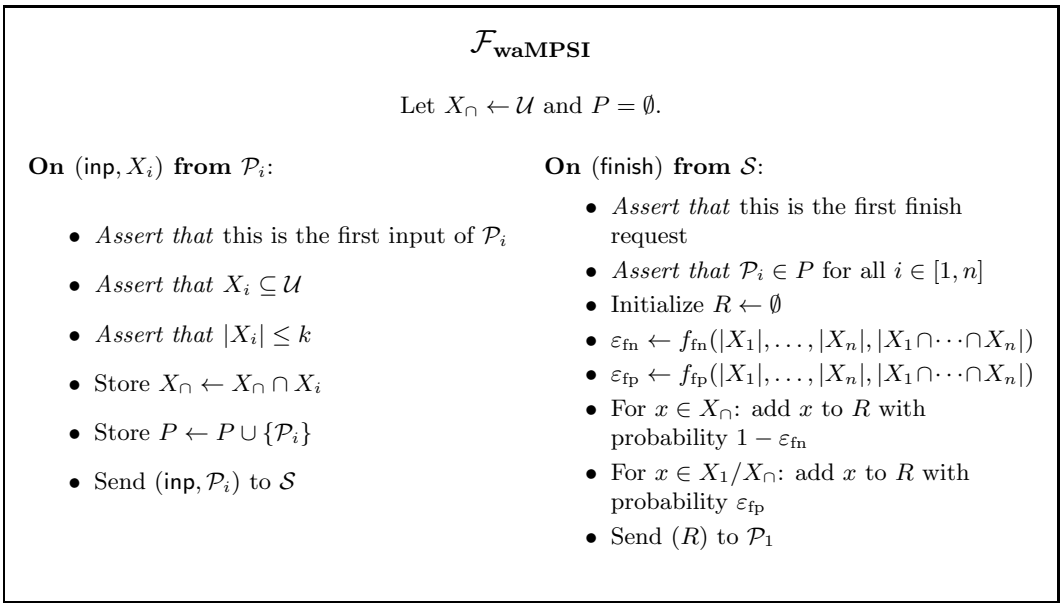

Figure 3: The ideal functionality $\mathcal{F}_{\text{waMPSI}}$.

is only induced by some values that can be permitted to be leaked. E.g. one might argue that the size of the input sets and the size of the exact intersection is not secret. As such, we define a weaker functionality in which ε_{fp} and ε_{fn} are indeed functions over the sizes of the sets: $|X_i|$ for $i = 1, \ldots, n$ and $|X_\cap|$. We denote these functions by f_{fp} and f_{fn}. We present the resulting ideal functionality $\mathcal{F}_{\text{waMPSI}}$ in Figure 3.

4 An abstraction of Bloom filter-based PSI

In this work, we set out to show that Bloom filters are fundamentally flawed. Instead of going through each Bloom filter-based protocol individually and showing that they suffer from security problems, we present an idealized abstraction of Bloom filter-based PSI. After that, we discuss previously proposed protocols and how each inherits the security problems from our idealized abstraction.

4.1 Our idealized abstraction

The idea of our idealized abstraction is to model the behavior of Bloom filters in isolation; without communication between individual parties or use of cryptographic primitives. We present this abstraction Π_{BF} in such a way that it has the same interface as the ideal functionalities defined in the previous section. Π_{BF} is conceptually simple: instead of combining the private sets using an actual intersection, it encodes sets as Bloom filters and combines those instead. It returns the intersection to the leader by returning the leader's elements that are contained in the resulting Bloom filter.

$\Pi_{\mathbf{BF}}$

Let $\hat{X}_\cap \leftarrow 1^m$ and $P = \emptyset$.

On (inp, X_i) **from** $\mathcal{P}_i$:

- *Assert that* this is the first input of $\mathcal{P}_i$
- *Assert that* $X_i \subseteq \mathcal{U}$
- *Assert that* $|X_i| \leq k$
- Store $\hat{X}_\cap \leftarrow \hat{X}_\cap \wedge \mathsf{encode}(X_i)$
- Store $P \leftarrow P \cup \{\mathcal{P}_i\}$
- Send $(\mathsf{inp}, \mathcal{P}_i)$ to $\mathcal{S}$

On (finish) **from** $\mathcal{S}$:

- *Assert that* this is the first finish request
- *Assert that* $\mathcal{P}_i \in P$ for all $i \in [1, n]$
- Initialize $R \leftarrow \emptyset$
- For $x \in X_1$: Add x to R if $\mathsf{contains}(\hat{X}_\cap, x)$
- Send (R) to $\mathcal{P}_1$

4.2 Two-party private set intersections

We first consider two-party protocols, explaining the general workings of these protocols and how it might be possible to create a simulator for them around Π_{BF}. The idea is that any problems inherent to Π_{BF} are inherited by the protocols below.

4.2.1 Debnath and Dutta

Debnath & Dutta [15] propose a PSI protocol using Goldwasser-Micali encryption and inverted Bloom filters. The client $\mathcal{P}_1$ and server $\mathcal{P}_2$ agree on k hash functions to make the Bloom filter, and the client generates an inverted and encrypted Bloom filter and sends it to the server. For each of its elements, the server selects the bins that the element maps to and homomorphically XORs a hash of the element onto these bins. So, if an element is contained in the inverted Bloom filter of $\mathcal{P}_1$, the result is an encryption of a hash of the element. If the element is not contained in it, the result is a distorted hash. The client can extract the intersection by checking which elements match the hashes it receives.

We can simulate this protocol using Π_{BF} with high probability. Notice that while Π_{BF} outputs $\{x \in X_1 \mid \mathsf{contains}(\hat{X}_1 \wedge \hat{X}_2, x)\}$, the protocol by Debnath & Dutta outputs $\{x \in X_2 \mid \mathsf{contains}(\hat{X}_1, x)\}$ with high probability. However, these are the same because Bloom filters do not cause false negatives, so $x \in X_2 \implies \mathsf{contains}(\hat{X}_2, x)$. Besides this, the simulator must still simulate the encryptions that are sent from the client to the server and back.

4.2.2 Davidson and Cid

Davidson and Cid [16] also propose a private set intersection protocol based on encrypted and inverted Bloom filters, which was reformulated by Bay et al. [11]. We discuss this reformulation. [1] The client $\mathcal{P}_1$ encodes their elements X_1 in a Bloom filter as usual and inverts it (i.e. flipping all bits of the filter) before encrypting it. It then sends this filter to the server $\mathcal{P}_2$. For each element $y \in X_2$ of the servers set, the server calculates the corresponding

[1] There seems to be a mistake in the original work because when an element is in the Bloom filter, the sum of the selected bins is 0, so the client would not learn the values that are in the intersection.

bins in the encrypted inverted Bloom filter and sums the values in these bins with outcome S. It then adds S to the encrypted value of y. It returns the pair $(S, S + \texttt{enc}(y))$ to the client who computes the intersection. For an element $y \in X_2$, all its corresponding bins in the inverted Bloom filter have value 0, and thus S is the encryption of 0. If this encrypted value of 0 is added to the encrypted value of y the decryption gives $0 + y$. For any element not in the intersection, the decryption reveals nothing about y. [2] For each pair, the client checks whether the decryption of S equals 0; if so, it decrypts the second value of the pair and assumes it to be in the intersection.

One would roughly simulate this protocol using Π_{BF} as follows. The encrypted inverted Bloom filter would be made up of m random encryptions, and the server returns k pairs of specific encryptions to the client. The elements in the pair are encryptions of 0 if the element is in the intersection returned by Π_{BF}, and random otherwise.

4.3 Multi-party private set intersections

Next, we discuss several Bloom filter-based multi-party private set intersection protocols and how they relate to Π_{BF}.

4.3.1 Bay, Erkin, Hoepman, Samardjiska, and Vos

The protocol proposed by Bay et al. [11] is an extension of [16] in the multi-party variant. The difference is that the server learns the intersection instead of the clients. The adjustments of the protocol are minor. All clients have a private secret key, but the public key corresponding to all private keys is shared. All clients calculate the encrypted inverted Bloom filter with the public key. The server combines the Bloom filters to calculate the sum S for all elements x in its set. Here S is the same as defined in Section 4.2.2. The clients jointly decrypt the encrypted value so that the server can learn if x is in the intersection. This protocol can be simulated using Π_{BF} in a similar way as in Section 4.2.2. A similar protocol was presented by Debnath et al. [10].

4.3.2 Vos, Conti, and Erkin

Vos et al. [12] propose a similar MPSI protocol for large universes using ElGamal encryption. Each of the parties starts by computing a Bloom filter for their input. Then they invert this Bloom filter so that an element x_i is in set X_i if all its corresponding bins have value 0. Then, the protocol securely performs an `OR` operation on all inverted Bloom filters. The resulting Bloom filter is inverted again, and then an element is in the intersection if all its corresponding bins have value 1. This is equivalent to performing an `AND` operation on regular Bloom filters. Vos et al. already show how to simulate the OR protocol, so the simulation around Π_{BF} is straightforward.

4.3.3 Ruan, Yan, Zhou, and Ai

Ruan et al. [17] present an MPSI protocol for unbalanced scenarios where the server $\mathcal{P}_1$ has a significantly larger set. Each of the clients $\mathcal{P}_2, ..., \mathcal{P}_n$ computes a Bloom filter for their input set X_i. The bins of the Bloom filter that contain 0 are randomized to any number but zero and one. This is needed to apply an ElGamal encryption scheme that cannot encrypt

[2] That said, both versions of this protocol seemingly reveal the number of bins that were set in the Bloom filter.

0. A trusted third-party generates an ElGamal public-private key pair (pk, sk) and divides the secret key over the clients sk_i. Each client $\mathcal{P}_i$ for $i \geq 2$ generates a Bloom filter on their input set X_i, randomizes the bins of value 0, and encrypts the filter with the public key. The resulting filter is sent to the server $\mathcal{P}_1$. The server then selects the bins of the Bloom filter $\hat{X}_i$ pertaining to its elements for each $i \geq 2$, homomorphically aggregates them, and sends the results back to the clients. The clients each perform a computation on the received Bloom filter such that the server is able to combine all filters to decrypt the intersection. Simulation using Π_{BF} would be a multi-party extension of the simulation described for the protocol by Debnath et al. [15].

4.3.4 Ruan and Ai

Ruan and Ai [18] made an MPSI protocol for the balanced scenario that is much like the unbalanced scenario. All clients compute the encrypted Bloom filter in the same manner as in [17]. The server does the exact same computation as the clients. All these encrypted Bloom filters are sent to the server, which combines them and sends the combined filter back to the clients. All clients decrypt this Bloom filter using their own private key and send the result to the server. The server then can combine all Bloom filters to find the decrypted Bloom filter of the intersection $\hat{X}_\cap$. The server then performs the normal $\mathsf{contains}(\hat{X}_\cap, x)$ function for all its elements $x \in X_1$.

4.4 Outsourced private set intersections

Since the ideal functionalities nor the idealized abstraction Π_{BF} describe *who* computes something, they also apply to the outsourced computation case, in which most of the computations are performed by a server that does not take part in the protocol. We cover two Bloom filter-based outsourced private set intersection protocols.

4.4.1 Qiu, Zhang, Liu, Yan, and Cheng

Qiu et al. [19] propose a PSI protocol where both clients $\mathcal{P}_1$ and $\mathcal{P}_2$ learn the intersection, and the computation is done by a computational powerful server S. Both clients compute a Bloom filter and encrypt this filter with a shared secret key. They permute the filter with a secret shared permutation π and forward it to the server. The server then computes which indices of the received filters are equal and forwards this set of indices to the clients. Both clients perform an inverted permutation on this set of indices to obtain the indices of the original Bloom filter. For each element x in the set X_1, $\mathcal{P}_1$ checks whether all bins in the Bloom filter are set to 1 and indicated by the server. If so, the element x is considered to be in the intersection. $\mathcal{P}_2$ does the same computation. Due to the fact that the parties undo the permutation, the resulting behavior is exactly the same as that in Π_{BF}. One would still have to simulate the communication between the parties and the server.

4.4.2 Miyaji and Nishida

Miyaji and Nishida [20] propose a multiparty private set intersection based on Bloom filters and a distributed ex-El Gamal encryption. For this protocol, they use a dealer D who does not participate in the intersection and only helps reduce the computational power for the clients. Each client $\mathcal{P}_i$ generates a secret key x_i and a public key g^{x_i}. The jointly public key of the clients is constructed as $pk = \prod_{i=1}^{n} g^{x_i}$. Each of the clients computes their Bloom

filter, encrypts it with the public key pk, and sends it to the dealer. The dealer aggregates the Bloom filters by homomorphically adding them. To compute the Bloom filter of the intersection, the dealer subtracts n (the number of clients) from each entry of the Bloom filter. It then sends the resulting Bloom filter to all the clients for joint decryption. If, for an element, all its corresponding bins have the value 0, the element is considered to be in the intersection. This is functionally the same as the protocol by Bay et al. [11], but without inverting the Bloom filters.

5 Analysis of Bloom filter-based PSI

In this section we show that there are fundamental limitations to the security of Bloom filter-based private set intersections. We do so by showing that a certain indistinguishability notion cannot be met when the false positive probability is not negligible. These problems are caused by the fact that a Bloom filter becomes deterministic when its hash functions are fixed. We first discuss the indistinguishability game that we use to define security, after which we present a distinguisher that reliably tells apart $\mathcal{F}_{\text{waMPSI}}$ from Π_{BF} when false positives occur with non-negligible probability. Based on these results we conclude that the upper bound on the false positive probability p of a Bloom filter must be less than $\frac{0.5}{|\mathcal{U}|-k}$ in practice. After that, we discuss how one can choose concrete Bloom filter parameters to securely realize $\mathcal{F}_{\text{waMPSI}}$ or $\mathcal{F}_{\text{aMPSI}}$ for small values of ε_{fp}.

5.1 The indistinguishability game

In the universal composability (UC) framework [21], the security of a protocol is defined by the advantage with which a distinguisher can tell it apart from the ideal functionality that it is designed to realize. In this section, we present lower bounds for this advantage, which allows us to show that there are many choices of parameters for which Bloom filter-based private set intersection protocols cannot be UC-secure (or the security guarantees would be broken with high probability). To do so, we consider the advantage of a distinguisher $\mathcal{D}$ in distinguishing the abstract Bloom filter-based MPSI protocol Π_{BF} from some ideal functionality $\mathcal{F}$:

$$\text{Adv}_{\text{ind}}^{\Pi_{\text{BF}}}(\mathcal{D}) = 2\left|\Pr[\mathcal{D}(\Pi) = \Pi \mid \Pi \in_R \{\mathcal{F}, \Pi_{\text{BF}}\}] - \frac{1}{2}\right| \tag{5}$$

Specifically, we are interested in analyzing the minimal advantage when distinguishing Π_{BF} from the weakest ideal functionality $\mathcal{F}_{\text{waMPSI}}$, which would allow us to draw the strongest conclusions. In the remainder of this section, we propose such a strong distinguisher, and we show that it only fails with low probability for any value of ε_{fp}.

5.2 A reliable distinguisher

In this section, we show that the false positive probability of a Bloom filter denoted by p must be negligible for such a Bloom filter-based private set intersection protocol to realize a (weakly-)approximate private set intersection. We do so by showing that the idealized Bloom filter-based PSI Π_{BF} can be distinguished with relative ease from $\mathcal{F}_{\text{waMPSI}}$ by a distinguisher that learns the result (so $\mathcal{P}_1$ is corrupted). Let $\varepsilon_{\text{fp}} = f_{\text{fp}}(k, k, 0)$. We propose

the following distinguisher:

$$\mathcal{D}(\Pi) \leftarrow \begin{cases} \mathcal{D}_{\text{FPs}}(\Pi) & \text{If } p(|\mathcal{U}| - k) \geq \varepsilon_{\text{fp}} k \\ \mathcal{D}_{\text{TNs}}(\Pi) & \text{Otherwise} \end{cases} \tag{6}$$

Note that this distinguisher would also apply to $\mathcal{F}_{\text{aMPSI}}$.

Depending on the parameters k, p, $|\mathcal{U}|$, and ε_{fp}, this distinguisher calls $\mathcal{D}_{\text{FPs}}$ or $\mathcal{D}_{\text{TNs}}$. We define $\mathcal{D}_{\text{FPs}}$ as follows:

1. $\mathcal{D}_{\text{FPs}}$ chooses random $X_2 \subset \mathcal{U}$ such that $|X_2| = k$, and computes $\hat{X_2} \leftarrow \mathsf{encode}(X_2)$.
2. $\mathcal{D}_{\text{FPs}}$ chooses $X_1 \subseteq \mathcal{U}/X_2$ such that $|X_1| = k$, maximizing the number of elements $x \in X_1$ for which it holds that $\mathsf{contains}(x, \hat{X_2})$; false positives.
3. $\mathcal{D}_{\text{FPs}}$ lets parties $\mathcal{P}_1$ and $\mathcal{P}_2$ input X_1 and X_2, respectively. It waits for $\mathcal{S}$ to send (finish).
4. $\mathcal{P}_1$ is corrupted, so $\mathcal{D}$ receives (R). If R matches the expected output of Π_{BF}, $\mathcal{D}_{\text{FPs}}$ guesses that $\Pi = \Pi_{\text{BF}}$. Otherwise, it guesses that $\Pi = \mathcal{F}_{\text{waMPSI}}$.

The other distinguisher, $\mathcal{D}_{\text{TNs}}$, follows $\mathcal{D}_{\text{FPs}}$ but it maximizes the number of true negatives, so step 2 is different:

2. $\mathcal{D}_{\text{TNs}}$ chooses $X_1 \subseteq \mathcal{U}/X_2$ such that $|X_1| = k$, maximizing the number of elements $x \in X_1$ for which $\mathsf{contains}(x, \hat{X_2})$ is false; true negatives.

It is easy to extend the distinguisher to more than two parties. The following two lemmas express the probability with which these distinguishers fail. We use these results to derive $\text{Adv}^{\Pi_{\text{BF}}}_{\text{ind}}(\mathcal{D})$, see (5).

Lemma 1. *$\mathcal{D}_{FPs}$ always correctly identifies Π_{BF}, but it sometimes misclassifies $\mathcal{F}_{waMPSI}$. It does so with the following probability:*

$$\Pr[\mathcal{D}_{FPs}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] = \sum_{i=0}^{k-1} \Pr[FPs = i] \varepsilon_{fp}^{i} (1 - \varepsilon_{fp})^{k-i} + \Pr[FPs \geq k] \varepsilon_{fp}^{k}$$

Proof. $\mathcal{D}_{\text{FPs}}$ maximizes the number of false positives in X_1, but it is not guaranteed to find such elements in $\mathcal{U}/X_2$. We use $\Pr[\text{FPs} = i]$ to denote the probability with which $\mathcal{D}_{\text{FPs}}$ finds i false positives. The final probability is:

$$\begin{aligned} \Pr[\mathcal{D}_{\text{FPs}}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] = \sum_{i=0}^{k-1} & \Pr[\text{FPs} = i] \cdot \Pr[\mathcal{D}_{\text{FPs}}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}} \mid \text{FPs} = i] \\ & + \Pr[\text{FPs} \geq k] \cdot \Pr[\mathcal{D}_{\text{FPs}}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}} \mid \text{FPs} \geq k] \end{aligned}$$

The probability that $\mathcal{D}_{\text{FPs}}$ misclassifies $\mathcal{F}_{\text{waMPSI}}$ is the probability that Π_{BF} would return R on inputs X_1 and X_2. So, each false positive in X_1 is included in R, which happens with probability $\varepsilon_{\text{fp}}^{i}$. Moreover, each true negative should *not* be in R, which happens with probability $(1 - \varepsilon_{\text{fp}})^{k-i}$. In other words:

$$\Pr[\mathcal{D}_{\text{FPs}}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}} \mid \text{FPs} \geq k] = \varepsilon_{\text{fp}}^{i} (1 - \varepsilon_{\text{fp}})^{k-i}$$

Notice that when $\text{FPs} \geq k$, the distinguisher simply chooses k false positives. As a result: $\Pr[\mathcal{D}_{\text{FPs}}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}} \mid \text{FPs} = j] = \varepsilon_{\text{fp}}^{k}$ for all $j \geq k$. This proves our lemma. □

Lemma 2. *$\mathcal{D}_{TNs}$ always correctly identifies Π_{BF}, but it sometimes misclassifies $\mathcal{F}_{waMPSI}$. It does so with the following probability:*

$$\Pr[\mathcal{D}_{TNs}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] = \sum_{i=0}^{k-1} \Pr[TNs = i](1-\varepsilon_{fp})^i \varepsilon_{fp}^{k-i} + \Pr[TNs \geq k](1-\varepsilon_{fp})^k.$$

Proof. This proof follows similarly to that of Lemma 1. □

The condition $p(|\mathcal{U}| - k) \geq \varepsilon_{\text{fp}} k$ marks the point beyond which one expects to find more false positives in $\mathcal{U}/X_2$ than the number of false positives that one expects Π_{BF} to output.

In the remainder of this section, we define three different scenarios based on two conditions, depending on the median of the binomial distribution of Pr[FPs]. These conditions are as follows (see Appendix A for more details):

- When $p < \frac{1}{|\mathcal{U}|-k}$, the median is at or below $\Pr[\text{FPs} = 0]$, so $\Pr[\text{FPs} = 0] \geq \frac{1}{2}$. See Lemma 7.
- When $p > \frac{k-1}{|\mathcal{U}|-k}$, the median is at or above $\Pr[\text{FPs} = k]$, so $\Pr[\text{FPs} = k] \geq \frac{1}{2}$. See Lemma 8.

5.3 Upper bounds on the failure probability

We want to obtain upper bounds on the failure probability independent of ε_{fp}. This allows us to make statements about the security gap that arises when trying to realize $\mathcal{F}_{\text{waMPSI}}$ using a Bloom filter-based protocol regardless of the choice of ε_{fp} in the ideal functionality. Our main result is an upper bound on the failure probability of our distinguisher for all values of p, which we present in Theorem 6. We provide a summary in Figure 4, showing among others, that the attack succeeds with high probability when $\frac{1}{|\mathcal{U}|-k} \leq p \leq \frac{k-1}{|\mathcal{U}|-k}$, or when $p > \frac{k-1}{|\mathcal{U}|-k}$ and k is large.

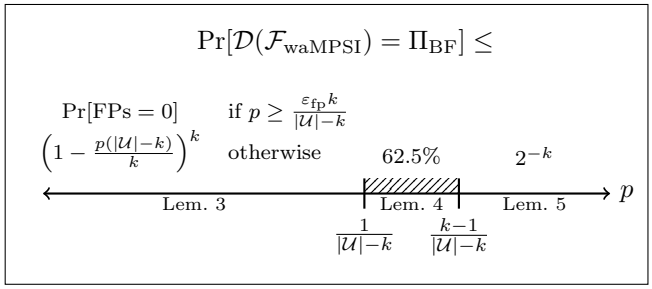

Figure 4: Upper bounds on the distinguisher's failure probability for different values of the false positive probability p. The bounds depend on the size of the input sets k, the size of the universe $|\mathcal{U}|$, and the false positive probability $\varepsilon_{\text{fp}} \leftarrow f_{\text{fp}}(k, k, 0)$ of $\mathcal{F}_{\text{waMPSI}}$. The attack success probability cannot be made negligible in the shaded area.

Our first lemma considers the case where p is so small that $\Pr[\text{FPs} = 0]$ is the most likely (and the same holds for $\Pr[\text{TNs} \geq k]$). We obtain different bounds depending on whether $\mathcal{D} = \mathcal{D}_{\text{FPs}}$ or $\mathcal{D} = \mathcal{D}_{\text{TNs}}$. For the proofs we often use the fact that $\varepsilon_{\text{fp}} \leq (1 - \varepsilon_{\text{fp}})$ for $\varepsilon_{\text{fp}} \leq \frac{1}{2}$.

Lemma 3. *If $p < \frac{1}{|\mathcal{U}|-k}$, $k \geq 2$, $|\mathcal{U}| \geq 2k$, and $\varepsilon_{fp} \leq \frac{1}{2}$, we can bound the failure probability of $\mathcal{D}$ from above:*

$$\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] \leq \begin{cases} \Pr[FPs = 0] & \text{If } p \geq \frac{\varepsilon_{fp} k}{|\mathcal{U}|-k} \\ \left(1 - \frac{p(|\mathcal{U}|-k)}{k}\right)^k & \text{otherwise.} \end{cases}$$

Proof. If $p \geq \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}|-k}$, then $\mathcal{D} = \mathcal{D}_{\text{FPs}}$ (see (6)). By Lemma 7, we have that $\Pr[\text{FPs} = 0] \geq \Pr[\text{FPs} = i]$ for $i = 1, 2, \ldots$, so (see Lemma 1):

$$\begin{aligned} \Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] &\leq \sum_{i=0}^{k-1} \Pr[\text{FPs} = 0] \cdot \varepsilon_{\text{fp}}^i (1 - \varepsilon_{\text{fp}})^{k-i} + \Pr[\text{FPs} = 0] \cdot \varepsilon^k \\ &= \Pr[\text{FPs} = 0] \cdot \sum_{i=0}^{k} \varepsilon_{\text{fp}}^i (1 - \varepsilon_{\text{fp}})^{k-i}. \end{aligned}$$

Claim: the term $\sum_{i=0}^{k-1} \varepsilon_{\text{fp}}^i (1 - \varepsilon_{\text{fp}})^{k-i}$ is at most 1, which it achieves when $\varepsilon_{\text{fp}} = 0$. This proves the first part of the lemma. To prove this claim, notice that this term effectively models a sum of sequences of k Bernoulli trials. Specifically, the sum of probabilities that the first i trials succeed with probability ε_{fp} each, and the next $k - i$ trials fail with probability $1 - \varepsilon_{\text{fp}}$ (ignoring the cases where trials succeed and fail in a different pattern). This is a well-defined probability distribution, so the term does not exceed 1.

In the other case, when $p < \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}|-k}$, $\mathcal{D} = \mathcal{D}_{\text{TNs}}$. By $\varepsilon_{\text{fp}} \leq (1 - \varepsilon_{\text{fp}})$, we have that $(1 - \varepsilon_{\text{fp}})^{k-i} \varepsilon_{\text{fp}}^i \leq (1 - \varepsilon_{\text{fp}})^k$ for $i = 0, 1, \ldots, k - 1$, so (see Lemma 2):

$$\begin{aligned} \Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] &\leq \sum_{i=0}^{k-1} \Pr[\text{TNs} = i] \cdot (1 - \varepsilon_{\text{fp}})^k + \Pr[\text{TNs} \geq k] \cdot (1 - \varepsilon_{\text{fp}})^k \\ &= (1 - \varepsilon_{\text{fp}})^k. \end{aligned}$$

Recall that $p < \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}|-k}$, so ε_{fp} is bounded from below:

$$\varepsilon_{\text{fp}} > \frac{p(|\mathcal{U}| - k)}{k} .$$

The largest value that $\Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}]$ can take occurs when ε_{fp} is as small as possible. This proves the second part of the lemma. □

In the first case, when $p \geq \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}|-k}$, the attack's failure rate is bounded by the probability that the distinguisher cannot find any false positives in $\mathcal{U}/X_2$. This is the same probability with which a distinguisher could tell apart Π_{BF} from an exact MPSI $\mathcal{F}_{\text{MPSI}}$. In other words, in this scenario, Bloom filters are not suitable when they approximate the intersection. In the other case, notice that:

$$\left(1 - \frac{p(|\mathcal{U}| - k)}{k}\right)^k \approx e^{-p(|\mathcal{U}|-k)} \leq 2^{-p(|\mathcal{U}|-k)} . \tag{7}$$

So, for the attack to succeed with negligible probability (i.e., the attack to fail with high probability), the exponent $-p(|\mathcal{U}| - k)$ must remain a small negative number. For example, for the attack to fail with overwhelming probability, we must have that $p \ll (|\mathcal{U}| - k)^{-1}$.

Our second lemma considers the case where p is neither very large nor small. In this case, the attack succeeds with high probability.

Lemma 4. *If $\frac{1}{|\mathcal{U}|-k} \leq p \leq \frac{k-1}{|\mathcal{U}|-k}$, $k \geq 2$, $|\mathcal{U}| \geq 2k$, and $\varepsilon_{fp} \leq \frac{1}{2}$, we can bound the failure probability of $\mathcal{D}$ from above:*

$$\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] \leq 62.5\% \; .$$

Proof. If $p \geq \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}|-k}$, then $\mathcal{D} = \mathcal{D}_{\text{FPs}}$ (see (6)). Since $0 \leq \varepsilon_{\text{fp}} \leq 0.5$, we have that $\varepsilon_{\text{fp}}(1-\varepsilon_{\text{fp}})^{k-1} \geq \varepsilon_{\text{fp}}^{i}(1-\varepsilon_{\text{fp}})^{k-i}$ for $i = 1, 2, \ldots, k$, so (see Lemma 1):

$$\Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] \leq \Pr[\text{FPs} = 0] + (1 - \Pr[\text{FPs} = 0]) \cdot \varepsilon_{\text{fp}}(1-\varepsilon_{\text{fp}})^{k-1} \; .$$

Next, we show that the supremum of $\varepsilon_{\text{fp}}(1-\varepsilon_{\text{fp}})^{k-1}$ occurs when $\varepsilon_{\text{fp}} = k^{-1}$, by checking when its derivative equals 0:

$$\begin{aligned} 0 &= (1-\varepsilon_{\text{fp}})^{k-1} - (k-1)\varepsilon_{\text{fp}}(1-\varepsilon_{\text{fp}})^{k-2} \; , \\ &= (1-\varepsilon_{\text{fp}})^{k-2}((1-\varepsilon_{\text{fp}}) - (k-1)\varepsilon_{\text{fp}}) \; , \\ &= (1-\varepsilon_{\text{fp}})^{k-2}(1-k\varepsilon_{\text{fp}}) \; . \end{aligned}$$

The only valid root occurs when the rightmost term is 0; i.e., $\varepsilon_{\text{fp}} = k^{-1}$. We ignore the bound $p \geq \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}|-k}$, making our final upper bound looser. We get that:

$$\begin{aligned} \Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] &\leq \Pr[\text{FPs} = 0] + (1 - \Pr[\text{FPs} = 0])\frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-1} \\ &\leq 0.5 + 0.5 \cdot \frac{1}{k}\left(1 - \frac{1}{k}\right)^{k-1} \\ &\leq 0.5 + 0.5 \cdot 0.5(0.5)^{1} = 62.5\% \; . \end{aligned}$$

This works because $\Pr[\text{FPs} = 0] < 0.5$ due to Lemma 7, and that $k^{-1}(1-k^{-1})^{k-1}$ is monotonically decreasing for $k = 2, 3, \ldots$ (we do not prove this), so we fill in $k = 2$. This concludes the proof for $\mathcal{D} = \mathcal{D}_{\text{FPs}}$.

In the other case, when $p < \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}|-k}$, $\mathcal{D} = \mathcal{D}_{\text{TNs}}$. We get a similar situation:

$$\Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] \leq (1 - \Pr[\text{TNs} \geq k])(1-\varepsilon_{\text{fp}})^{k-1}\varepsilon_{\text{fp}} + \Pr[\text{TNs} \geq k],$$

where $\Pr[\text{TNs} \geq k] < 0.5$ due to Lemma 8, so we obtain the same bound. □

Our final lemma relating to these upper bounds is for the case where p is large, such that $\Pr[\text{FPs} \geq k]$ is the most likely (and the same holds for $\Pr[\text{TNs} = 0]$). In this case, we do not obtain different bounds depending on $\mathcal{D}$; the only possible case is $\mathcal{D} = \mathcal{D}_{\text{FPs}}$.

Lemma 5. *If $p > \frac{k-1}{|\mathcal{U}|-k}$, $k \geq 2$, $|\mathcal{U}| \geq 2k$, and $\varepsilon_{fp} \leq \frac{1}{2}$, we can bound the failure probability of $\mathcal{D}$ from above:*

$$\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] \leq 2^{-k} \; .$$

Proof. If $p(|\mathcal{U}| - k) < \varepsilon_{\text{fp}} k$, then $\mathcal{D} = \mathcal{D}_{\text{TNs}}$ (see (6)). However, we have that:

$$p < \frac{\varepsilon_{\text{fp}} k}{|\mathcal{U}| - k} \leq \frac{k-1}{|\mathcal{U}| - k} < p \; ,$$

which is a contradiction, so $\mathcal{D} = \mathcal{D}_{\text{FPs}}$.

When $\mathcal{D} = \mathcal{D}_{\text{FPs}}$ we claim that an upper bound is 2^{-k} thus we will show that

$$\sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^i(1-\varepsilon_{\text{fp}})^{k-i} + \Pr[\text{FPs} \geq k] \cdot \varepsilon_{\text{fp}}^k \leq 2^{-k}.$$

Note that $\varepsilon_{\text{fp}} \leq \frac{1}{2}$, thus the LHS is smaller than $\sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^i(1-\varepsilon_{\text{fp}})^{k-i} + \Pr[\text{FPs} \geq k]2^{-k}$, which gives us the following equation:

$$\begin{aligned}\sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^i(1-\varepsilon_{\text{fp}})^{k-i} &\leq 2^{-k}(1 - \Pr[\text{FPs} \geq k]) \\ &\leq 2^{-k}\Pr[\text{FPs} < k].\end{aligned}$$

By Lemma 9, the LHS is a monotonic increasing function, so it reaches its maximum at the edge of the domain ($\varepsilon_{\text{fp}} = \frac{1}{2}$). The maximum is given by:

$$\begin{aligned}\sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^i(1-\varepsilon_{\text{fp}})^{k-i} &= \sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\left(\frac{1}{2}\right)^k \\ &= 2^{-k}\sum_{i=0}^{k-1}\Pr[\text{FPs} = i] \\ &= 2^{-k}\Pr[\text{FPs} < k].\end{aligned}$$

This is equal to the upper bound we proposed, so 2^{-k} is indeed an upper bound. □

Our main result is the following theorem, which combines the above lemmas.

Theorem 6. *Given $k \geq 2$, $|\mathcal{U}| \geq 2k$, and $0 \leq \varepsilon_{fp} \leq \frac{1}{2}$, we have the following upper bounds for $\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}]$, as summarized in Figure 4:*

- $\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] \leq 2^{-k}$ *if* $p > \frac{k-1}{|\mathcal{U}|-k}$.
- $\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] \leq 62.5\%$ *if* $\frac{1}{|\mathcal{U}|-k} \leq p \leq \frac{k-1}{|\mathcal{U}|-k}$.
- *Otherwise,* $\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] \leq \Pr[FPs = 0]$ *if* $p \geq \frac{\varepsilon_{fp}k}{|\mathcal{U}|-k}$.
- *Otherwise,* $\Pr[\mathcal{D}(\mathcal{F}_{waMPSI}) = \Pi_{BF}] \leq \left(1 - \frac{p(|\mathcal{U}|-k)}{k}\right)^k$.

Proof. We refer the reader to the following lemmas:

- If $p > \frac{k-1}{|\mathcal{U}|-k}$, then this holds by Lemma 5.
- If $\frac{1}{|\mathcal{U}|-k} \leq p \leq \frac{k-1}{|\mathcal{U}|-k}$, then this holds by Lemma 4.
- If $p < \frac{1}{|\mathcal{U}|-k}$ and $p \geq \frac{\varepsilon_{\text{fp}}k}{|\mathcal{U}|-k}$, then this holds by Lemma 3.
- If $p < \frac{1}{|\mathcal{U}|-k}$ and $p < \frac{\varepsilon_{\text{fp}}k}{|\mathcal{U}|-k}$, this holds by Lemma 3. □

5.4 Obtaining secure parameters

While the bounds we derive above provide lower bounds for the success probability of the attack, they may underestimate it. Moreover, they quantify over all values of ε_{fp}, while in practice, one may wish to only choose small values (or there would be many false positives). We now consider how to choose the smallest Bloom filter for which our attack does not work. We formulate this as the following constrained optimization problem, in which we want to realize $\mathcal{F}_{\text{waMPSI}}$ with at most false positive probability $\varepsilon^*_{\text{fp}}$:

$$\begin{aligned} \max \quad p \quad \text{s.t.} \quad & \text{Adv}^{\Pi_{\text{BF}}}_{\text{ind}}(\mathcal{D}) \leq 2^{-\lambda} \\ & 0 \leq \varepsilon_{\text{fp}} \leq \varepsilon^*_{\text{fp}} \\ & p < \tfrac{1}{|\mathcal{U}|-k} \end{aligned} \tag{8}$$

After all, the larger the false positive probability p, the smaller the Bloom filter can be. Note that the last constraint is implied by the first constraint when $k > \lambda$. We assume this to be the case because λ, the statistical security parameter that decides the chance of the attack succeeding, is typically a value such as 40 or 128, whereas the number of elements in a set k can be orders of magnitude higher. We note that the constraints imply that $\varepsilon^*_{\text{fp}} \leq \frac{1}{k}$.

Since we only look for $p < \frac{1}{|\mathcal{U}|-k}$, and since $\mathcal{D}$ only misclassifies $\mathcal{F}_{\text{waMPSI}}$, we get that:

$$\begin{aligned} \text{Adv}^{\Pi_{\text{BF}}}_{\text{ind}}(\mathcal{D}) &= \left|1 - \frac{1}{2}\right| + \left|1 - \Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] - \frac{1}{2}\right| \\ &= 1 - \Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] \ . \end{aligned}$$

This uses that $\Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}] \leq \frac{1}{2}$.

Instead of using the bounds described in Figure 4, we will use the equations from Lemmas 1 & 2 as not to underestimate the distinguisher. That said, we use a heuristic to decide for which ε_{fp} we have that $\Pr[\mathcal{D}(\mathcal{F}_{\text{waMPSI}}) = \Pi_{\text{BF}}]$ is maximal. Specifically, when $\mathcal{D} = \mathcal{D}_{\text{FPs}}$ and p tends to 0, most of the probability mass occurs at $\Pr[\text{FPs} = 0]$ (see Lemma 7). It is easy to see that $\varepsilon_{\text{fp}} = 0$ maximizes the failure probability. For the case when $\mathcal{D} = \mathcal{D}_{\text{TNs}}$, most of the probability mass occurs at $\Pr[\text{TNs} \geq k]$, so we also want to minimize ε_{fp}. However, when $\varepsilon_{\text{fp}} = 0$, we always get that $\mathcal{D} = \mathcal{D}_{\text{FPs}}$. We get that:

$$\text{Adv}^{\Pi_{\text{BF}}}_{\text{ind}}(\mathcal{D}) \approx 1 - \Pr[\text{FPs} = 0] = 1 - (1-p)^{|\mathcal{U}|-k} \ . \tag{9}$$

This corresponds to the same scenario as exact MPSI ($\mathcal{F}_{\text{MPSI}}$), in which any false positive occurring is enough for the distinguisher to succeed. Notice that it is not enough for the probability of a false positive to occur in a set of k elements to be negligible; this would only defend against semi-honest parties. In the augmented semi-honest model and beyond, in which corrupt parties can choose their own inputs, the false positive probability must decrease when the gap between $|\mathcal{U}|$ and k increases.

Finally, given the largest p, we can generate Bloom filter parameters m and h as described in Section 2. We provide examples of these parameters in Table 1 corresponding to $|\mathcal{U}| = 2^{32}$, and compare them against old parameters, as used by previous work. We generate these parameters by iterating over $p = 2^{-1}, 2^{-2}, \ldots$ until the constraints from (8) hold, using the approximation from (9). For the old parameters we use $p = \frac{1}{k}$, because this would be expected to realize $\mathcal{F}_{\text{waMPSI}}$ with $\varepsilon_{\text{fp}} = \frac{1}{k}$. Note that the parameters we propose only protect against this distinguisher, but stronger attacks may exist.

Table 1: Parameters for $\mathcal{F}_{\text{waMPSI}}$ with $\varepsilon_{\text{fp}} \leq k^{-1}$ and $|\mathcal{U}| = 2^{32}$. Secure parameters are larger by an order of magnitude.

Setting		Old parameters			$\mathcal{F}_{\text{waMPSI}}$ parameters			
λ	k	p	h	m	p	h	m	Factor
40	256	2^{-6}	6	2,962	2^{-72}	72	26,645	9×
	4096	2^{-8}	8	70,922	2^{-72}	72	425,522	6×
	65536	2^{-12}	12	1,512,788	2^{-72}	72	6,807,543	4.5×
80	256	2^{-6}	6	2,962	2^{-112}	112	41,447	13×
	4096	2^{-8}	8	70,922	2^{-112}	112	661,922	9.33×
	65536	2^{-12}	12	1,512,788	2^{-112}	112	10,589,510	7×
128	256	2^{-6}	6	2,962	2^{-160}	160	59,210	20×
	4096	2^{-8}	8	70,922	2^{-160}	160	945,602	13.33×
	65536	2^{-12}	12	1,512,788	2^{-160}	160	15,127,871	10×

5.5 A note on PSI-cardinality

While the distinguisher $\mathcal{D}$ examines the set output by the protocol Π to determine if it is interacting with $\mathcal{F}_{\text{waMPSI}}$ or Π_{BF}, we note that it is also possible to base this choice solely on the cardinality of the output. As such, this slightly weaker distinguisher $\mathcal{D}'$ would also apply to PSI-cardinality that protocols, which only output the size of the resulting set $|R|$. The condition at which $\mathcal{D}'$ classifies Π as Π_{BF} is when $|R| \neq \text{FPs}$. If $\varepsilon_{\text{fp}} \approx 0$, then $\text{Adv}_{\text{ind}}^{\Pi_{\text{BF}}}(\mathcal{D}') \approx \text{Adv}_{\text{ind}}^{\Pi_{\text{BF}}}(\mathcal{D})$. Since the weakness originates in the Bloom filter, this security problem also affects quantum PSI-cardinality protocols based on Bloom filters [22].

6 Practical attack on Bloom filter-based PSI

In this section, we study membership inference attacks performed by a corrupted leader in the augmented semi-honest model. We present a practical attack on the parameters used by many works [11, 12, 10] discussed in Section 4.

6.1 Security game

An intuitive definition of a private set intersection's security property is that a leader should only learn about another party's elements when they appear in the intersection. In other words, a leader cannot infer information about the elements that do not appear in the intersection. We formalize this using the concept of membership inference attacks, in which the corrupted leader (the adversary) must guess an element in the other party's set. For simplicity, we only consider two parties, but the concept extends beyond the two-party setting.

To incorporate the fact that the leader learns the result of the protocol, we model membership inference against PSI as an adaptive security game in which the adversary consists of two algorithms: $\mathcal{A}_{\text{pre}}$ and $\mathcal{A}_{\text{post}}$. The first algorithm inputs $\mathcal{U}_2$, which is a superset of the victim's set X_2, for which it holds that $|X_2| = k$. $\mathcal{A}_{\text{pre}}$ outputs the leaders input to the protocol Π and an element $t \in \mathcal{U}_2$. The second algorithm $\mathcal{A}_{\text{post}}$ inputs the

element t and the result of the protocol, and outputs the guess of the adversary. We assume the adversary already has access to all public parameters (for us, this includes the Bloom filter's hash functions). We define an adversary's probability of beating the membership inference security game against a PSI protocol Π as follows:

$$\Pr[\mathcal{A} \text{ succeeds} \mid u] = \Pr\left[\begin{array}{l|l} & \mathcal{U}_2 \subseteq_R \mathcal{U} \text{ s.t. } |\mathcal{U}_2| = u \\ \xrightarrow{(\mathsf{inp}, X_1)} \boxed{\Pi} \xleftarrow{(\mathsf{inp}, X_2)} & (X_1, t) \leftarrow \mathcal{A}_{\text{pre}}(\mathcal{U}_2) \\ \quad \xleftarrow{(R)} & \quad \text{s.t. } (t \in \mathcal{U}_2) \wedge (t \notin X_1) \\ \quad \mathcal{A}_{\text{post}}(t, R) = b & b \in_R \{0,1\} \\ & X_2 \subseteq_R \mathcal{U}_2 \text{ s.t. } (|X_2| = k) \wedge \\ & \quad (t \in X_2 \iff b = 1) \end{array}\right] \tag{10}$$

The advantage of an adversary $\mathcal{A} = (\mathcal{A}_{\text{pre}}, \mathcal{A}_{\text{post}})$ over random guessing is:

$$\text{Adv}^{\Pi}_{\text{memb}}(\mathcal{A}, |\mathcal{U}_2|) = 2\left|\Pr[\mathcal{A} \text{ succeeds} \mid u = |\mathcal{U}_2|] - \frac{1}{2}\right| \tag{11}$$

If one can show that the advantage is negligible for all adversaries, then the protocol is secure against membership inference attacks. In the next subsections, we present attacks in which the advantage is non-negligible when the false positive probability is non-negligible.

6.2 Proposed attack

The foundation of our proposed attack lies in two key observations: First, when a Bloom filter's hash functions are known, we may determine which elements in $\mathcal{U}$ have overlapping bins when encoded in a Bloom filter. Second, for Bloom filter encodings of most subsets of $\mathcal{U}$, there are bins that are set only by a single element of this subset. We first expand on both observations.

We use the first observation to find probabilities of Bloom filter false positives. Given a set X, and its Bloom filter encoding, $\hat{X}$, we observe that knowledge of the hash functions collapses the false positive probability. Let us denote $\mathsf{contains}(\hat{X}, x)$ by $x \in \hat{X}$. For any $y \notin X$, we verify whether y is a false positive in $\hat{X}$ by calculating $H_i(y)$, for $i \in 1, \ldots, h$, and verifying that each bin is set. If we know X, the conditional probabilities $\Pr[y \in \hat{X} \mid X]$ become deterministic. If we do not know X, yet instead, we know the distribution of X sampled from $\mathcal{U}$, we find $\Pr[y \in \hat{X}] = \sum_{X, y \in \hat{X}} \Pr[X]$.

The second observation helps detect the target's presence in the Bloom filter encoding. Some bins may *uniquely identify* an element x amongst a subset $\mathcal{U}_2$ of the total universe. In other words, a bin b may exist such that only one element $x \in \mathcal{U}_2$ maps to this bin with any hash function. From Section 2, recall that one chooses the parameters of a Bloom filter such that, for k elements, the probability of any element having no unique bin is at most p. However, an element y not present in $\mathcal{U}_2$ may hash to this bin. When y is a false positive in a Bloom filter encoding of a subset of $\mathcal{U}_2$, we know that this subset contained x. The probability of finding any element with a uniquely identifying bin depends on both $\mathcal{U}_2$ and k.

We define the adversary's universe as $\mathcal{U}_1 = \mathcal{U} \setminus \mathcal{U}_2$. We realize A_{pre} by means of the following steps:

1. We find an optimal target $t \in \mathcal{U}_2$ which can be uniquely identified amongst the other elements of $\mathcal{U}_2$,

2. we reduce $\mathcal{U}_1$ to C, leaving only elements that hash to the bins that are unique to t and which have a non-zero probability of occurring as a false positive,

3. we rank all elements $y \in C$ based on the probability of appearing in the Bloom filter encoding of X_2,

4. we return t and the top k elements of C.

To determine the bit b, by testing the result of the protocol execution R. If R is non-empty, we successfully identified t and $\mathcal{A}_{\text{post}}$ returns 1. Otherwise, $\mathcal{A}_{\text{post}}$ guesses that t is not part of X_2 and returns 0.

6.2.1 Choosing the target

As the target t, we select an element from $\mathcal{U}_2$ with the least overlap with other bins of other elements in $\mathcal{U}_2$. Bins that are unique to the target t can be used to conclusively decide whether the target is included in the input set X_2 of the victim $\mathcal{P}_2$. We find an optimal target $t \in \mathcal{U}_2$ using an exhaustive search:

$$t = \min_{x \in \mathcal{U}_2} |\{x' \in \mathcal{U}_2 \setminus \{x\} \mid \exists (i,j) : H_i(x) = H_j(x')\}|. \tag{12}$$

6.2.2 Filtering the attack universe

Given a target t the adversary $\mathcal{P}_1$ searches for an attack input set X_1 that maximizes the detection rate of the target element t. We can filter any element from $\mathcal{U}_1$, which does not contribute to detecting t. We use the following two criteria to narrow down $\mathcal{U}_1$ to the candidate subset C: First, for any candidate $y \in C$, at least one hash function must exist such that y is mapped to a bin that *uniquely identifies* t from the other elements in $\mathcal{U}_2$. Second, for each hash function H, H maps y to a bin that at least one element in $\mathcal{U}_2$ hashes to.

6.2.3 Selecting the attack set

If for the number of candidates C it holds that $|C| \leq k\ |C|$, we set $X_1 = C$. Otherwise, we aim to find the input set $X_1 \subset C$ with $|X_1| = k$ that is most likely to successfully identify t. Calculating this 'effectiveness' of an input set X_1 requires that we assume knowledge of the distribution of $X_2 \subseteq \mathcal{U}_2$. For any X_1, the probability that at least one of the elements exists in the Bloom filter encoding is equal to:

$$\Pr[\exists y \in X_1 \mid y \in \hat{X}_2] = \sum_{i=1}^{|X_1|} \Pr[y_i \in \hat{X}_2 \mid y_j \notin \hat{X}_2 \forall j < i].$$

For the sake of reducing the computational cost of finding an attack set, we disregard the probabilistic dependency of the elements of C on one another. This gives us the following approximation:

$$\Pr[\exists y \in X_1 \mid y \in \hat{X}_2] \approx \sum_{y \in X_1} \Pr[y \in \hat{X}_2]. \tag{13}$$

For the remaining part of this section, we assume that $\mathcal{P}_2$ uniformly samples its private input set X_2 from $\mathcal{U}_2$. This aligns with our definition of $\text{Adv}^{\Pi_{\text{BF}}}_{\text{memb}}$. The probability $\Pr[y \in \hat{X}_2]$

equals the number of private input sets X_2 that activate all bins of y, divided by the number of different input sets. We rank all elements in C based on the number of X_2 for which $y \in \hat{X}_2$. Afterward, we select the k elements with the highest probability of appearing in the Bloom filter as the attack set.

6.2.4 Finding the number of X_2 for which $y \in \hat{X}_2$

To find the number of X_2 for which $y \in \hat{X}_2$ holds, we focus on finding a formula for the number of possible input sets X_2 that activate the bins of y. We can construct lists $b_1, .., b_h$, where each list b_i contains the elements of $\mathcal{U}_2$ that hashes to bin $H_i(y)$, i.e.

$$b_i = \{x \in \mathcal{U}_2 \mid \exists j \in 1..h H_j(x) = H_i(y)\}.$$

We distinguish two cases: In the first case, all elements from all these lists b_i are distinct. In the second case, one or more elements appear in at least two bins. We focus on the first case and later show how the second case can be reduced to the first.

All elements are distinct. For the first case, we emulate picking elements from $\mathcal{U}_2$ to create X_2. Let $\mathcal{U}'$ denote the universe we are picking from, and $|\mathcal{U}'|$ the number of elements in this universe. The goal is to fill the k' spots of the private input set with elements from $\mathcal{U}'$ with the following restrictions:

- None of the elements can appear more than once.
- The order of the elements does not matter, i.e. "1 2" and "2 1" are considered the same.
- There are r lists $b_1, .., b_r$ each consisting of a number of elements. These lists are all pairwise distinct, i.e., for each $i, j \leq r$ holds that $b_i \cap b_j = \emptyset$. From each of these lists, at least one element should be taken.

Example 2. *Take $\mathcal{U}' = \{1..6\}$, $b_1 = \{1, 2\}$, $b_3 = \{3\}$ and $k' = 3$. The order does not matter. Therefore, we choose to first write the element of bin 1, then the element of bin 2, and then fill the last spot.*

1 3 2	~~2 3 1~~
1 3 4	2 3 4
1 3 5	2 3 5
1 3 6	2 3 6

See that 1 3 2 *is equivalent to* 2 3 1 *and should not be counted. There are thus 7 options.*

Example 2 shows that the elements we can choose for the last bin must not occur in the sequence yet, and neither must create a sequence that has already been counted. We solve this by creating $r + 1$ different lists. We use b_0 to denote $\mathcal{U}' \setminus (b_1 \cup b_2 \cup \cdots \cup b_r)$; The list that contains all elements that occur in no other. This simplifies the problem to dividing k' choices over $r + 1$ lists, where for b_i $i = 1 \ldots r$ at least one element must be chosen. We calculate the answer to this problem by expressing it as a product of the generating function, and then finding the coefficient of the term with degree k'.

For each list b_i we find a corresponding generating function that expresses in how many ways we can choose elements from that list. The exponent of the variable z denotes the

number of elements chosen from the list b_i. The coefficient of z^j tells us in how many ways j elements can be chosen from b_i. Since we pick without replacement, choosing j elements from b_i can be done in $\binom{|b_i|}{j}$ ways. Given that we may pick no elements from b_0, we find the following functions for b_0 and b_i with $i \neq 0$ respectively:

$$f(|b_0|, z) = \sum_{j=0}^{|b_0|} \binom{|b_0|}{j} z^j, \quad g(|b_i|, z) = \sum_{j=1}^{|b_i|} \binom{|b_i|}{j} z^j.$$

The combined formula allows us to determine the number of ways to pick k' elements for X_2 such that $y \in \hat{X}_2$, by calculating the coefficient of the term $z^{k'}$. The formula for the number of combinations is as follows:

$$[z^{k'}] \colon f\left(|b_0|, z\right) \cdot \prod_{i=1}^{r} g\left(|b_i|, z\right). \tag{14}$$

Not all elements are distinct. If an element $x \in \mathcal{U}_2$ activates two bins of the Bloom filter that the element y hashes to, we lose the independence requirement and, as such, Eq. (14) does not hold. We observe that given $\mathcal{U}', k', \{b_1, \ldots, b_r\}$, the number of valid combinations is equal to the number of valid combinations with $x \in \mathcal{U}'$ plus the number of valid combinations without x. We can define the number of valid combinations with x as the number of combinations of universe $\mathcal{U}' \setminus \{x\}$, spots $k' - 1$, and $\{b_i \mid x \notin b_i\}$. Likewise the number of valid combinations where x is not included is the number of combinations of universe $\mathcal{U}' \setminus \{x\}$, spots k', and $\{b_1 \setminus \{x\}, \ldots, b_r \setminus \{x\}\}$. By reducing $\mathcal{U}_2$, k, and the lists to a summation of Eq. (14) with different arguments, we find an exact expression of the number of X_2 for which $y \in \hat{X}_2$ holds.

6.3 Results

To evaluate our attack, we implement it in Rust.[3] We use constants hash seeds for each experiment and search an attack input set X_1 and target t once. Afterward, we determine the advantage by uniformly sampling $X_2 \subset \mathcal{U}_2$ 1 million times with $t \in X_2$ and 1 million times with $t \notin X_2$. We limit the time for searching an attack input set to two hours. The results are shown in Table 2. We indicate executions of our attack that did not finish computing with '-'.

7 Mitigations

Our theoretical analysis and practical attack require protocol designers to choose Bloom filter parameters that can easily be an order of magnitude larger, as shown in Table 1. This may be acceptable in protocols that use oblivious transfers [14], but this may significantly decrease the efficiency of protocols based on homomorphic encryption [11, 12, 10]. Fortunately, one may still design Bloom filter-based protocols that prevent these attacks in other ways, without having to increase the size of the Bloom filter significantly. In this section, we provide suggestions as to such mitigations.

[3]The source code is available at: `https://github.com/seratym/Bloom-filter-PSI-attack`

Table 2: Overview of the results of our practical attack applied to different false positive rates p and set sizes k. We evaluate the attack for $|\mathcal{U}_2| = 2k, 3k, 4k$. The attack succeeds with non-negligible probability, even for $p = 2^{-20}$.

Setting		Parameters		Results		
p	k	h	m_{opt}	$\text{Adv}^{\Pi}_{\text{memb}}(\mathcal{A}, 2k)$	$\text{Adv}^{\Pi}_{\text{memb}}(\mathcal{A}, 3k)$	$\text{Adv}^{\Pi}_{\text{memb}}(\mathcal{A}, 4k)$
2^{-5}	256	5	1,852	1.00	1.00	1.00
2^{-10}	256	10	3,702	1.00	0.82	0.61
	4,096	10	59,102	1.00	-	-
	65,536	10	945,493	1.00	-	-
2^{-20}	256	20	7,403	0.15	0.01	$2^{-8.9}$
	4,096	20	118,202	0.98	-	-
	65,536	20	1,890,985	0.68	-	-
2^{-30}	256	30	11,103	0.03	2^{-15}	-
	4,096	30	177,302	0.02	-	-
	65,536	30	2,836,477	2^{-10}	-	-

7.1 Replacing hash functions with OPRFs

In all protocols discussed in Section 4, the parties have access to the hash functions of the Bloom filter. However, this also allows corrupted parties to identify elements they can exploit in an attack, as shown in Section 6. Instead, one could replace the hash functions with oblivious pseudo-random functions (OPRFs) [6] with a secret seed. This would limit the parties in the number of queries they can make to the hash functions. Specifically, we can limit each party to k calls to each OPRF, which may be significantly smaller than the number of calls $|\mathcal{U}| - k$ that we use in our attack.

Designing a protocol around OPRFs requires answering the question of which parties can know the secret seed. We suggest to assign t parties who choose their own secret seed. Each party then engages in an OPRF protocol with each of these t parties to generate t pseudo-random values, from which one can derive a single pseudo-random value for which one can only find the preimage if the t parties collude. This approach would add two rounds to the total protocol's execution because each party must first run kt parallel OPRF evaluations before they can construct their Bloom filter.

7.2 Replacing hash functions with PBKDFs

Another approach is to replace the hash functions with extremely slow hash functions, such as password-based key derivation functions [7]. We note that this approach does not prevent polynomial-time attackers because the evaluation of the hash functions only increases with a polynomial factor. However, in practice, it would be infeasible (or at least extremely expensive) for an attacker to make $|\mathcal{U}| - k$ queries to the PBKDF.

While this approach offers a simple patch to existing Bloom filter-based PSI protocols, it is in stark contrast with the common choice of choosing extremely fast statistical hash functions without cryptographic guarantees. As a result, the protocol is also significantly slower to execute for honest parties. Depending on the number of elements each party encodes in their Bloom filter, choosing larger Bloom filter parameters may be cheaper.

Next to that, it may be hard to parameterize the PBKDF because one would have to estimate the concrete cost of the PBKDF and the computational abilities of an attacker. One solution may be to balance the cost of the hash function evaluation with larger Bloom filter parameters to find a set of parameters that punish attackers without significantly slowing down the protocol for honest parties.

7.3 Authorized private set intersections

If the parties running a Bloom filter-based PSI protocol trust a semi-honest third party, this party may authorize their sets, preventing the elements in these sets from being selected maliciously. An example of such a protocol is by Kerschbaum [23], who designed an authorized PSI protocol based on Bloom filters, in which a judge prevents the client from picking elements that are likely to cause a false positive in the Bloom filter of the server. To prevent a client from doing so, the elements of the server X_2 are encrypted before they are added to the Bloom filter. This encryption ensures that the client does not have prior knowledge of the server's Bloom filter. In order to perform the intersection, however, the client's elements should be encrypted with the same key.

In practice, the judge takes the elements of the client $\mathcal{P}_1$ and raises them to some secret power e. These elements are then stored in a Bloom filter by the judge. The Bloom filter is encrypted using a public key and signed, after which the encryption and signature are returned to the client. The client forwards the Bloom filter and the signed Bloom filter to the server, which then can verify that the judge signed this Bloom filter. Only the non-signed filter is used for the remainder of the protocol.

Applying authorization to the sets may not be possible in practice because the parties may not have a semi-honest third party they trust. Additionally, a judge that inspects (a subset of) the private input sets also learns private elements from the honest parties' sets. In any case, it adds at least two rounds to the execution of the protocol, as the clients have to submit their sets, and the judge has to approve them.

8 Conclusion

In this work, we propose both theoretical and practical attacks against Bloom filter-based private set intersection protocols. We show that secure parameters must be an order of magnitude larger than parameters where the false positive probability is not negligible. As a result, Bloom filter-based PSI cannot use the approximation provided by Bloom filters to speed up the protocol. Alternatively, one might consider consider replacing the hash functions with OPRFs or PBKDFs, but both approaches cause the protocol to become slower.

With these results, we are not aware of any approximate (M)PSI protocols that outperform exact (M)PSI protocols. As such, an open question is whether there are efficient alternatives for Bloom filter-based approximate (M)PSI protocols. Other future work may look at the following questions:

- Is it possible to extend our attack to protocols like that by Zhu et al. [24] and those based on garbled Bloom filters that go beyond regular Bloom filters?
- How are other deterministic approximate data structures such as approximate membership query filters affected by these results?

On a positive note, we notice that Bloom filters still offer useful characteristics for designing (M)PSI protocols, even when their parameters must be large. Specifically, it is still convenient to compute a Bloom filter representing the intersection. For example, the work by Ben Efraim et al. [14] uses parameters that are significantly larger than the parameters suggested in our work (our work considers the augmented semi-honest model while theirs considers the malicious model), but it is still concretely efficient.

References

[1] C. Dong, L. Chen, and Z. Wen, "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.

[2] J. Vos, M. Conti, and Z. Erkin, "Sok: Collusion-resistant multi-party private set intersections in the semi-honest model," *IACR Cryptol. ePrint Arch.*, p. 1777, 2023.

[3] P. Rindal and M. Rosulek, "Improved private set intersection against malicious adversaries," in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I* (J. Coron and J. B. Nielsen, eds.), vol. 10210 of *Lecture Notes in Computer Science*, pp. 235–259, 2017.

[4] K. Liu, X. Li, and T. Takagi, "Review the cuckoo hash-based unbalanced private set union: Leakage, fix, and optimization," in *Computer Security - ESORICS 2024 - 29th European Symposium on Research in Computer Security, Bydgoszcz, Poland, September 16-20, 2024, Proceedings, Part II* (J. García-Alfaro, R. Kozik, M. Choras, and S. K. Katsikas, eds.), vol. 14983 of *Lecture Notes in Computer Science*, pp. 331–352, Springer, 2024.

[5] C. Hazay and Y. Lindell, "A note on the relation between the definitions of security for semi-honest and malicious adversaries," *IACR Cryptol. ePrint Arch.*, p. 551, 2010.

[6] S. Casacuberta, J. Hesse, and A. Lehmann, "Sok: Oblivious pseudorandom functions," in *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*, pp. 625–646, IEEE, 2022.

[7] B. Kaliski, "Rfc2898: Pkcs# 5: Password-based cryptography specification version 2.0," 2000.

[8] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[9] A. Goel and P. Gupta, "Small subset queries and bloom filters using ternary associative memories, with applications," in *SIGMETRICS 2010, Proceedings of the 2010 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, New York, USA, 14-18 June 2010* (V. Misra, P. Barford, and M. S. Squillante, eds.), pp. 143–154, ACM, 2010.

[10] S. K. Debnath, P. Stanica, N. Kundu, and T. Choudhury, "Secure and efficient multi-party private set intersection cardinality," *Adv. Math. Commun.*, vol. 15, no. 2, pp. 365–386, 2021.

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

[12] 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.

[13] A. Miyaji, K. Nakasho, and S. Nishida, "Privacy-preserving integration of medical data - A practical multiparty private set intersection," *J. Medical Syst.*, vol. 41, no. 3, pp. 37:1–37:10, 2017.

[14] A. Ben-Efraim, O. Nissenbaum, E. Omri, and A. Paskin-Cherniavsky, "Psimple: Practical multiparty maliciously-secure private set intersection," in *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022* (Y. Suga, K. Sakurai, X. Ding, and K. Sako, eds.), pp. 1098–1112, ACM, 2022.

[15] S. K. Debnath and R. Dutta, "Secure and efficient private set intersection cardinality using bloom filter," in *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings* (J. López and C. J. Mitchell, eds.), vol. 9290 of *Lecture Notes in Computer Science*, pp. 209–226, Springer, 2015.

[16] A. Davidson and C. Cid, "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 22*, pp. 261–278, Springer, 2017.

[17] O. Ruan, C. Yan, J. Zhou, and C. Ai, "A practical multiparty private set intersection protocol based on bloom filters for unbalanced scenarios," *Applied Sciences*, vol. 13, no. 24, 2023.

[18] O. Ruan and C. Ai, "An efficient multi-party private set intersection protocols based on bloom filter," in *Second International Conference on Algorithms, Microchips, and Network Applications (AMNA 2023)* (K. Subramaniam and A. P. Muthuramalingam, eds.), vol. 12635, p. 1263518, International Society for Optics and Photonics, SPIE, 2023.

[19] S. Qiu, Z. Zhang, Y. Liu, H. Yan, and Y. Cheng, "Se-psi: Fog/cloud server-aided enhanced secure and effective private set intersection on scalable datasets with bloom filter," *Mathematical Biosciences and Engineering*, vol. 19, no. 2, pp. 1861–1876, 2022.

[20] A. Miyaji and S. Nishida, "A scalable multiparty private set intersection," in *Network and System Security* (M. Qiu, S. Xu, M. Yung, and H. Zhang, eds.), (Cham), pp. 376–385, Springer International Publishing, 2015.

[21] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pp. 136–145, IEEE Computer Society, 2001.

[22] B. Liu, O. Ruan, R. Shi, and M. Zhang, "Quantum private set intersection cardinality based on bloom filter," *Scientific Reports*, vol. 11, no. 1, p. 17332, 2021.

[23] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, (New York, NY, USA), p. 85–86, Association for Computing Machinery, 2012.

[24] H. Zhu, M. Chen, M. Sun, X. Liao, and L. Hu, "Outsourcing set intersection computation based on bloom filter for privacy preservation in multimedia processing," *Security and Communication Networks*, vol. 2018, no. 1, p. 5841967, 2018.

A Conditions on the false positive probability

Lemma 7. $\Pr[FPs = 0] \geq \frac{1}{2}$ *if and only if* $p < \frac{1}{|\mathcal{U}|-k}$.

Proof. First, consider that the distribution of the number of false positives is binomial $\Pr[\text{FPs}] \sim \mathcal{B}(|\mathcal{U}| - k, p)$, because $|\mathcal{U}/X_2| = |\mathcal{U}| - k$ and p is the false positive probability. The median is given by $\lfloor (|\mathcal{U}| - k)p \rfloor$ or $\lceil (|\mathcal{U}| - k)p \rceil$.

Our lemma holds when the median M is at or below FPs = 0. Since $\lfloor (|\mathcal{U}| - k)p \rfloor \leq \lceil (|\mathcal{U}| - k)p \rceil$, we get that:

$$M \leq 0 \tag{15}$$
$$\lfloor (|\mathcal{U}| - k)p \rfloor \leq 0 \tag{16}$$
$$(|\mathcal{U}| - k)p < 1 \tag{17}$$
$$p < \frac{1}{|\mathcal{U}| - k}$$ □

Lemma 8. $\Pr[FPs = k] \geq \frac{1}{2}$ *if and only if* $p > \frac{k-1}{|\mathcal{U}|-k}$.

Proof. Like in Lemma 7, the distribution of the number of false positives is binomial $\Pr[\text{FPs}] \sim \mathcal{B}(|\mathcal{U}| - k, p)$, and the median is given by $\lfloor (|\mathcal{U}| - k)p \rfloor$ or $\lceil (|\mathcal{U}| - k)p \rceil$.

Our lemma holds when the median M is at or above FPs = k. Since $\lfloor (|\mathcal{U}| - k)p \rfloor \leq \lceil (|\mathcal{U}| - k)p \rceil$, we get that:

$$M \geq k \tag{18}$$
$$\lceil (|\mathcal{U}| - k)p \rceil \geq k \tag{19}$$
$$(|\mathcal{U}| - k)p > k - 1 \tag{20}$$
$$p > \frac{k-1}{|\mathcal{U}| - k}$$ □

B Additional lemmas for proving upper bounds

Lemma 9. *Given* $p > \frac{k-1}{|\mathcal{U}|-k}$ *and* $0 \leq \varepsilon_{fp} \leq \frac{1}{2}$, *the following is monotonically increasing with* ε_{fp}*:*

$$\sum_{i=0}^{k-1} \Pr[FPs = i] \varepsilon_{fp}^{i} (1 - \varepsilon_{fp})^{k-i}$$

Proof. To show that the function is monotonically increasing on the interval $[0, \frac{1}{2}]$, we use the derivative.

$$\begin{aligned}
&\frac{\delta}{\delta\varepsilon_{\text{fp}}} \sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^{i}(1-\varepsilon_{\text{fp}})^{k-i} \\
&= \sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\left(i\varepsilon_{\text{fp}}^{i-1}(1-\varepsilon_{\text{fp}})^{k-i} + \varepsilon_{\text{fp}}^{i}(i-k)(1-\varepsilon_{\text{fp}})^{k-i-1}\right) \\
&= \sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^{i-1}(1-\varepsilon_{\text{fp}})^{k-i-1}\left(i(1-\varepsilon_{\text{fp}}) + \varepsilon_{\text{fp}}(i-k)\right) \\
&= \sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^{i-1}(1-\varepsilon_{\text{fp}})^{k-i-1}(i-k\varepsilon_{\text{fp}})
\end{aligned}$$

Note that the derivative for $\varepsilon_{\text{fp}} = 0$ is not defined because 0^{i-1} is not defined for $i = 0$. We thus take the limit:

$$\begin{aligned}
&\lim_{\varepsilon_{\text{fp}}\to 0^+} \sum_{i=0}^{k-1} \Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^{i-1}(1-\varepsilon_{\text{fp}})^{k-i-1}(i-k\varepsilon_{\text{fp}}) \\
&= \sum_{i=0}^{k-1} \Pr[\text{FPs} = i](\lim_{\varepsilon_{\text{fp}}\to 0^+} \varepsilon_{\text{fp}}^{i-1})(\lim_{\varepsilon_{\text{fp}}\to 0^+} (1-\varepsilon_{\text{fp}})^{k-i-1})(\lim_{\varepsilon_{\text{fp}}\to 0^+} (i-k\varepsilon_{\text{fp}})) \\
&= \sum_{i=0}^{k-1} \Pr[\text{FPs} = i] \cdot \infty \cdot 1 \cdot i = \infty.
\end{aligned}$$

The derivative is thus positive for $\varepsilon_{\text{fp}} = 0$, and it might be clear that for small ε_{fp}, the value will also be positive. For $0 < \varepsilon_{\text{fp}} \leq \frac{1}{2}$, we must show that the derivative is also positive. To show that the derivative is always positive at the given interval, we will find the minimum and show that the minimum is indeed positive. We claim that the derivative is monotonically decreasing with ε_{fp}, so we will evaluate it on the edge of the domain: $\varepsilon_{\text{fp}} = \frac{1}{2}$.

We do not give a detailed proof of the claim that the sum reaches its minimum value when most negative terms occur. However, we will give the intuition behind this claim. The intuition is that when i grows, the sum terms get larger (or more negative). The value of the terms does not depend as much on the value of ε_{fp}, and thus, the minimum is reached when most terms are negative.

To see that ε_{fp} does not contribute much to the value of the terms, we use the following intuition. The false positive rate p expects more than k false positives, thus $\Pr[\text{FPs} = i] \leq \Pr[\text{FPs} = i + 1]$ holds for all $i < k$. The factor $\varepsilon_{\text{fp}}^{i-1}(1-\varepsilon_{\text{fp}})^{k-i-1}$ is negligible compared to $\Pr[\text{FPs} = i]$.

The sum thus reaches the minimum value when most negative terms occur. The factor $\Pr[\text{FPs} = i]\varepsilon_{\text{fp}}^{i-1}(1-\varepsilon_{\text{fp}})^{k-i-1}$ is always positive. The factor $(i - k\varepsilon_{\text{fp}})$, however, is negative for $i \leq k\varepsilon_{\text{fp}}$. This factor reaches its largest value at $\varepsilon_{\text{fp}} = \frac{1}{2}$. Filling this in, we get that the sum terms are negative for all $i \leq \frac{k}{2}$. For this value, the sum has the most negative terms

The least outcome the sum can have is thus for $\varepsilon_{\text{fp}} = \frac{1}{2}$. However, even in this case, the

sum has a positive outcome, as can be seen in the following calculation

$$\begin{aligned}
&\sum_{i=0}^{k-1} \Pr[\text{FPs} = i] \left(\frac{1}{2}\right)^{i-1} \left(1 - \frac{1}{2}\right)^{k-i-1} (i - k\varepsilon_{\text{fp}}) \\
=&\sum_{i=0}^{k-1} \Pr[\text{FPs} = i] \left(\frac{1}{2}\right)^{k-2} (i - k\varepsilon_{\text{fp}}) \\
=&\left(\frac{1}{2}\right)^{k-2} \sum_{i=0}^{k-1} \Pr[\text{FPs} = i](i - k\varepsilon_{\text{fp}}).
\end{aligned}$$

For $i \leq \frac{k}{2}$, the terms are negative, and the other terms are positive. The positive terms contribute the most to the sum and thus the sum is positive. In conclusion, the derivative is always positive on the interval $[0, \frac{1}{2}]$ and thus is the original function monotonically increasing on the interval $[0, \frac{1}{2}]$ as desired. □