This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2721360, IEEE Transactions on Information Forensics and Security

Approximating Private Set Union/Intersection Cardinality with Logarithmic Complexity

Changyu Dong, Grigorios Loukides

Abstract—The computation of private set union/intersection cardinality (PSU-CA/PSI-CA) is one of the most intensively studied problems in Privacy Preserving Data Mining (PPDM). However, existing protocols are computationally too expensive to be employed in real-world PPDM applications. In response, we propose efficient approximate protocols, whose accuracy can be tuned according to application requirements. We first propose a two-party PSU-CA protocol based on Flajolet-Martin sketches. The protocol has logarithmic computational/communication complexity and relies mostly on symmetric key operations. Thus, it is much more efficient and scalable than existing protocols. In addition, our protocol can hide its output. This feature is necessary in PPDM applications, since the union cardinality is often an intermediate result that must not be disclosed. We then propose a two-party PSI-CA protocol, which is derived from the PSU-CA protocol with virtually no cost. Both our two-party protocols can be easily extended to the multiparty setting. We also design an efficient masking scheme for $\binom{1}{n}$-OT. The scheme is used in optimizing the two-party protocols and is of independent interest, since it can speed up $\binom{1}{n}$-OT significantly when n is large. Last, we show through experiments the effectiveness and efficiency of our protocols.

1 Introduction

We are in an era where data becomes increasingly important. On one hand, data drives scientific research, business analytics, and government decision making. Advanced technologies for discovering interesting knowledge from large amounts of data have become an indispensable part of nearly everything. On the other hand, data privacy becomes of paramount importance as evidenced by the increasingly tighter legal obligation imposed by legislation (e.g. HIPAA, COPPA, and GLB in the US, European Union Data Protection Directive, and more specific national privacy regulations). Driven by both, recently we have seen a significant advancement in privacy preserving data mining (PPDM). In many scenarios, mining the union of data held by two or more parties could deliver a clear benefit. For example, online retailers want to find correlations between products bought by their common customers to boost sales, policy makers want to link healthcare data held by public and private healthcare providers to develop better public policies, and geneticists want to associate mutations in human genomes with diagnoses in medical records to identify genetic causes of cancers. In all these scenarios, privacy concerns and/or privacy regulations prohibit the sharing of data between parties. Thus, the application of conventional data mining methods is not possible, and PPDM methods are needed to perform data mining in a distributed fashion, without disclosing or pooling the data of any party.

This paper investigates a long-established problem in PPDM: how to securely compute the cardinality of the union or the intersection of some private sets (PSU-CA/PSI-CA). More formally, consider two parties each holding a private set S_i. The PSU-CA problem is to securely compute the union cardinality $|S_1 \cup S_2|$, and the PSI-CA problem is to securely compute the intersection cardinality $|S_1 \cap S_2|$. At the end, parties should obtain the cardinality of the union/intersection but nothing else about other parties' sets. PSU-CA and PSI-CA can be defined similarly in the multiparty (> 2) case. PSU-CA and PSI-CA are closely related: often solving one problem leads to an easy solution to the other problem. Thus they can be treated as one problem. The problem is of practical importance because protocols for solving the problem are important building blocks in PPDM. For example, PSU-CA/PSI-CA protocols have been used as subroutines in privacy preserving association rule mining [1], privacy preserving classification (e.g. decision trees [2] and Support Vector Machine [3]), and privacy preserving mining of social network data [4].

The PSU-CA/PSI-CA problem can be solved by using generic secure computation protocols (e.g. garbled circuits [5], GMW [6]). However those protocols have high computational and communication costs and are difficult to scale to large sets that are required in real-world data mining applications. Thus several custom protocols have been proposed to solve the PSU-CA/PSI-CA problem. Many of them aim to compute the exact cardinality [7], [8], [9], [1], [10], [11], [12]. Yet, their high computational cost makes their application to PPDM infeasible. For example, let us consider a scenario in which two social network providers need to find out the total number of friends of each user that has registered in both networks. For each such user, the two providers can locally construct a set that contains all friends of the user in their own social network. Then the two providers can run a PSU-CA protocol to find the union cardinality of the two sets, which is equal to the total number of friends of the user. The input to the protocol can be large because a user may have thousands of friends. Even with the most efficient protocol to date, finding the union cardinality would need tens or even hundreds of seconds. Furthermore, there are millions of users registered in both social networks. Thus, the protocol needs to run millions of times, and the task may take months or even years. This is clearly impractical.

- *Changyu Dong is with the School of Computing Science, Newcastle University, Newcastle Upon Tyne, UK. Email: changyu.dong@newcastle.ac.uk.*
- *Grigorios Loukides is with the Department of Informatics, King's College London, London, UK. Email: grigorios.loukides@kcl.ac.uk.*
- *This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material includes the appendices of this paper. Contact changyu.dong@newcastle.ac.uk for further questions about this work.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2721360, IEEE Transactions on Information Forensics and Security

Recently, there has been much interest in approximate PSU-CA and PSI-CA protocols [13], [14], [15]. It is well-known that in data mining, a close approximation is often as good as the exact result [16]. Approximation is widely used in data mining to handle extremely large datasets when the exact answer is hard to compute [17]. With a bounded loss of accuracy, it is possible to make the whole data mining process much more efficient. The same principle applies to PPDM. That is, approximation can simplify protocol design and reduce the size of input to the protocol which both could help making the protocol more efficient. Unfortunately, the current approximate PSU-CA and PSI-CA protocols are either insecure ([13], [14]) or not very efficient ([15]), as we discuss in Section 2.

A more subtle problem of most of existing protocols ([7], [8], [1], [11], [14], [9], [10], [13], [15], [12]) is that they output the intersection/union cardinality directly to one or all parties. This is perfectly fine as per the definition of the problems. However, the cardinalities may not be the end result the parties want. Especially in PPDM, these protocols are often used as subroutines and the cardinalities are only intermediate results. Outputting cardinalities increases the leakage of the PPDM protocol: ideally intermediate results should be hidden and the only output should be the data mining result. For instance, in the social network example above, the two providers want to find out the k most influential users. The union cardinality that represents the number of friends is only a feature to be used in finding such users. It is too excessive if the total number of friends of every user has to be revealed in the mining process. Often what we need is, rather than outputting the cardinalities directly, to keep the cardinalities secret and run a secure sorting protocol on the secret cardinalities to output the users who have the most friends, or run a secure comparison protocol to output the users who have more friends than a pre-defined threshold. Avoiding this leakage is not easy, due to the design of the protocols. To find the cardinalities, those protocols all depend on a party (or all parties) in the protocol to count either the number of matches among some ciphertexts [7], [8], [1], [11], [14] or the occurrences of a specific value (e.g. 0) [9], [10], [13], [15], [12]. Then the party that does the counting inevitably learns the cardinalities.

Contributions Our main contribution is a secure two-party approximate PSU-CA protocol. The protocol is based on the Flajolet-Martin (FM) sketch [18], which is a space efficient data structure for cardinality estimation. The parties use their FM sketches as inputs to the protocol and compute the union cardinality. At the end each party gets a share that can be used to reconstruct the cardinality, but none of them learns the cardinality directly. The protocol has the following important features:

- *Highly efficient and accurate*. The protocol is efficient for two reasons. First, unlike all existing protocols, our protocol has computational and communication complexities logarithmic in $\mathcal{N}$, the maximum possible cardinality of the private sets[1]. This is achieved by using FM sketches. Second, our protocol is mainly based on symmetric key operations, while most existing protocols (except generic protocols) are based on much slower public key operations. The accuracy of the estimation of our protocol is adjustable with a public parameter m. In practice, larger m values improve the estimation accuracy. More precisely, by setting m to a large enough value, the protocol can guarantee the relative error $(|\widetilde{N}-N|)/N$ is at most ϵ with probability at least $1-\delta$ for any arbitrary $\epsilon, \delta \in (0,1)$, where N is the true cardinality and $\widetilde{N}$ is the estimated cardinality. The minimum allowable value for m depends on the parameters δ and ϵ, as it will be explained in Section 4.1. We evaluated the protocol based on our prototype implementation and found that, to compute the union cardinality of two sets with size up to 1 million, it only needed 2.97 seconds when $\epsilon = 1\%$ and $\delta = 0.001$. In contrast, the state-of-the-art two-party approximate protocol [15] took 488.48 seconds to do the same computation with the same accuracy.
- *Eliminates unwanted information leakage*. We design the protocol such that by default, no party learns the cardinality at the end of the protocol. The result is split into secret shares and each party holds one share. The shares leak no information about the result. If the cardinality is the final result, the parties can reconstruct it from the shares easily and reveal the cardinality. Or if the protocols are used as subroutines in a larger PPDM protocol, the shares can be fed into the next step of the PPDM protocol without leaking the cardinality. We decided to output the result as secret shares for three reasons: (1) it is simple and incurs virtually no cost to produce the shares; (2) it allows local computation on the shares (e.g. summing the output of Protocol 1); (3) the shares can be easily converted into other encrypted forms using existing techniques (e.g. to Boolean shares or garbled bit strings [19], to ciphertexts of homomorphic encryption schemes [20], and to ciphertexts of other encryption schemes [21]). The last reason is important because the subsequent protocol that will use the cardinality is dependent on the application and may require input other than secret shares. Being able to convert shares into other encrypted form makes our protocol flexible in such a situation.
- *Extensible*. We extend the PSU-CA protocol into a PSI-CA protocol with virtually no cost using the inclusion-exclusion principle. This is similar to the approach in [11], where the authors extend a PSI-CA protocol into a PSU-CA protocol using the same principle. We also extend the two-party PSU-CA and PSI-CA protocols to the multiparty setting. The multiparty extensions retain the good properties and can be implemented easily using generic secret-sharing based multiparty secure computation frameworks.

A second contribution is the design of a fixed-key masking scheme for $\binom{1}{n}$-OT. In the past, the computation cost of $\binom{1}{n}$-OT was dominated by the $\log(n)$ invocations of $\binom{1}{2}$-OT. Now since $\binom{1}{2}$-OT can be obtained by OT extension, the cost of masking the n strings becomes dominating. The idea of the fixed-key masking scheme is similar to previous work in fixed-key garbling schemes [22]. That is, we use a block cipher with a fixed key as a random permutation, and apply the fixed-key block cipher to mask the strings. The benefit of the fixed-key masking is significant: it reduces the number of invocations of the encryption function from $\log(n)$ per string to 1 per string; it can take advantage of the AES-NI set that is widely available on recent X86 CPUs; and using fixed-key avoids costly key scheduling so the encryption can be fully pipelined. The efficiency of masking is improved by more than one order of magnitude compared to the previous masking scheme.

[1] In this paper, we use $\mathcal{N}$, N and $\widetilde{N}$ to denote the maximum possible cardinality of the private sets, the actual union cardinality, and the approximate union cardinality. The complexity is in $\mathcal{N}$ because the protocol must not leak the cardinality of the parties' input sets. This is usually achieved by padding the sets to a fixed size with random dummy elements or using fixed-size sketches.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2721360, IEEE Transactions on Information Forensics and Security

2 Related Work

Since the groundbreaking work of [23], [24], there has been extensive research in PPDM. Much research focuses on the development of a few primitive protocols. This is because there are many data mining techniques, and it is infeasible or not cost-effective to develop solutions for individual ones. One observation is that data mining techniques often perform similar computations at various stages. Therefore a more viable strategy [7] is to build a "toolkit" of primitive protocols that can be assembled to solve specific real-world problems. PSI-CA/PSU-CA is one of the primitive protocols identified in [7] and is widely used in PPDM.

Exact Protocols There are several exact PSI-CA/PSU-CA protocols. However they all require at least $O(\mathcal{N})$ public key operations, where $\mathcal{N}$ is the maximum possible cardinality of the private sets. Thus they are less efficient than our protocol.

Both [7] and [8] proposed a PSI-CA protocol based on commutative encryption. The ideas of the two protocols are very similar. The main difference is that [7] was presented in the multiparty setting and [8] in the two-party setting. Both protocols require $O(\mathcal{N})$ public key operations. In [9], a PSI-CA protocol was proposed based on oblivious polynomial evaluation. The computation requires $O(\mathcal{N} \log\log \mathcal{N})$ public key operations. In [1], a multiparty PSI-CA protocol was proposed based on commutative one-way hash functions that can be constructed from public key encryption schemes such as Pohlig-Hellman. Each party needs to hash $\tau \cdot \mathcal{N}$ times where τ is the number of parties. In [10], a multiparty PSI-CA protocol was proposed based on oblivious polynomial evaluation. The computation requires $O(\mathcal{N}^2)$ public key operations. In [11] a PSI-CA protocol based on an ElGamal like encryption was proposed. The protocol requires $O(\mathcal{N})$ public key operations and can be trivially extended to a PSU-CA protocol. In [12], the authors proposed PSI-CA/PSU-CA protocols based on Bloom filter and homomorphic encryption. These protocols also require $O(\mathcal{N})$ public key operations.

Approximate Protocols In data mining, approximation is widely used when mining extremely large datasets. Approximate PSI-CA/PSU-CA protocols were proposed in the hope that they can be more efficient. However, unlike our protocol, they are either insecure or still have complexity $O(\mathcal{N})$.

In [13] a multiparty PSI-CA protocol was designed but it is not secure. Adversaries can easily guess any other party's set elements, as explained in [15]. In [14], a two-party PSI-CA protocol was proposed. The protocol uses [11] as a subprotocol and is based on Minwise sketches [25]. The protocol estimates Jaccard index from Minwise sketches and then approximates the intersection cardinality from Jaccard index. However, this protocol is not secure because party 2 in the protocol leaks the cardinality of its private set to party 1. Our multiparty PSI-CA protocol also estimates intersection cardinality from Jaccard index. The differences are: (1) We use the Min-Max sketch [26], which is more efficient than the Minwise sketch. The use of Min-Max sketches also reduces the offline computation by a factor of 2 compared to Minwise sketches [26]. (2) The parties do not need to reveal the cardinality of their own sets. In [15] the authors proposed PSU-CA/PSI-CA protocols that use Bloom filters to estimate the cardinalities. However, the protocols in [15] need $O(\mathcal{N})$ time. The reason is that Bloom filters were originally designed for set membership queries, and thus they have to encode much more information than needed to estimate cardinality.

From PSI to PSI-CA A related line of work is on Private Set Intersection (PSI) which requires computing the intersection of private sets (see e.g. [9], [10], [27], [28], [29], [30]). Intersection cardinality can be obtained from PSI output, but PSI also reveals the elements in the intersection. Thus PSI protocols cannot be used as a replacement for PSI-CA in applications where only the cardinality of a set must be revealed.

It is possible in some cases to extend PSI to PSI-CA. For example, the PSI-CA protocols in [8], [9], [10], [11], [12] we mentioned earlier are all extended from PSI protocols. However, their underlying PSI protocols are public key based, which makes the above protocols inefficient. There are PSI protocols that require mostly symmetric key operations ([31], [27], [28], [29]). Specifically [31], [28], [29], [30] propose Boolean circuits for the PSI function that can be evaluated securely using generic secure computation protocols. It is always possible to extend the PSI Boolean circuits to support PSI-CA. The extended circuits would have similar complexity as the PSI circuits, which is at least $O(\mathcal{N})$. On the other hand, the garbled Bloom filter [27] and OT based [28], [29], [30] PSI protocols cannot be easily extended to PSI-CA because one party always learns the intersection due to the way the intersection is obtained. In the garbled Bloom filter based PSI protocol, one party receives a garbled Bloom filter that encodes the intersection and allows set membership query. The party then queries the garbled Bloom filter using every element in its own set. If the element is in the intersection, the query returns a fixed string, otherwise it returns a random string. Similarly in the OT-based PSI protocols, one party first gets some random-looking strings for each element in its set by running OT. Then the other party sends a set of strings that are mapped from its set elements. Next, the first party checks, for each element, whether there is a string associated with it in the set. If so this element is in the intersection. The only obvious solution to extend [27], [28], [29], [30] to support PSI-CA is to interactively blind and randomly permute the set (of the party who obtains the intersection) at the start of the protocol, so that at the end the party can query to check whether a blinded element is in the intersection, without knowing which element is being queried. This however seems to require at least $O(\mathcal{N})$ public key operations and cannot hide the cardinality from the party.

3 Preliminaries

3.1 Notation

For a set X, we denote by $x \overset{R}{\leftarrow} X$ the process of choosing an element x of X uniformly at random. For a vector V, we denote by $V[i]$ the ith element in the vector. The index of all vectors in the paper starts from 0. For an integer a, we denote by $[\![a]\!]_i$ the secret share of a held by party i. In a loop or a multi-round protocol, we use superscription in angle brackets to differentiate variables with the same name in different iterations, e.g. $r^{\langle i \rangle}$ means variable r in the ith iteration. All logarithms in this paper are base 2.

3.2 Oblivious Transfer (OT)

Oblivious transfer [32], [33] is a protocol between a sender and a receiver. The most basic type of OT is 1-out-of-2 OT, which will be denoted as $\binom{1}{2}$-OT. In $\binom{1}{2}$-OT, the sender holds a pair of strings (x_0, x_1) and the receiver holds a bit b. The goal is for the receiver to receive x_b such that the receiver learns nothing about the other string and the server learns nothing about b. The idea can be extended naturally to 1-out-of-n OT, denoted as $\binom{1}{n}$-OT, in which a sender holds a vector of n strings $(x_0, \ldots, x_{n-1})$ and the receiver holds an index $0 \leq I \leq n-1$. At the end the receiver only learns x_I and the server learns nothing about I. A $\binom{1}{n}$-OT protocol can be constructed by invoking a $\binom{1}{2}$-OT protocol $\log(n)$ times [34] as follows:

- The sender holds a vector of messages $P = (x_0, ..., x_{n-1})$ and the receiver holds an index $0 \leq I \leq n-1$.
- The sender chooses l pairs of uniformly random keys $(k_{0,0}, k_{0,1}), \ldots, (k_{l-1,0}, k_{l-1,1})$ for a pseudorandom function F, where $l = \lceil \log(n) \rceil$. For the ith message in P, the sender masks the message $\widetilde{P}[i]{=}(\bigoplus_{j=0}^{l-1} F_{k_{j,b_j}}(i)){\oplus}P[i]$ where b_j is the jth bit in the binary representation of i and $\oplus$ is the bitwise XOR operation. The sender sends $\widetilde{P}$ to the receiver.
- Let $I_0 ... I_{l-1} \in \{0,1\}^l$ be the binary representation of I, then l $\binom{1}{2}$-OT are performed, where during the jth OT, the sender sends $(k_{j,0}, k_{j,1})$ and the receiver uses I_j to receive k_{j,I_j}.
- The receiver now has $k_{0,I_0}, ..., k_{l-1,I_{l-1}}$, then can unmask $P[I]{=}(\bigoplus_{j=0}^{l-1} F_{k_{j,I_j}}(I)){\oplus}\widetilde{P}[I]$.

OT protocols inevitably require public key operations, thus computing a large number of OTs is expensive. Fortunately, it has been shown by Beaver [35] that it is possible to obtain a large number oblivious transfers given only a small number of actual oblivious transfer calls. This is called OT extension. The first practical OT extension scheme was proposed by Ishai et. al. [36]. Recently more efficient OT extension schemes were proposed [37], [38], [39], [40]. In short, in those schemes only a small number (a few hundred) of $\binom{1}{2}$-OT ("base OTs") are required at the bootstrapping phase, then the subsequent $\binom{1}{2}$-OT can be obtained with the cost of just a few cheap symmetric key operations. Therefore those schemes can significantly improve the performance of protocols based on OT. Our two-party protocols rely heavily on OT. Thus they can benefit from OT extension schemes. To be clear, in the rest of paper when we write OT we mean OT obtained though an OT extension scheme.

3.3 Secret Sharing

Secret sharing is widely used in secure computation protocols. In general, in a (t, n)-secret sharing scheme, a dealer splits a secret s into n shares. The scheme is correct if s can be reconstructed efficiently with any subset of t or more shares. The scheme is secure if given any subset of less than t shares, the secret is unrecoverable and the shares give no information about the secret. Some secret sharing schemes have homomorphic properties, i.e. certain operations on the secret can be performed with the shares as input. For example, Shamir's secret sharing scheme [41] is additively homomorphic.

In our two-party protocols, we will use a simple additively homomorphic $(2, 2)$-secret sharing scheme. In this scheme, the secret and shares are integers in the additive group $\mathbb{Z}_q$ for some integer $q \geq 2$. Note q can be any integer and does not need to be a prime number. To share a secret s, choose a uniformly random r from $\mathbb{Z}_q$, and the two shares are $[\![s]\!]_1 = r$ and $[\![s]\!]_2 = s - r$. To reconstruct the secret, simply add the two shares together $s = [\![s]\!]_1 + [\![s]\!]_2$. The correctness of the scheme is easy to verify and the scheme is unconditionally secure if r is chosen uniformly at random. The homomorphic property is obvious: let $[\![a]\!]_1, [\![a]\!]_2$ be the two shares of a and $[\![b]\!]_1, [\![b]\!]_2$ be the two shares of b, then $[\![c]\!]_1 = [\![a]\!]_1 + [\![b]\!]_1$ and $[\![c]\!]_2 = [\![a]\!]_2 + [\![b]\!]_2$ are the two shares of c such that $c = a + b$.

3.4 Security Model

All protocols in this paper are secure in the semi-honest model [42]. In this model, adversaries are honest-but-curious, i.e. they will follow the protocol specification but try to get more information about the honest party's input. The semi-honest model is weaker than the malicious model, in which the adversaries can deviate from the protocol in arbitrary ways. However, designing protocols in the semi-honest model is still very meaningful as it captures many realistic scenarios. For example, when the parties' behaviors are monitored or audited. Also, protocols for the semi-honest setting are often the stepping-stones towards protocols with stronger security guarantees. There exist generic ways of obtaining full security against malicious adversaries from protocols for the semi-honest setting, e.g. by using zero-knowledge proofs. The formal definitions can be found in Appendix B.

4 Two-Party Protocols

In this section, we present the PSU-CA and PSI-CA protocols in the two-party setting. In this setting, there are two parties $\mathcal{P}_1$ and $\mathcal{P}_2$ each holding a private set (S_1 and S_2 respectively). Our focus is on computing the union cardinality because the intersection cardinality can be obtained trivially by applying the inclusion-exclusion principle: $|S_1 \cap S_2| = |S_1| + |S_2| - |S_1 \cup S_2|$. We will start by reviewing the FM sketches, then present protocols designed around this data structure, as well as a few optimizations.

4.1 Flajolet-Martin (FM) Sketches

We briefly review FM sketches. More details and analysis can be found in [18]. An FM sketch is a probabilistic counter of the number of distinct elements in a multiset[2]. The data structure is a w-bit binary vector. We will use F_S to denote an FM sketch built from a set S, and $F_S[i]$ $(0 \leq i \leq w-1)$ to denote the ith bit in F_S. An FM sketch comes with a hash function $h : \{0,1\}^* \to \{0,1\}^w$ that maps an input uniformly to w-bit output. A function $\rho : \{0,1\}^w \to [0, w]$ is defined that takes a w-bit string as input and returns the number of trailing zeroes in the string. Initially, all bits in F_S are set to 0. To count a multiset S, for each element $x \in S$, we hash x and set $F_S[\rho(h(x))] = 1$ if $\rho(h(x)) < w$. The number of distinct elements in S can be estimated using an estimator z that is the index of the first 0 bit in F_S, i.e. $F_S[z] = 0$ and $\forall 0 \leq i < z$, $F_S[i] = 1$. The expected value of z is close to $\log(\phi N)$, where $\phi = 0.77351$ is a correction factor and N is the number of distinct element in S. Given z, we can estimate N by $\widetilde{N} = \frac{2^z}{\phi}$. It is clear that the size of the sketch w must be larger than $\log(\phi N)$, otherwise we might not be able to obtain z. It was suggested in [18] that $w \geq \log(N) + 4$ should suffice.

The standard deviation of the estimator z is 1.12, which is too high. An estimate $\widetilde{N}$ using z will typically be one binary order of magnitude off n. To remedy this problem, we can use m sketches each with an independent hash function. We obtain m estimators $z^{\langle 0 \rangle}, ..., z^{\langle m-1 \rangle}$ and sum them $Z = z^{\langle 0 \rangle} + \ldots + z^{\langle m-1 \rangle}$. We can then use the average $\frac{Z}{m}$ to estimate n. The standard deviation of $\frac{Z}{m}$ is $\frac{1.12}{\sqrt{m}}$, which is much smaller.

Another problem of FM sketches is that they give bad estimates for small sets. This has been studied in [43] and the authors suggested a modified formula to correct the small set bias:

$$\widetilde{N} = \frac{2^{\frac{Z}{m}} - 2^{-\kappa \cdot \frac{Z}{m}}}{\phi} \tag{1}$$

[2] A set is treated as a special case of multiset (multiplicity = 1 for all elements), and all the following applies to sets as well.

where $\widetilde{N}$ is the cardinality estimate from m sketches, and $\kappa = 1.75$ is a correcting factor. Equation (1) gives very good estimates for both small sets and large sets [43].

In Theorem 1, we show that the relative error between the true and estimated cardinality does not exceed ϵ with probability at least $1 - \delta$, when m is sufficiently large. This implies that the accuracy of the estimation can be adjusted to the desired level, by choosing a suitable m. The proof of Theorem 1 is in Appendix A.

Theorem 1. *Let S_1, S_2 be two sets and $N = |S_1 \cup S_2|$. Let $\widetilde{N}$ be the estimate obtained from computed using Equation (1). For any $\epsilon, \delta \in (0, 1)$, it holds that:*

$$Pr[\frac{|\widetilde{N} - N|}{N} \leq \epsilon] \geq 1 - \delta \tag{2}$$

when $m \geq 2.5088 \cdot (\frac{\mathrm{erf}^{-1}(1-\delta)}{min(-\log(1-\epsilon),\log(1+\epsilon))})^2$, where erf^{-1} is the inverse error function[3].

An important property of FM sketches that we use in the design of our algorithms is that they can be merged. If we have two FM sketches F_{S_1} and F_{S_2} built with the same hash function, then bit-wisely ORing the two sketches produces a new FM sketch $F_{S_1 \cup S_2}$ that counts the union of the two sets S_1 and S_2. This process is lossless: $F_{S_1 \cup S_2}$ is exactly the same as the sketch built using the union from the scratch. This extends to the union of multiple sets easily. However, an FM sketch of set intersection cannot be obtained by combining sketches. This is because there are no known estimators for the intersection cardinality of two sets that can be applied to a combined FM sketch, which is derived from the two sketches by any bit-wise operation. In other words, the bit patterns that appear in the combined sketch cannot be used to derive an estimate of the set intersection cardinality. In fact, union is the only supported set operation that can be performed by combining FM sketches [44].

4.2 Secure Estimator Computation

To securely compute the cardinality of the union of two private sets using FM sketches, the first step is to securely compute the estimator of the union cardinality. The two parties each hold F_{S_1} and F_{S_2} that are FM sketches built from their private sets using the same hash function and same sketch size w. As we have seen in Section 4.1, the union sketch $F_{S_1 \cup S_2}$ can be then computed by bit-wisely ORing F_{S_1} and F_{S_2} in a secure way. However, how to securely extract the estimator from $F_{S_1 \cup S_2}$ is a non-trivial task.

4.2.1 Data Oblivious Algorithm

Recall that the estimator z is the index of the first 0 bit in the sketch $F_{S_1 \cup S_2}$. When computing in the clear, z can be trivially obtained by checking whether $F_{S_1 \cup S_2}[i] = 0$ from $i = 0$ and return the index i when hits the first 0 bit in the sketch. However, this algorithm is not data oblivious (i.e. the control flow and access pattern are dependent on the data). Thus the algorithm execution leaks information about the data and cannot be used in secure computation. In fact this is one of the biggest challenges when porting data structures to secure computation: most data structure based algorithms are not data oblivious and generic approaches for achieving data obliviousness incur a substantial cost.

To solve this problem, we design a data oblivious algorithm to combine the sketches and extract the estimator. The algorithm is shown in Algorithm 1 and a small example is shown in Figure 1. Conceptually, the algorithm does 3 things: (1) it creates $F_{S_1 \cup S_2}$ by bitwisely ORing F_{S_1} and F_{S_2}; (2) it sets all bits in $F_{S_1 \cup S_2}$ after the first 0 bit to 0; (3) it then adds up all bits. The sum equals the estimator z.

Algorithm 1: $Combine\text{-}then\text{-}sum(F_{S_1}, F_{S_2}, w)$

```
input : Two $w$-bit FM sketches $F_{S_1}$, $F_{S_2}$ and the size $w$
output: the index of the first 0 bit in $F_{S_1 \cup S_2}$
$\widehat{F}_{S_1 \cup S_2}$= new $w$-bit FM sketch;
$\widehat{F}_{S_1 \cup S_2}[0]$= $F_{S_1}[0] \vee F_{S_2}[0]$;
$sum = 0$;
for $i = 1$ to $w - 1$ do
    $\widehat{F}_{S_1 \cup S_2}[i]$=$(F_{S_1}[i] \vee F_{S_2}[i]) \wedge \widehat{F}_{S_1 \cup S_2}[i-1]$;
    $sum = sum + \widehat{F}_{S_1 \cup S_2}[i]$;
end
return $sum$;
```

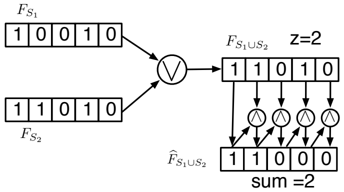


Fig. 1: Example of *Combine-then-sum* algorithm

The correctness of the algorithm is easy to prove. Let $F_{S_1 \cup S_2} = F_{S_1} \vee F_{S_2}$ and z be the estimator. If $z > 0$, then by the definition of z, for all $0 \leq i < z$ we have $F_{S_1 \cup S_2}[i] = 1$. In Algorithm 1, $\widehat{F}_{S_1 \cup S_2}[0]$=$F_{S_1 \cup S_2}[0]$=1 and for $1 \leq i < z$, $\widehat{F}_{S_1 \cup S_2}[i]$=$(F_{S_1}[i] \vee F_{S_2}[i]) \wedge \widehat{F}_{S_1 \cup S_2}[i-1]$=$F_{S_1 \cup S_2}[i] \wedge \widehat{F}_{S_1 \cup S_2}[i-1]$. By induction, for all $0 \leq i < z$, we have $\widehat{F}_{S_1 \cup S_2}[i]$=1 as well. Therefore, $\sum_{i=0}^{z-1} \widehat{F}_{S_1 \cup S_2}[i]$=$z$. For $i \geq z$, since $\widehat{F}_{S_1 \cup S_2}[z]$=0, we have $\widehat{F}_{S_1 \cup S_2}[z+1]$=$F_{S_1 \cup S_2}[z+1] \wedge \widehat{F}_{S_1 \cup S_2}[z]$=0, and similarly all the bits after are 0. Then the sum of the whole $\widehat{F}_{S_1 \cup S_2}$ is still z. If $z = 0$, then all bits in $\widehat{F}_{S_1 \cup S_2}$ are 0 and the sum equals $z = 0$.

It is also easy to verify that the algorithm is data oblivious: for any two input tuples (F_{S_1}, F_{S_2}, w) and (F'_{S_1}, F'_{S_2}, w), the memory access pattern and the control flow are exactly the same when executing the algorithm.

4.2.2 Efficient Protocol

The protocol is presented as Protocol 1. In the protocol, $\mathcal{P}_1$ and $\mathcal{P}_2$ jointly compute $\widehat{F}_{S_1 \cup S_2}$ and share each bit in it, then they use the homomorphic property to sum their shares locally to get a share of $\sum_{i=0}^{w-1} \widehat{F}_{S_1 \cup S_2}[i]$, i.e. z. The key idea in the protocol design is to fully utilize $\mathcal{P}_1$'s local knowledge to generate correlated secret shares, and to base computation on the correlated shares. By using secret sharing, the result can be obtained but is kept secret at the end of the protocol.

In the following, we explain the protocol. Let's start from step 1 which computes $\widehat{F}_{S_1 \cup S_2}[0] = F_{S_1}[0] \vee F_{S_2}[0]$, the first bit in $\widehat{F}_{S_1 \cup S_2}$. In this step, $\mathcal{P}_1$ generates the shares for 0 and 1 in a correlated way. Observe that $r^{\langle 0 \rangle} + r_0^{\langle 0 \rangle} = 0$ and $r^{\langle 0 \rangle} + r_1^{\langle 0 \rangle} = 1$, therefore $(r^{\langle 0 \rangle}, r_0^{\langle 0 \rangle})$ is a pair of shares for bit 0 and $(r^{\langle 0 \rangle}, r_1^{\langle 0 \rangle})$ is a pair of shares for bit 1. Although the shares are correlated, this does not affect security because $\mathcal{P}_2$ can only get one of $r_0^{\langle 0 \rangle}$,

[3] The inverse error function: $\mathrm{erf}^{-1}(x) = \sum_{k=0}^{\infty} \frac{c_k}{2k+1}(\frac{\sqrt{\pi}}{2}x)^{2k+1}$, where $c_k = \sum_{m=0}^{k-1} \frac{c_m c_{k-1-m}}{(m+1)(2m+1)}$ and $-1 < x < 1$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2721360, IEEE Transactions on Information Forensics and Security

Protocol 1 Secure Estimator Computation Protocol

Inputs The private inputs of $\mathcal{P}_1$ and $\mathcal{P}_2$ are the FM sketches F_{S_1} and F_{S_2} respectively. Each sketch encodes the party's private set. The auxiliary inputs include the security parameter λ, the sketch size w and a group $\mathbb{Z}_q$ where $q = m(w-1)+1$ for some integer m.

Outputs Let z be the index of the first 0 bit in the union sketch $F_{S_1 \cup S_2}$ that will be used later to estimate the union cardinality (see Section 4.1). $\mathcal{P}_1$ and $\mathcal{P}_2$ obtain $[\![z]\!]_1, [\![z]\!]_2 \in \mathbb{Z}_q$ respectively. Each party's output is a secret share of z such that $[\![z]\!]_1 + [\![z]\!]_2 = z$.

1) In round 0, $\mathcal{P}_1$ chooses a random number $r^{\langle 0\rangle} \xleftarrow{R} \mathbb{Z}_q$, then sets $r_0^{\langle 0\rangle} = -r^{\langle 0\rangle}$ and $r_1^{\langle 0\rangle} = 1 - r^{\langle 0\rangle}$. Then $\mathcal{P}_1$ and $\mathcal{P}_2$ run a $\binom{1}{2}$-OT in which $\mathcal{P}_2$ uses $F_{S_2}[0]$ as the selection bit and $\mathcal{P}_1$ uses $(r_0^{\langle 0\rangle}, r_1^{\langle 0\rangle})$ as input if $F_{S_1}[0] = 0$ or $(r_1^{\langle 0\rangle}, r_1^{\langle 0\rangle})$ if $F_{S_1}[0] = 1$.
2) Then in round i $(1 \le i < w)$, the two parties do the following:
   a) $\mathcal{P}_1$ chooses a random number $r^{\langle i\rangle} \xleftarrow{R} \mathbb{Z}_q$, then sets $r_0^{\langle i\rangle} = -r^{\langle i\rangle}$ and $r_1^{\langle i\rangle} = 1 - r^{\langle i\rangle}$.
   b) $\mathcal{P}_1$ prepares 4 strings to send:
$$\begin{cases} (r_0^{\langle i\rangle}, r_0^{\langle i\rangle}, r_0^{\langle i\rangle}, r_1^{\langle i\rangle}) & \text{if } r_0^{\langle i-1\rangle} \text{ is even} \wedge F_{S_1}[i] = 0 \\ (r_0^{\langle i\rangle}, r_0^{\langle i\rangle}, r_1^{\langle i\rangle}, r_1^{\langle i\rangle}) & \text{if } r_0^{\langle i-1\rangle} \text{ is even} \wedge F_{S_1}[i] = 1 \\ (r_0^{\langle i\rangle}, r_1^{\langle i\rangle}, r_0^{\langle i\rangle}, r_0^{\langle i\rangle}) & \text{if } r_0^{\langle i-1\rangle} \text{ is odd} \wedge F_{S_1}[i] = 0 \\ (r_1^{\langle i\rangle}, r_1^{\langle i\rangle}, r_0^{\langle i\rangle}, r_0^{\langle i\rangle}) & \text{if } r_0^{\langle i-1\rangle} \text{ is odd} \wedge F_{S_1}[i] = 1 \end{cases}$$
   c) Let $x^{\langle i-1\rangle}$ be the string received by $\mathcal{P}_2$ in round $i-1$. Let bit $b_0 = 0$ if $x^{\langle i-1\rangle}$ is even and $b_0 = 1$ if $x^{\langle i-1\rangle}$ is odd, let bit $b_1 = F_{S_2}[i]$. $\mathcal{P}_2$ gets a 2-bit integer $j = b_0 || b_1$.
   d) $\mathcal{P}_1$ and $\mathcal{P}_2$ run a $\binom{1}{4}$-OT in which $\mathcal{P}_1$ uses the 4 strings prepared in step 2b as input and $\mathcal{P}_1$ uses j obtained in step 2c as input.
3) $\mathcal{P}_1$ outputs $[\![z]\!]_1 = \sum_{i=0}^{w-1} r^{\langle i\rangle}$, $\mathcal{P}_2$ outputs $[\![z]\!]_2 = \sum_{i=0}^{w-1} x^{\langle i\rangle}$ where $x^{\langle i\rangle}$ is the string (an integer) received in round i.

$r_1^{\langle 0\rangle}$ through OT. In any case, $\mathcal{P}_1$ always keeps $r^{\langle 0\rangle}$ and the actual shared value then is determined by which share $\mathcal{P}_2$ receives. Now if $F_{S_1}[0] = 1$ then $\mathcal{P}_1$ knows the value $F_{S_1}[0] \vee F_{S_2}[0] = 1$ regardless of whether $F_{S_2}[0]$ is 0 or 1. Thus in the OT, $\mathcal{P}_1$ can use $(r_1^{\langle 0\rangle}, r_1^{\langle 0\rangle})$ as input so that $\mathcal{P}_2$ always gets the share of bit 1. If $F_{S_1}[0] = 0$ then the value of $F_{S_1}[0] \vee F_{S_2}[0]$ depends on $F_{S_2}[0]$. So in the OT, $\mathcal{P}_1$ uses $(r_0^{\langle 0\rangle}, r_1^{\langle 0\rangle})$ as input so that $\mathcal{P}_2$ always gets the share of 0 if $F_{S_2}[0] = 0$ or the share of 1 if $F_{S_2}[0] = 1$. Take the example in Fig. 1: $F_{S_1}[0] = 1$ so in the protocol $\mathcal{P}_1$ sends $(r_1^{\langle 0\rangle}, r_1^{\langle 0\rangle})$ and $\mathcal{P}_2$ always gets $r_1^{\langle 0\rangle}$ that is the share of bit 1.

In step 2, the two parties compute $\widehat{F}_{S_1 \cup S_2}[i] = (F_{S_1}[i] \vee F_{S_2}[i]) \wedge \widehat{F}_{S_1 \cup S_2}[i-1]$. At the end of each round in this step, $\mathcal{P}_2$ should receive a share $x^{\langle i\rangle}$. If $\widehat{F}_{S_1 \cup S_2}[i] = 0$, $\mathcal{P}_2$ should receive a share of bit 0, i.e. $x^{\langle i\rangle} = r_0^{\langle i\rangle}$; otherwise $\mathcal{P}_2$ should receive a share of bit 1, i.e. $x^{\langle i\rangle} = r_1^{\langle i\rangle}$. This step is much more difficult than ORing two bits because $\widehat{F}_{S_1 \cup S_2}[i-1]$ is required in the computation but may not be known by any party. Our insight is that we can solve this problem by utilizing the parity of the correlated shares. The two shares $r_0^{\langle i\rangle}, r_1^{\langle i\rangle}$ always differ by 1, therefore it is guaranteed that one of the shares is even and the other is odd. Thus the two parties can carry out the computation by matching parities of some shares which they can observe locally. In step 2b of the protocol, $\mathcal{P}_1$ prepares strings to send by observing the parity of the share of bit 0 generated in the last round ($r_0^{\langle i-1\rangle}$) and its current sketch bit $F_{S_1}[i]$, and later $\mathcal{P}_2$ receives a string by observing the parity of the share received in the last round ($x_0^{\langle i-1\rangle}$) and its current sketch bit $F_{S_s}[i]$. Again take the example in Fig. 1: for the second bit, $F_{S_1}[1] = 0$, thus if $r_0^{\langle 0\rangle}$ is even, $\mathcal{P}_1$ will send $(r_0^{\langle 1\rangle}, r_0^{\langle 1\rangle}, r_0^{\langle 1\rangle}, r_1^{\langle 1\rangle})$. Now since $F_{S_2}[1] = 1$ and $\mathcal{P}_2$ received $x^{\langle 0\rangle} = r_1^{\langle 0\rangle}$ in the last round that must be odd (because $r_0^{\langle 0\rangle}$ is even), then $j = (1||1)_b = 3$. So in this round $\mathcal{P}_2$ will receive the last string $r_1^{\langle 1\rangle}$ that is a share of bit 1. If $r_0^{\langle 0\rangle}$ is odd, $\mathcal{P}_1$ will send $(r_0^{\langle 1\rangle}, r_1^{\langle 1\rangle}, r_0^{\langle 1\rangle}, r_0^{\langle 1\rangle})$. Now $r_1^{\langle 0\rangle}$ must be even, then $j = (0||1)_b = 1$. So in this round $\mathcal{P}_2$ will receive the second string $r_1^{\langle 1\rangle}$ that is a share of bit 1.

The parity of $r_0^{\langle i-1\rangle}$, the parity of $x^{\langle i-1\rangle}$, the bit in $F_{S_1}[i]$ and the bit in $F_{S_2}[i]$ each have two possible values, thus there are $2^4 = 16$ combinations. The correctness of the protocol can be shown through Figure 2 that depicts the correspondence between each combination and the share received by $\mathcal{P}_2$ in round i. In each branch of the tree in the figure, if $r_0^{\langle i-1\rangle}$ and $x^{\langle i-1\rangle}$ have the same parity then $x^{\langle i-1\rangle}$ must equal $r_0^{\langle i-1\rangle}$ which is a share of bit 0, thus $\widehat{F}_{S_1 \cup S_2}[i-1] = 0$. Otherwise $\widehat{F}_{S_1 \cup S_2}[i-1] = 1$. $\widehat{F}_{S_1 \cup S_2}[i] = (F_{S_1}[i] \vee F_{S_2}[i]) \wedge \widehat{F}_{S_1 \cup S_2}[i-1]$ can be computed along the branch and the leaf nodes is the share that $\mathcal{P}_2$ receive. For example, in the leftmost branch $r_0^{\langle i-1\rangle}$ and $x^{\langle i-1\rangle}$ have the same parity (even) so $\widehat{F}_{S_1 \cup S_2}[i-1] = 0$ and $F_{S_1}[i], F_{S_2}[i]$ both are 0, so $\widehat{F}_{S_1 \cup S_2}[i] = (0 \vee 0) \wedge 0 = 0$. Corresponding to this branch, in the protocol step 2b $\mathcal{P}_1$ prepares four strings in case 1 because $r_0^{\langle i-1\rangle}$ is even and $F_{S_1}[i] = 0$, and $\mathcal{P}_2$ receives the first of the four strings because $x^{\langle i-1\rangle}$ is even and $F_{S_2}[i] = 0$. The string received by $\mathcal{P}_2$ is $r_0^{\langle i\rangle}$ which is a share of $\widehat{F}_{S_1 \cup S_2}[i] = 0$.

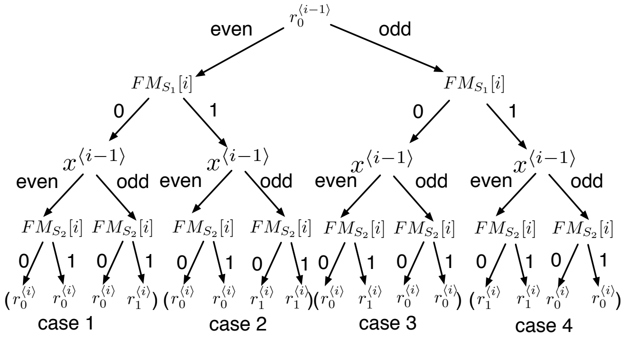

$r_0^{\langle i-1\rangle}$
even
odd
$FM_{S_1}[i]$
$FM_{S_1}[i]$
0
1
0
1
$x^{\langle i-1\rangle}$
$x^{\langle i-1\rangle}$
$x^{\langle i-1\rangle}$
$x^{\langle i-1\rangle}$
even
odd
even
odd
even
odd
even
odd
$FM_{S_2}[i]$
0
1
$(r_0^{\langle i\rangle}$ $r_0^{\langle i\rangle}$ $r_0^{\langle i\rangle}$ $r_1^{\langle i\rangle})$
$(r_0^{\langle i\rangle}$ $r_0^{\langle i\rangle}$ $r_1^{\langle i\rangle}$ $r_1^{\langle i\rangle})$
$(r_0^{\langle i\rangle}$ $r_1^{\langle i\rangle}$ $r_0^{\langle i\rangle}$ $r_0^{\langle i\rangle})$
$(r_1^{\langle i\rangle}$ $r_1^{\langle i\rangle}$ $r_0^{\langle i\rangle}$ $r_0^{\langle i\rangle})$
case 1
case 2
case 3
case 4

Fig. 2: All cases in step 2

In step 3, $\mathcal{P}_1$ and $\mathcal{P}_2$ locally sum the shares. Each $r^{\langle i\rangle}$ held by $\mathcal{P}_1$ and each $x^{\langle i\rangle}$ held by $\mathcal{P}_2$ are the two shares of $\widehat{F}_{S_1 \cup S_2}[i]$. By the homomorphic property, the sums $\sum_{i=0}^{w-1} r^{\langle i\rangle}$ and $\sum_{i=0}^{w-1} x^{\langle i\rangle}$ are the shares of z.

Our final remark is regarding the choice of q, which is $m \cdot (w-1)+1$. The reason is that in order to increase accuracy, we need to average estimators extracted from m sketches. The value of $z^{\langle i\rangle}$ is at most $w-1$, then the sum of m estimators is at most $m \cdot (w-1)$. We need $\mathbb{Z}_q$ to be large enough to accommodate this sum. Then q needs to be at least $m \cdot (w-1)+1$. For efficiency, we choose q to be exactly $m \cdot (w-1)+1$.

4.2.3 Efficiency Comparison to the Generic Approach

It is possible to implement Algorithm 1 using generic techniques such as garbled circuits (GC). The cost of the GC protocol consists of two parts:

1) *Transferring input wires*: This requires w invocations of $\binom{1}{2}$-OT, which can use the C-OT optimization in [37].
2) *Building, transferring and evaluating a garbled Boolean circuit*: The circuit consists of w OR-gates, $w-1$ AND-gates and a circuit to compute the Hamming weight of a w-bit

string. The most efficient Hamming weight circuit [45] requires $w - \mathsf{HW}(w)$ AND-gates, where $\mathsf{HW}(w)$ is the Hamming weight of the binary representation of the integer w. In total, the number of non-free gates is $3w-1-\mathsf{HW}(w)$. However, the w OR-gates can be evaluated at the wire transferring step using OT (similar to what we do in step 1 of Protocol 1), thus the number of non-free gates can be reduced to $2w-1-\mathsf{HW}(w)$.

In comparison, our protocol (Protocol 1) requires 1 invocations of $\binom{1}{2}$-OT and $w-1$ invocations of $\binom{1}{4}$-OT. We can use the C-OT optimization for the $\binom{1}{2}$-OT and the R-OT optimization (also from [37]) for the $\binom{1}{4}$-OT (See Appendix D for details).

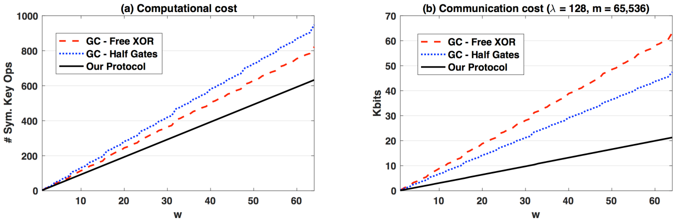


Fig. 3: Efficiency comparison

Computational Cost: We estimate the computational cost by counting the number of cryptographic operations. Each $\binom{1}{2}$-OT requires 3 symmetric key operations when obtained from OT extension. The cost of each non-free gate is dependent on which optimization technique is used. At present, the most popular strategy for optimizing GC is to make the XOR gates free. To do so, one can use either the Free XOR [46] with point-and-permute [47] and garbled row reduction [48] technique (Free XOR for short) or the Half Gates technique [49]. Free XOR requires 4 symmetric key operations to garble and 1 symmetric key operations to evaluate a non-free gate, while Half Gates requires 4 and 2 symmetric key operations respectively.

The total number of symmetric key operations required by the GC protocol is then $3w+5\cdot(2w-1-\mathsf{HW}(w)) = 13w-5-5\cdot\mathsf{HW}(w)$ when using Free XOR or $3w+6\cdot(2w-1-\mathsf{HW}(w)) = 15w-6-6\cdot\mathsf{HW}(w)$ when using Half Gates. The total number of symmetric key operations in our protocol is $3\cdot(2w-1)+4\cdot(w-1) = 10w-7$. In practice, w is a small integer and the computational cost of our protocol is about 80% of that of the GC protocol if it uses Free XOR, or about 70% if it uses Half Gates. Fig. 3(a) plots the number of symmetric key operations in each protocol when w varies from 1 to 64.

Communication Cost: The total communication cost of the GC protocol again depends on the optimization technique. Using Free XOR, each non-free gate has 3 entries. Using Half Gates, each non-free gate has 2 entries. The size of each entry is λ bits, where λ is the security parameter. The cost for transferring input wires is 2λ bits per wire (using C-OT) in both cases.

In total, the communication cost of the GC protocol is $(2w-1-\mathsf{HW}(w))\cdot 3\lambda + w\cdot 2\lambda = 8w\lambda - 3\lambda - 3\cdot\mathsf{HW}(w)\lambda$ bits if Free XOR is used, or $(2w-1-\mathsf{HW}(w))\cdot 2\lambda + w\cdot 2\lambda = 6w\lambda - 2\lambda - 2\cdot\mathsf{HW}(w)\lambda$ bits if Half Gates is used. The total communication of our protocol is $\lambda + \log q$ bits for the first $\binom{1}{2}$-OT (using C-OT), and $(2w-2)\cdot\lambda + 4(w-1)\cdot\log q$ bits for the following $w-1$ invocations of $\binom{1}{4}$-OT (using R-OT), where $q = m\cdot(w-1)+1$. Roughly, for reasonable λ (128 or 256), w (1 to 64) and m (4,096 to 1,048,576), the communication cost of our protocols is about 35% of that of the GC protocol if it uses Free XOR, or 45% if it uses Half gate. Fig. 3(b) plots the communication cost in each protocol when fixing $\lambda = 128, m = 65,536$ and varying w from 1 to 64.

4.3 Secure Cardinality Estimation

4.3.1 The Protocol

In Section 4.2.2, we showed how to compute the estimator z of the union cardinality from a pair of FM sketches. As mentioned in Section 4.1, in order to increase accuracy, we need to compute m estimators from m different pairs of sketches. This requires m executions of Protocol 1. As illustrated in Fig. 4, in the ith run, $\mathcal{P}_1$ and $\mathcal{P}_2$ each obtains a share $[\![z]\!]_1^{\langle i\rangle}, [\![z]\!]_2^{\langle i\rangle}$ respectively. Then the two parties can locally sum the shares to get $[\![Z]\!]_1 = \sum_{i=0}^{m-1}[\![z]\!]_1^{\langle i\rangle}$ and $[\![Z]\!]_2 = \sum_{i=0}^{m-1}[\![z]\!]_2^{\langle i\rangle}$, where $[\![Z]\!]_1$ and $[\![Z]\!]_2$ are shares of $Z = \sum_{i=0}^{m-1} z^{\langle i\rangle}$. The two parties can then use Protocol 2 to compute Equation 1 and estimate the union cardinality.

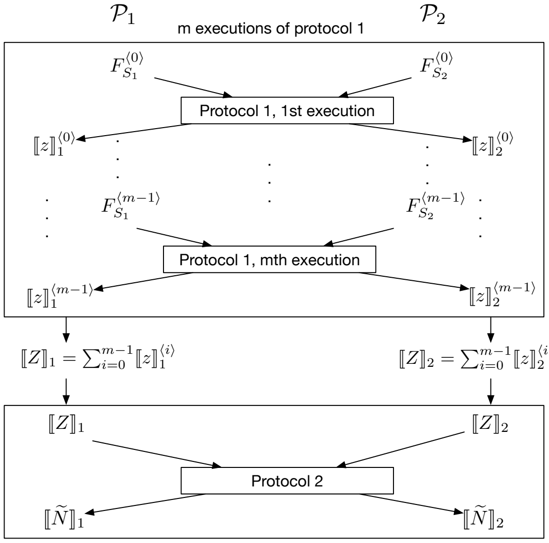

$\mathcal{P}_1$
m executions of protocol 1
$\mathcal{P}_2$
$F_{S_1}^{\langle 0\rangle}$
$F_{S_2}^{\langle 0\rangle}$
Protocol 1, 1st execution
$[\![z]\!]_1^{\langle 0\rangle}$
$[\![z]\!]_2^{\langle 0\rangle}$
$F_{S_1}^{\langle m-1\rangle}$
$F_{S_2}^{\langle m-1\rangle}$
Protocol 1, mth execution
$[\![z]\!]_1^{\langle m-1\rangle}$
$[\![z]\!]_2^{\langle m-1\rangle}$
$[\![Z]\!]_1 = \sum_{i=0}^{m-1}[\![z]\!]_1^{\langle i\rangle}$
$[\![Z]\!]_2 = \sum_{i=0}^{m-1}[\![z]\!]_2^{\langle i\rangle}$
$[\![Z]\!]_1$
$[\![Z]\!]_2$
Protocol 2
$[\![\widetilde{N}]\!]_1$
$[\![\widetilde{N}]\!]_2$

Fig. 4: Connection between Protocol 1 to Protocol 2

Our idea is to use a lookup table. A lookup table is a data structure that encodes a function with a small input domain to speed up computation. As we can see, in Equation 1, m, κ, ϕ are all public constants. So the equation is a function with a single argument Z which is an integer from $\mathbb{Z}_q$ where $q = m\cdot(w-1)+1$. In practical cases, m needs to be of the order of $10^3 - 10^4$ so the standard deviation of Z is about 10^{-2}, and w is unlikely to be greater than 50. Then $m\cdot(w-1)$ is of the order of $10^4 - 10^5$ which is small enough for a lookup table. Our lookup table based protocol is presented below (Protocol 2).

In the protocol, $\mathcal{P}_1$ first computes Equation 1 for each possible value of Z, and stores the result in the lookup table. So $T[i]$ stores the estimated cardinality when $Z = i$. Note this computation is all in plaintext, thus we avoid entirely the expensive secure floating point computation. Each $T[i]$ is an integer after rounding and is in $\mathbb{Z}_{2^w}$ because the way we choose w ensures that 2^w is larger than any possible cardinality. Then $\mathcal{P}_1$ picks a single r and creates correlated shares of all entries in T. Again, since later the protocol uses OT and $\mathcal{P}_2$ is guaranteed to receive only one share, this is secure. The next thing $\mathcal{P}_1$ does is to ensure that $\mathcal{P}_2$ can receive the correct entry $T'[Z]$. None of the parties knows Z but each holds a share. The combined effect of shifting and OT is that $\mathcal{P}_2$ will receive $T''[[\![Z]\!]_2] = T'[[\![Z]\!]_2 + [\![Z]\!]_1] = T'[Z]$, and

Protocol 2 Secure Cardinality Estimation Protocol

Inputs The private input of $\mathcal{P}_1$ is a secret share $[\![Z]\!]_1$ and the private input of $\mathcal{P}_1$ is a secret share $[\![Z]\!]_2$ such that $[\![Z]\!]_1 + [\![Z]\!]_2 = Z = \sum_{i=0}^{m-1} z^{\langle i \rangle}$, i.e. Z is the sum of m union cardinality estimators. The auxiliary inputs include the security parameter λ, the public parameters $m, w, \kappa, \phi, q = m(w-1)+1$, and $\mathbb{Z}_{2^w}$.

Outputs Let $\widetilde{N}$ be the estimate to be computed, $\mathcal{P}_1$ and $\mathcal{P}_2$ obtain $[\![\widetilde{N}]\!]_1, [\![\widetilde{N}]\!]_2 \in \mathbb{Z}_{2^w}$ respectively. Each party's output is a secret share of $\widetilde{N}$ and satisfies $[\![\widetilde{N}]\!]_1 + [\![\widetilde{N}]\!]_2 = \widetilde{N}$.

1) $\mathcal{P}_1$ computes a lookup table T which is a vector that has q entries. $\mathcal{P}_1$ first computes for $0 \le i \le q-1$

$$T[i] = \lceil \frac{2^{\frac{i}{m}} - 2^{-\kappa \cdot \frac{i}{m}}}{\phi} \rfloor$$

$\mathcal{P}_1$ then picks a single $r \xleftarrow{R} \mathbb{Z}_{2^w}$ and for all $0 \le i \le q-1$ computes $T'[i] \equiv T[i] - r \mod 2^w$. $\mathcal{P}_1$ then circularly shifts T' to the left $[\![Z]\!]_1$ places, i.e. let $T''[i] = T'[j]$ where $j \equiv i + [\![Z]\!]_1 \bmod q$.

2) $\mathcal{P}_1$ and $\mathcal{P}_2$ run a $\binom{1}{q}$-OT in which $\mathcal{P}_1$ uses T'' as input, $\mathcal{P}_2$ uses $[\![Z]\!]_2$ as input and receives $T''[[\![Z]\!]_2]$.

3) $\mathcal{P}_1$ outputs $[\![\widetilde{N}]\!]_1 = r$ and $\mathcal{P}_2$ outputs $[\![\widetilde{N}]\!]_2 = T''[[\![Z]\!]_2]$.

$T''[[\![Z]\!]_2] + r = T'[Z] + r = T[Z]$. So each party indeed obtains a share of the estimated cardinality.

4.3.2 Efficiency Comparison to the Generic Approach

The computation of Equation 1 needs to be done in floating point numbers. More concretely, Equation 1 requires 2 divisions, 2 exponentiations and 1 subtraction operations in floating point numbers. In the past, secure floating point computation was both complex and inefficient [50], [51], [52]. Recently, it has been shown that optimized floating point circuits can be generated using hardware circuits synthesis tools [53] and the GC protocol using the optimized circuits can be quite efficient. We now compare the efficiency of Protocol 2 against that of the GC protocol.

Computational Cost: The sizes of the floating point operation circuits are dependent on the bit size of the floating point number. For 32-bit single precision floating point operations, the addition circuit has 1,820 AND-gates, the division circuit has 5,395 AND-gates and the base 2 exponentiation gate has 9,740 AND-gates. For 64-bit double precision floating point operations, the gate numbers are 4,303, 22,741 and 21,431 respectively. For building and evaluating a 32-bit circuit to compute the estimation, the total number of symmetric key operations is 160,450 if the circuit uses Free XOR, or 192,540 if it uses Half Gate. For building and evaluating a 64-bit circuit to compute the estimation, the total number of symmetric key operations is 463,235 if the circuit uses Free XOR, or 555,882 if it uses Half Gate. There will also be some other costs, e.g. converting the arithmetic shares from Protocol 1 to Boolean shares [20], and OT for transferring input wires. However the additional costs are small (a few hundreds symmetric key operations) and can be safely omitted. Our protocol requires one invocation of $\binom{1}{q}$-OT, in which the computation is dominated by the q symmetric key operations for masking the strings. The number q equals $m \cdot (w-1)+1$. Therefore depending on the value of the parameters m and w, one can decide whether to use Protocol 2 or the GC protocol. In Table 1, we show some concrete examples. In Table 1, each row corresponds to a fixed value for m, and each cell in the row shows the largest value of w, for which Protocol 2 is computationally more efficient than the GC protocol based on Free XOR or Half Gates. For example, if $m = 4096$ and single precision (32-bit) is enough, then we should use Protocol 2 whenever $w \le 40$ (or equivalently if the set cardinality will not exceed 2^{36}); but if $m = 65536$, we should use a GC protocol in almost all cases because $w \le 3$ is too small to be useful.

GC / **m**	32-bit		64-bit	
	Free XOR	Half Gates	Free XOR	Half Gates
4096	40	48	114	136
16384	10	12	29	34
65536	3	3	8	9

TABLE 1: Each cell contains the largest value of w for which Protocol 2 is computationally more efficient than the GC protocol with Free XOR or with Half Gates.

Communication Cost: For the GC protocol that uses a circuit of c gates, the communication cost is $c \cdot 3\lambda$ bits if Free XOR is used, or $c \cdot 2\lambda$ bits if Half Gates is used. For Protocol 2, the communication cost is dominated by transferring the q masked strings, which is in total $q \cdot \ell$ where ℓ is the bit-size of the floating point numbers. Then which one is more efficient depends on the parameters and the optimization technique. Again we worked out the largest value of w for which Protocol 2 is more efficient than the GC protocol with Free XOR and the GC protocol with Half Gates (see Table 2).

GC / **m**	32-bit		64-bit	
	Free XOR	Half Gates	Free XOR	Half Gates
4096	95	63	136	91
16384	24	16	34	23
65536	6	4	9	6

TABLE 2: Each cell contains the largest value of w for which Protocol 2 is more efficient than the GC protocol with Free XOR or with Half Gates, in terms of communication cost.

4.4 Fixed-key Masking

The efficiency of Protocol 2 depends on the underlying $\binom{1}{n}$-OT ($n = q$ in our protocol). As we have already shown in Section 3.2, the cost of $\binom{1}{n}$-OT consists of two parts: $\log(n)$ invocations of $\binom{1}{2}$-OT and $n\log(n)$ pseudorandom function invocations for masking the sender's strings. In the past when $\binom{1}{2}$-OT had to be based on public key operations, the first part dominated the cost. But now we can obtain $\binom{1}{2}$-OT through OT extension, then the second part becomes dominating. A more efficient masking scheme then implies more efficient $\binom{1}{n}$-OT.

In this section, we present an efficient fixed-key masking scheme. Our design is in line with the fixed-key garbling schemes that have led to a significant improvement in the computation of garbled circuits [22]. In the new masking scheme, the cost of masking in $\binom{1}{n}$-OT is reduced to n invocations of a random permutation. The random permutation can be instantiated using a block cipher such as AES with a public and fixed key. This allows further efficiency improvement by taking advantage of the AES-NI set [54] and avoiding the cost caused by frequent key scheduling.

Let $(\mathcal{E}, \mathcal{D})$ be a block cipher where $\mathcal{E}, \mathcal{D}$ are the encryption and decryption algorithms respectively. Let ck be a key generated for the cipher and is public. We will model $\mathcal{E}_{ck}(\cdot)$ as a random permutation π and $\mathcal{D}_{ck}(\cdot)$ as the inverse permutation π^{-1} [55]. The masking scheme has four algorithms:

- $\mathsf{Gen}(n, \lambda)$: given n the algorithm uniformly generates a $l \times 2$ key matrix K where $l = \lceil \log(n) \rceil$ and each cell $K_{i,j}$ is a uniformly random λ-bit key. Without loss of generality, we assume the block size of the cipher is also λ bits.
- $\mathsf{Key}(K, i)$: given the key matrix K and an integer $0 \le i \le n-1$, return the ith masking key. Let the binary representation

of i be $b_0 b_1, ..., b_{l-1}$, the masking key $mk = \bigoplus_{j=0}^{l-1} K_{j,b_j}$, where b_j is the jth bit of i.

- $\mathsf{Mask}(K, P)$: Mask an n-vector P such that each element in P is a bit string of λ-bit. For each element $P[i]$, compute $mk_i = \mathsf{Key}(K, i)$. The masked element $\widetilde{P}[i] = \pi(mk_i) \oplus mk_i \oplus P[i]$
- $\mathsf{Unmask}(\widetilde{P}, i)$: given the masked vector $\widetilde{P}$ and an index i, unmask $P[i]=\widetilde{P}[i]\oplus\pi(mk_i)\oplus mk_i$

This masking scheme is to be used with $\binom{1}{n}$-OT. The sender runs $\mathsf{Gen}, \mathsf{Key}, \mathsf{Mask}$ to mask the n strings that will be sent to the receiver. The receiver will use its selection number I to receive $\log(n)$ keys in $\log(n)$ $\binom{1}{2}$-OT. The keys will allow the receiver to reconstruct mk_I but not the other masking keys. With mk_I, the receiver can unmask the sender's Ith string $P[I]$. The formal security definition and proof can be found in Appendix C. At a high level, the masking scheme is secure if an adversary who knows the $\log(n)$ keys corresponding to I and $\widetilde{P}$ can learn $P[I]$ but nothing about all other $P[j], j \neq I$, even if the adversary has oracle access to π and π^{-1}.

A remark on $\binom{1}{n}$-OT extension In our paper, we implement $\binom{1}{n}$-OT by invoking $\binom{1}{2}$-OT extension $\log(n)$ times and masking the n strings to be sent. Alternatively, one can use $\binom{1}{n}$-OT extension with the masking scheme to implement $\binom{1}{n}$-OT.

In [38], a $\binom{1}{n}$-OT extension protocol was presented such that the $\log(n)$ invocations of $\binom{1}{2}$-OT extension can be replaced by one invocation to the $\binom{1}{n}$-OT extension (the masking part remains the same). This protocol works when $n \leq 2 \cdot \lambda$, where λ is the security parameter. The cost of one invocation of $\binom{1}{n}$-OT extension is about the same as 2 invocations of $\binom{1}{2}$-OT extension. In [56], a new protocol for $\binom{1}{n}$-OT extension based on pseudorandom code was presented. The new protocol works for arbitrary n. The cost of one invocation is about the same as the cost of 4 invocations of $\binom{1}{2}$-OT extension.

We present some analysis regarding whether $\binom{1}{n}$-OT extension could improve the efficiency of our protocols:

- The $\binom{1}{n}$-OT extension protocol requires an additional 2λ ([38]) or 3λ to 4λ ([56]) base OTs to setup. Since we use OT extension already, these base OTs can obtained through OT extension.
- In Protocol 1, we need $\binom{1}{4}$-OT. If we use [38], the cost of 1 invocation is the same as 2 invocations of $\binom{1}{2}$-OT extension. If we use [56], the cost of 1 invocation is actually higher. Thus, using $\binom{1}{n}$-OT extension will not improve the efficiency of Protocol 1.
- In Protocol 2, we need $\binom{1}{q}$-OT. The parameter q is too large for [38]. We can use [56] in this case. If so, we will use 1 invocation of the $\binom{1}{n}$-OT extension instead of $\log(q)$ invocations of $\binom{1}{2}$-OT extension. However, since Protocol 2 is only invoked once in the PSU-CA protocol, this improvement will be offset by the increased number of base OT. Recall that q equals $m \cdot (w-1)$ and in most practical cases, q is of the order of $10^4 - 10^5$. Therefore $\log(q)$ is usually no more than 20. In comparison, when $\lambda = 128$, the $\binom{1}{n}$-OT extension in [56] requires at least 384 more invocations of base OT (that can be obtained using the same $\binom{1}{2}$-OT extension). Thus, using $\binom{1}{n}$-OT extension will not improve the efficiency of Protocol 2.

4.5 PSU-CA and PSI-CA Protocols

We now present the PSU-CA and PSI-CA protocols. As we mentioned earlier, the PSI-CA protocol can be obtained from the PSU-CA protocol. The PSU-CA protocol is presented in Protocol 3 and the security analysis of the protocol can be found in Appendix B. Building sketches does not involve cryptographic operations and can be done offline, as this does not require interacting with the other party. Thus we assume the parties have pre-computed the sketches before running the protocol and they use the sketches as the input to the protocol.

Protocol 3 PSU-CA Protocol

Inputs The private inputs of $\mathcal{P}_1$ and $\mathcal{P}_2$ are the m FM sketches $F_{S_1}^{\langle 0 \rangle}, ..., F_{S_1}^{\langle m-1 \rangle}$ and $F_{S_2}^{\langle 0 \rangle}, ..., F_{S_2}^{\langle m-1 \rangle}$ respectively. Each pair of sketches $(F_{S_1}^{\langle i \rangle}, F_{S_2}^{\langle i \rangle})$ encodes the private sets of the parties, using the same hash function h_i. The auxiliary inputs include the security parameter λ, the sketch size $w = \log(\mathcal{N}) + 4$ where $\mathcal{N}$ is the max possible cardinality of the private sets, the parameter m that controls accuracy, and constants κ, ϕ.

Outputs $\mathcal{P}_1$ and $\mathcal{P}_2$ obtain $[\![\widetilde{N}]\!]_1, [\![\widetilde{N}]\!]_2 \in \mathbb{Z}_{2^w}$ respectively. Each party's output is a secret share of $\widetilde{N}$ and satisfies $[\![\widetilde{N}]\!]_1 + [\![\widetilde{N}]\!]_2 = \widetilde{N}$, where $\widetilde{N}$ is the estimated union cardinality.

1) $\mathcal{P}_1$ and $\mathcal{P}_2$ run Protocol 1 exactly m times. In the ith run, they use $(F_{S_1}^{\langle i \rangle}, F_{S_2}^{\langle i \rangle})$ and obtain $[\![z]\!]_1^{\langle i \rangle}$, $[\![z]\!]_2^{\langle i \rangle}$ respectively.
2) $\mathcal{P}_1$ and $\mathcal{P}_2$ compute locally $[\![Z]\!]_1 = \sum_{i=0}^{m-1} [\![z]\!]_1^{\langle i \rangle}$ and $[\![Z]\!]_2 = \sum_{i=0}^{m-1} [\![z]\!]_2^{\langle i \rangle}$.
3) $\mathcal{P}_1$ and $\mathcal{P}_2$ run Protocol 2 with $[\![Z]\!]_1$ and $[\![Z]\!]_2$ as input and obtain $[\![\widetilde{N}]\!]_1$ and $[\![\widetilde{N}]\!]_2$ respectively.
4) $\mathcal{P}_1$ outputs $[\![\widetilde{N}]\!]_1$, $\mathcal{P}_2$ outputs $[\![\widetilde{N}]\!]_2$.

For the PSI-CA protocol, the only difference is that in the last step $\mathcal{P}_1$ outputs $[\![\widetilde{I}]\!]_1 = |S_1| - [\![\widetilde{N}]\!]_1$, $\mathcal{P}_2$ outputs $[\![\widetilde{I}]\!]_2 = |S_2| - [\![\widetilde{N}]\!]_2$. In this step each party converts its own share of the union cardinality to a share of the intersection cardinality using the cardinality of its own set. This step is done locally and the parties do not need to know the cardinality of the other party's set. The output of $\mathcal{P}_1$ and $\mathcal{P}_2$ in PSI-CA adds up to $|S_1| - [\![\widetilde{N}]\!]_1 + |S_2| - [\![\widetilde{N}]\!]_2 = |S_1| + |S_2| - \widetilde{N} \approx |S_1| + |S_2| - |S_1 \cup S_2| = |S_1 \cap S_2|$.

Complexity: In both the PSU-CA and PSI-CA protocols, the parties first run m times of Protocol 1, whose cost is $2w - 1$ invocations of $\binom{1}{2}$-OT. Then the parties run Protocol 2 whose cost is one $\binom{1}{q}$-OT and $q = m \cdot (w-1) + 1$. The parameter m is a constant once the desirable error bound is fixed. The parameter $w = \log(\mathcal{N}) + 4$ where $\mathcal{N}$ is the maximum possible cardinality of the private sets. So the computational and communication complexities are both $O(\log(\mathcal{N}))$. Note that $\mathcal{N}$ is the maximum possible cardinality instead of the actual cardinalities of the private sets. This is because the parties do not and should not know the cardinality of the other party's private set. They can set a large enough w so that the sketches can encode any set that is smaller than 2^{w-4}. For example, to encode any sets with size up to 1 million, we set $w = 24$.

Relative Error of PSI-CA: In the PSI-CA protocol we obtain the intersection cardinality from the estimated union cardinality. The relative error then is in terms of the union cardinality $\frac{|\widetilde{N} - |S_1 \cup S_2||}{|S_1 \cap S_2|}$. We can adjust m to control the relative error if $|S_1 \cap S_2|$ is not very small compared to $|S_1 \cup S_2|$. In many data mining applications the condition often holds and intersection cardinality can be estimated fairly well based on the inclusion-exclusion principle [57], [58].

5 Multiparty Protocols

Since Protocol 1 and Protocol 2 require OT, they cannot be directly migrated to the multiparty setting. However, the protocols can be re-implemented with standard secret sharing-based multiparty secure computation schemes e.g. [59], [60], [61]. Computing the estimator requires bitwise OR and AND protocols, and an integer addition protocol. Those are standard building blocks that are readily available in all schemes mentioned above. We use a lookup table in estimating union cardinality, and secure lookup tables are also available [62]. Then the PSU-CA protocol can be easily built after the two building blocks are built. Note here our intention is to show feasibility. More efficient protocols are possible but we do not intend to do any optimization now and leave this for future investigation. Working with standard secure computation building blocks implies that the protocol is secure (by the composition theorem [63]), thus the security proof for the multiparty protocols is omitted.

However, migrating the PSI-CA protocol to multiparty setting is not easy. This is because computing the intersection cardinality of τ sets using the inclusion exclusion principle requires exponential time in τ. For example, in the three-parties setting, $|S_1 \cap S_2 \cap S_3| = |S_1 \cup S_2 \cup S_3| - |S_1| - |S_2| - |S_3| + |S_1 \cap S_2| + |S_1 \cap S_3| + |S_2 \cap S_3|$. Therefore in the multiparty setting we do not use the inclusion and exclusion principle. Instead we use Min-Max sketches [26] to compute the intersection cardinality from shared union cardinality. This reduces the complexity to linear in τ. In the following, we introduce Min-Max sketches and then present the protocol.

5.1 Min-Max Sketches

A Min-Max sketch [26] is a summary of a set that can be used for estimating Jaccard index of sets. In this paper, we use it to obtain the cardinality of the intersection of multiple sets.

A Min-Max sketch consists of 2 vectors of k hash values. Let S be a set and $h_0, \dots, h_{k-1}$ be k independent collision resistant hash functions that map inputs uniformly to l bit integers. We define $h_i^{min}(S)$ as the element in S that has the lowest hash value, i.e. $h_i^{min}(S) = x$ such that $x \in S$ and $\forall y \in S \wedge y \neq x, h_i(x) < h_i(y)$. Similarly, we define $h_i^{max}(S)$ as the element in S that has the highest hash value. A k-Min-Max sketch of S (denoted by M_S) consists of two vectors: $M_S^{min} = (h_0^{min}(S), ..., h_{k-1}^{min}(S))$ and $M_S^{max} = (h_0^{max}(S), ..., h_{k-1}^{max}(S))$.

For two sets S_1 and S_2, the Jaccard index is defined as $J = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$. Given two k-Min-Max sketches M_{S_1} and M_{S_2} built with the same set of k hash functions, it is clear that $Pr[h_i^{min}(S_1) = h_i^{min}(S_2)] = J$ since $h_i^{min}(S_1) = h_i^{min}(S_2)$ only happens when the element is in the intersection and the probability that this element is the minimal is $\frac{1}{|S_1 \cup S_2|}$. Similarly, $Pr[h_i^{max}(S_1) = h_i^{max}(S_2)] = J$ follows the same line of reasoning

Thus we have an unbiased estimator of J:

$$\widetilde{J} = \frac{1}{2k} \sum_{i=0}^{k-1} (eq(M_{S_1}^{min}[i], M_{S_2}^{min}[i]) + eq(M_{S_1}^{max}[i], M_{S_2}^{max}[i])) \quad (3)$$

where eq is the equality function such that $eq(x, y) = 1$ if $x = y$ and $eq(x, y) = 0$ otherwise. The standard deviation of $\widetilde{J}$ is $\sqrt{\frac{J}{2k} \cdot \frac{|S_1 \cup S_2| + |S_1 \cap S_2| - 2}{|S_1 \cup S_2| - 1} - \frac{J^2}{k}} \leq \sqrt{\frac{J(1-J)}{2k}} \leq \sqrt{\frac{1}{8k}}$.

This can be generalized to τ sets case where the *generalized* Jaccard index is defined as $J_\tau = \frac{|\bigcap_{i=1}^{\tau} S_i|}{|\bigcup_{i=1}^{\tau} S_i|}$. Now $\widetilde{J_\tau}$ with τ sketches $M_{S_1}, \dots, M_{S_\tau}$ is defined similarly as:

$$\widetilde{J}_\tau = \frac{\sum_{i=0}^{k} (eq(M_{S_1}^{min}[i], \dots, M_{S_\tau}^{min}[i]) + eq(M_{S_1}^{max}[i], \dots, M_{S_\tau}^{max}[i]))}{2k} \quad (4)$$

where eq now is the equality function over multiple values such that $eq(x_1, \dots, x_\tau) = 1$ if $x_1 = \dots = x_\tau$ and $eq(x_1, \dots, x_\tau) = 0$ otherwise. The accuracy of the estimation using Min-Max sketches can be adjusted by changing k. More specifically:

Theorem 2. *Let $S_1 \dots, S_\tau$ be sets and J_τ be the generalized Jaccard index. Let $\widetilde{J_\tau}$ be the estimate obtained from computed using Equation (4). For any $\epsilon, \delta \in (0, 1)$, it holds that:*

$$Pr[\frac{|\widetilde{J_\tau} - J_\tau|}{J_\tau} \leq \epsilon] \geq 1 - \delta \quad (5)$$

when $k \geq \frac{(\text{erf}^{-1}(1-\delta))^2}{2\epsilon^2 J_\tau^2}$ where erf^{-1} is the inverse error function.

The proof of Theorem 2 is similar to that of Theorem 1 and is omitted. As we can see, the threshold for k depends on J_τ. For example, if we fix $\delta = 0.001$, then when $\epsilon = 0.04$, k should be at least $\frac{364}{J_\tau^2}$; when $\epsilon = 0.01$, k should be at least $\frac{5817}{J_\tau^2}$.

5.2 Multiparty PSI-CA Protocol

Given the generalized Jaccard index (GJI) over τ sets and the union cardinality of the τ sets, we can compute the intersection cardinality $|\bigcap_{i=1}^{\tau} S_i| = J_\tau \cdot |\bigcup_{i=1}^{\tau} S_i|$. We can estimate the union cardinality using FM sketches, then what is left is to estimate the GJI over the sets. To estimate GJI and then compute the intersection cardinality, we only need secure equality test, integer multiplication and floating point division protocols, which are also basic building blocks in secret sharing-based multiparty schemes (e.g. [64], [65]). We can just use them as black boxes. The protocol is shown in Protocol 4.

Protocol 4 Multiparty PSI-CA Protocol

Inputs The private inputs of $\mathcal{P}_1, ..., \mathcal{P}_\tau$ are the Min-Max sketches $M_{S_1}, ..., M_{S_\tau}$ and shares of estimated union cardinality $[\![\widetilde{N}]\!]_1, ..., [\![\widetilde{N}]\!]_\tau$ respectively. The sketches are generated using the same set of hash function $h_0, ..., h_{k-1}$. The auxiliary inputs include the security parameter λ, the sketch size k.

Outputs $\mathcal{P}_1, ..., \mathcal{P}_\tau$ obtain the shares $[\![\widetilde{I}]\!]_1, ..., [\![\widetilde{I}]\!]_\tau$ respectively, where $\widetilde{I}$ is the estimated intersection cardinality, i.e. $[\![\widetilde{I}]\!]_1 + \cdots + [\![\widetilde{I}]\!]_\tau = \widetilde{I} = \frac{\widetilde{J} \cdot \widetilde{N}}{2k}$.

1) For $0 \leq i \leq k - 1$, $\mathcal{P}_1, ..., \mathcal{P}_\tau$ run the equality test protocol to compute $eq(M_{S_1}^{min}[i], \dots, M_{S_\tau}^{min}[i])$ and $eq(M_{S_1}^{max}[i], \dots, M_{S_\tau}^{max}[i])$. The two equality test results are output as shares to each party and the party can sum the shares locally. At the end of each iteration, each party holds a share $[\![t^{\langle i \rangle}]\!]_1, ..., [\![t^{\langle i \rangle}]\!]_\tau$ of the result of $eq(M_{S_1}^{min}[i], \dots, M_{S_\tau}^{min}[i]) + eq(M_{S_1}^{max}[i], \dots, M_{S_\tau}^{max}[i])$.
2) For each $\mathcal{P}_j$, the party computes the sum of shares locally $[\![u]\!]_j = \sum_{i=0}^{k-1} [\![t^{\langle i \rangle}]\!]_j$. Each $[\![u]\!]_j$ is a share of

$$u = \sum_{i=0}^{k-1} (eq(M_{S_1}^{min}[i], \dots, M_{S_\tau}^{min}[i]) + eq(M_{S_1}^{max}[i], \dots, M_{S_\tau}^{max}[i]))$$

3) $\mathcal{P}_1, ..., \mathcal{P}_\tau$ run the multiplication protocol and the floating point division protocol to compute $\widetilde{I} = \frac{u \cdot \widetilde{N}}{2k}$, using their shares $[\![u]\!]_1, ..., [\![u]\!]_\tau$ and $[\![\widetilde{N}]\!]_1, ..., [\![\widetilde{N}]\!]_\tau$. The output are shares $[\![\widetilde{I}]\!]_1, ..., [\![\widetilde{I}]\!]_\tau$.

Note if the cardinality is the final output, then in the last step the multiparty division can be omitted. The parties can compute just $u \cdot \widetilde{N}$ using their shares and output the result. Then each party can locally compute $\frac{u \cdot \widetilde{N}}{2k}$ since k is public. This makes the protocol more efficient.

6 Performance Evaluation

In this section, we show performance figures for our two-party protocol. We implemented the two-party protocol. We did not implement the multiparty protocols because the performance relies largely on the implementation of the underlying secret sharing-based multiparty secure computation framework. Our prototype is written in C and uses TCP sockets for communication between two distributed parties. We used OpenSSL for the underlying cryptographic operations. We implemented the OT extension protocol in [36] with the C-OT and R-OT optimizations from [37]. The base OT protocol is the Naor-Pinkas OT [34]. In all experiments, we set the security parameter to 128 and chose key size and cryptographic functions accordingly as recommended by NIST [66]. This should provide adequate security for most applications in median and long term (2031 and beyond) [66]. All experiments were run on two commodity computers: party 1 ran on a Ubuntu PC with an Intel Core i7 3.4 GHz CPU (i7-3770) and 8 GB RAM, party 2 ran on a Macbook pro (2011) with an Intel Core i7 2.2 GHz CPU (i7-2720QM) and 16 GB RAM. Switching computers for the parties did not cause significant difference in performance. The two computers are connected by switched 1 Gbit Ethernet. Our prototype is single-threaded, although the computation is fairly easy to parallelize.

We first show the accuracy of the estimates obtained from FM sketches. The sketches were built using Murmurhash 3 that has been widely used in large systems like Hadoop and Cassandra. Our initial tests showed that the difference in accuracy using sketches built from Murmurhash and SHA-1 is negligible and Murmurhash is much faster (3.3 ns per hash) than SHA-1 (170 ns per hash). We tested with sets of random 64-bit integers whose union cardinality ranges from 10 to 10^6 (1 million). For each union size, we tested with different $m = 4096, 16384$, and 65536 to guarantee that the relative error does not exceed $\epsilon = 4\%, 2\%$ and 1% respectively with a probability at least $1 - \delta = 0.999$ [4]. All experiments were repeated 100 times. Figure 5 shows the mean and the ranges of the estimation errors measured from our experiments[5]. The accuracy is good for both small and large sets. In all cases the mean of estimation error is less than 0.3% and falls within the desired range ($\pm 4\%$, $\pm 2\%$, and $\pm 1\%$). For extremely small sets (union cardinality = 10), we observed no errors. This shows that the formula with correction (Equation 1) is very effective.

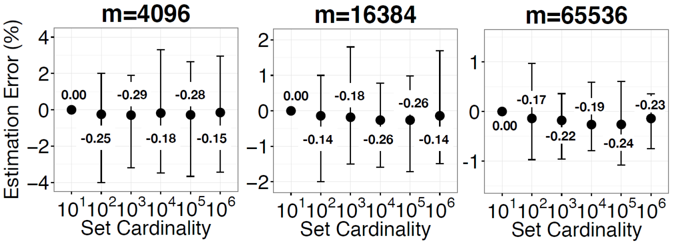


Fig. 5: Mean and range for estimation error $\frac{N - \widetilde{N}}{N}$. For $m = 4096, 16384, 65536$, the estimation error is in [-4%,4%], [-2%, 2%], [-1%,1%] with probability 0.999.

Then we show the pre-computation performance, i.e. the time for generating the FM sketches. The result is shown in Table 1. In the experiment, we set $w = 24$ and used random sets with different cardinalities from 10 to 10^6. For each set, we measured the time for generating $m = 4096, 16384$ and 65536 sketches from it. Note that sketch generation does not require cryptographic operations. Thus the sketches can be generated once and reused many times. This is often not possible in protocols that take sets as input. Although parties may be able to encrypt the sets before engaging in such protocols, the encrypted sets cannot be re-used because fresh randomness is needed to keep the protocol secure. Our pre-computation is also different from offline computation in some protocols that generate data independent values, which will be consumed in protocol execution and need to be regenerated for each protocol execution. We consider pre-computation as a one-off cost and do not include it in the protocol running time that will be shown later.

Card. \ **m**	4096	16384	65536
10	1.53×10^{-4}	6.42×10^{-4}	2.61×10^{-3}
10^2	1.40×10^{-3}	6.01×10^{-4}	2.32×10^{-2}
10^3	1.37×10^{-2}	5.62×10^{-2}	2.25×10^{-1}
10^4	1.32×10^{-1}	5.58×10^{-1}	2.21
10^5	1.26	5.50	2.19×10^{1}
10^6	1.25×10^{1}	5.45×10^{1}	2.19×10^{2}

TABLE 1: Performance: FM sketches generation (in seconds).

Next, we compare the performance of our protocol to existing protocols. The result is shown in Table 2. All numbers in the table are obtained by averaging 100 executions, except for the test of the exact protocol with 10^6-element private sets which was too slow (an execution took about 1 hour). The protocols we compared to are the two-party exact PSI-CA protocol in [11] and the two-party approximate PSU-CA protocol in [15], the state-of-the-art in each kind. We implemented these two protocols in C and use OpenSSL for the cryptographic operations. For the ease of implementation, we did not implement socket communication for these two protocols. Instead, we simply ran both parties on the Linux PC and let the two parties communicate through shared memory. This clearly favors [11] and [15] in terms of running time. In the experiment, both parties' private sets have the same cardinality. The running time of the approximate protocol in [15] does not include the time for building Bloom filters (as they can be pre-computed). Adding this time, as well as the times from Table 1 for our protocol, does not change the results qualitatively (our protocol is still faster). In the experiment, we use random sets with cardinality ranging from 10 to 10^6. For our protocol we set $w = \lceil \log(N) \rceil + 4$, so that the sketches are large enough for the cardinality N, e.g. $w = 24$ for $N = 10^6$. As we can see in Table 2, for small sets (cardinality in the order of 10^2 or less), the exact protocol in [11] is a better choice. But for larger sets (cardinality $\geq 10^3$) which are commonly encountered in PPDM, the approximate protocols are better and the difference becomes larger when the sets get larger. We tested with $m = 4096, 16384$ and 65536 so that the relative error bounds are $\epsilon = 1\%, 2\%$ and 4% respectively ($\delta = 0.001$). In all cases, the performance of our protocol is much better than the protocol in [15]. The difference is about 1 - 2 orders of magnitude.

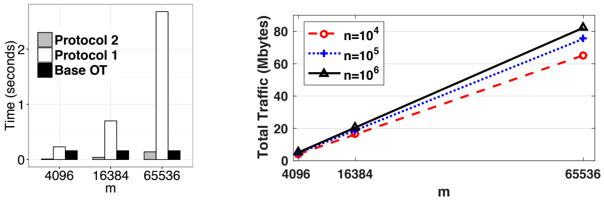
In Figure 6a, we show the break down of the running time of our protocol (cardinality $= 10^6$). For bootstrapping the OT extension scheme, we need λ base OT where λ is the security parameter. In the experiment $\lambda = 128$. The base OT cost is about 0.16 second. Recall that if the protocol needs to run multiple times, the base OT only needs to run once and its cost can be amortized. Most of the time is spent on the m iterations of Protocol

[4] The m values are slightly larger than the required values for the (ϵ, δ) pairs calculated from the formula in Theorem 1. We rounded them to the nearest powers of 2 for the ease of implementation.

[5] Appendix E includes histograms showing more detailed error distributions.

Cardinality		10	10^2	10^3	10^4	10^5	10^6
[11] (exact)		0.035	0.35	3.54	35.35	356.93	3507.38
$\epsilon = 4\%$	[15] (approximate)	0.65	0.71	1.50	6.92	45.09	337.51
	Ours ($m = 4096$)	0.29	0.31	0.33	0.35	0.38	0.40
$\epsilon = 2\%$	[15] (approximate)	2.33	2.42	3.37	10.19	56.72	397.13
	Ours ($m = 16384$)	0.46	0.54	0.62	0.73	0.82	0.89
$\epsilon = 1\%$	[15] (approximate)	9.06	9.17	10.22	18.69	77.06	488.48
	Ours ($m = 65536$)	1.17	1.51	1.84	2.29	2.64	2.97

TABLE 2: Performance comparison: computation time (in seconds, $\delta = 0.001$ for approximate protocols)



(a)

(b)

Fig. 6: PSU-CA protocol: (a) computation time breakdown, and (b) communication cost.

# of pairs	10^3	10^4	10^5	10^6
Non-fixed Key	0.28	3.90	44.50	489.638
Fixed Key	0.009	0.14	1.23	9.92
Improvement	28×	28×	36×	49×

TABLE 5: Fixed-key vs non-fixed key masking (time in ms)

1 to extract the estimators. After that, the cardinality estimation which involves one $\binom{1}{q}$-OT is very fast. We also measured the communication cost with different m and cardinalities. The result is shown in Figure 6b. With the smallest parameters (cardinality $= 10^4$ and $m = 4096$) the cost is about 4 MB, and with the largest parameters (cardinality $= 10^6$ and $m = 65536$) the cost is about 82 MB.

Last we show the performance of the fixed-key masking in Table 5. We compare it against non-fixed masking. The experiment ran on the Linux PC. As we can see the fixed-key masking scheme is more than one order of magnitude faster. In our implementation, we used the AES-NI set indirectly through the OpenSSL EVP engine. The performance can be further improved by writing code that directly uses the AES instructions.

7 Discussion

A limitation of our protocols is that they are not suitable for applications requiring exact computation of the cardinality of set union or intersection. However, in PPDM, this is often not a problem, as long as the approximation accuracy can be tuned according to the requirements of the PPDM algorithm. Firstly, data mining is about extracting useful information from data such as finding frequent patterns or finding similar regions or clustering the data. To some degree approximation is inherent in all mining algorithms that seek to approximate the unknown ground truth from data. Secondly, real world data is never perfect. For example past research suggested that the average error rate of a dataset in a data mining application is around 5%-10% [67], [68]. Although pre-processing can remedy the problem, it cannot completely remove the noise. To ensure robustness of the mining results, data mining algorithms are often designed to tolerate noise to some level. Thirdly, in cases such as mining extremely large data and data streams, approximation is more commonly used than exact algorithms. In these cases, computing exact answers is usually not possible because of limited computational resources such as RAM, CPU and disk space. Fourthly, as we have already mentioned, exact PPDM protocols often fail to deliver timely answers due to their huge computational cost. With time constraints, approximate but timely answers are often preferable.

8 Conclusion and Future Work

The secure computation of the union or intersection cardinality of sets belonging to different parties is a fundamental primitive in PPDM. However, the existing protocols are too inefficient for practical use in PPDM and may cause unwanted information leakage when used as subroutines. Thus, in this paper, we proposed novel protocols for the PSU-CA/PSI-CA problems. Our two-party protocols are very efficient and accurate, substantially outperforming the existing state-of-the-art protocols as shown in our experimental evaluation. The protocols compute the resulting PSI-CA and PSU-CA in a secret-shared form before disclosing them, which makes them more flexible and thereby more suitable for PPDM. The protocols can be extended to multiparty settings while retaining the good properties. A by-product of the protocol optimization is a fixed-key masking scheme that can significantly speed up $\binom{1}{n}$-OT when n is large.

Efficiency and scalability are already big challenges for data mining in the clear, and even bigger challenges for PPDM that requires more computation on the data in order to preserve data privacy. To this end, we would like to investigate the following directions: (1) protocols with sub-linear complexities which would greatly improve the efficiency; (2) protocols that are secure in a concurrently composable model. In this paper the protocols guarantee sequential composability. This could be a limitation because parallelization is another key tool to improve scalability and concurrent composability is necessary to ensure security in parallel protocol executions.

List of Appendices

The appendices of this paper are downloadable as supplementary material at http://ieeexplore.ieee.org. The appendices include:

A Proof of Theorem 1
B Security Analysis of the Two-party PSU-CA protocol (and its sub-protocols)
C Security of Fixed-key masking
D Optimized OT Extension in Our Protocols
E Estimation Error Distribution

Acknowledgements

We would like to thank the anonymous reviewers. The first author is supported in part by an EPSRC research grant (EP/M013561/2).

References

[1] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *Journal of Computer Security*, vol. 13, no. 4, pp. 593–622, 2005.

[2] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *TKDD*, vol. 2, no. 3, 2008.

[3] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving SVM classification on vertically partitioned data," in *PAKDD*, 2006, pp. 647–656.

[4] B. Liu and U. Hengartner, "Privacy-preserving social recommendations in geosocial networks," in *PST*, 2013, pp. 69–76.

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

[6] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *STOC*, 1987, pp. 218–229.

[7] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving data mining," *SIGKDD Explorations*, vol. 4, no. 2, pp. 28–34, 2002.

[8] R. Agrawal, A. V. Evfimievski, and R. Srikant, "Information sharing across private databases," in *SIGMOD*, 2003, pp. 86–97.

[9] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *EUROCRYPT*, 2004, pp. 1–19.

[10] L. Kissner and D. X. Song, "Privacy-preserving set operations," in *CRYPTO*, 2005, pp. 241–257.

[11] E. D. Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," in *CANS*, 2012, pp. 218–231.

[12] A. Davidson and C. Cid, "Computing private set operations with linear complexities," *IACR Cryptology ePrint Archive*, vol. 2016, p. 108, 2016. [Online]. Available: http://eprint.iacr.org/2016/108

[13] V. G. Ashok and R. Mukkamala, "A scalable and efficient privacy preserving global itemset support approximation using bloom filters," in *DBSec*, 2014, pp. 382–389.

[14] C. Blundo, E. D. Cristofaro, and P. Gasti, "Espresso: Efficient privacy-preserving evaluation of sample set similarity," *Journal of Computer Security*, vol. 22, no. 3, pp. 355–381, 2014.

[15] R. Egert, M. Fischlin, D. Gens, S. Jacob, M. Senker, and J. Tillmanns, "Privately computing set-union and set-intersection cardinality via bloom filters," in *ACISP*, 2015, pp. 413–430.

[16] D. Hand, P. Smyth, and H. Mannila, *Principles of Data Mining*. MIT Press, 2001.

[17] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press, 2011.

[18] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *J. Comput. Syst. Sci.*, vol. 31, no. 2, pp. 182–209, 1985.

[19] D. Demmler, T. Schneider, and M. Zohner, "ABY - A framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[20] W. Henecka, S. Kögl, A. Sadeghi, T. Schneider, and I. Wehrenberg, "TASTY: tool for automating secure two-party computations," in *ACM CCS*, 2010, pp. 451–462.

[21] G. Couteau, T. Peters, and D. Pointcheval, "Encryption switching protocols," in *CRYPTO*, 2016, pp. 308–338.

[22] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *IEEE Symposium on Security and Privacy*, 2013, pp. 478–492.

[23] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *CRYPTO*, 2000, pp. 36–54.

[24] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *SIGMOD*, 2000, pp. 439–450.

[25] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," in *STOC*, 1998, pp. 327–336.

[26] J. Ji, J. Li, S. Yan, Q. Tian, and B. Zhang, "Min-max hash for jaccard similarity," in *ICDM*, 2013, pp. 301–309.

[27] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *ACM CCS*, 2013, pp. 789–800.

[28] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *USENIX Security*, 2014, pp. 797–812.

[29] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *USENIX Security*, 2015, pp. 515–530.

[30] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on OT extension," *IACR Cryptology ePrint Archive*, vol. 2016, p. 930, 2016. [Online]. Available: http://eprint.iacr.org/2016/930

[31] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *NDSS*, 2012.

[32] M. O. Rabin, "How to exchange secrets by oblivious transfer," *Technical Report TR-81, Harvard Aiken Computation Laboratory*, 1981.

[33] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.

[34] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *STOC*, 1999, pp. 245–254.

[35] D. Beaver, "Correlated pseudorandomness and the complexity of private computations," in *STOC*, 1996, pp. 479–488.

[36] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *CRYPTO*, 2003, pp. 145–161.

[37] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *ACM CCS*, 2013, pp. 535–548.

[38] V. Kolesnikov and R. Kumaresan, "Improved OT extension for transferring short secrets," in *CRYPTO*, 2013, pp. 54–70.

[39] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer extensions with security for malicious adversaries," in *EUROCRYPT*, 2015, pp. 673–701.

[40] M. Keller, E. Orsini, and P. Scholl, "Actively secure OT extension with optimal overhead," in *CRYPTO*, 2015, pp. 724–741.

[41] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, 1979.

[42] O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[43] B. Scheuermann and M. Mauve, "Near-optimal compression of probabilistic counting sketches for networking applications," in *DIALM-POMC*, 2007.

[44] K. S. Beyer, R. Gemulla, P. J. Haas, B. Reinwald, and Y. Sismanis, "Distinct-value synopses for multiset operations," *Commun. ACM*, vol. 52, no. 10, pp. 87–95, 2009.

[45] J. Boyar and R. Peralta, "The exact multiplicative complexity of the hamming weight function," *Electronic Colloquium on Computational Complexity (ECCC)*, no. 049, 2005.

[46] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *ICALP 2008*, 2008, pp. 486–498.

[47] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols (extended abstract)," in *STOC*, 1990, pp. 503–513.

[48] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *EC*, 1999, pp. 129–139.

[49] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole - reducing data transfer in garbled circuits using half gates," in *EUROCRYPT*, 2015, pp. 220–250.

[50] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steele, "Secure computation on floating point numbers," in *NDSS*, 2013.

[51] L. Kamm and J. Willemson, "Secure floating point arithmetic and private satellite collision analysis," *Int. J. Inf. Sec.*, vol. 14, no. 6, pp. 531–548, 2015.

[52] P. Pullonen and S. Siim, "Combining secret sharing and garbled circuits for efficient private IEEE 754 floating-point computations," in *WAHC*, 2015, pp. 172–183.

[53] D. Demmler, G. Dessouky, F. Koushanfar, A. Sadeghi, T. Schneider, and S. Zeitouni, "Automated synthesis of optimized circuits for secure computation," in *ACM CCS*, 2015, pp. 1504–1517.

[54] S. Gueron, "Intel advanced encryption standard (AES) new instructions set," Intel, Tech. Rep., 2012.

[55] P. Rogaway and J. P. Steinberger, "Constructing cryptographic hash functions from fixed-key blockciphers," in *CRYPTO*, 2008, pp. 433–450.

[56] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious PRF with applications to private set intersection," in *ACM CCS*, 2016, pp. 818–829.

[57] S. Michel and T. Neumann, "Search for the best but expect the worst - distributed top-k queries over decreasing aggregated scores," in *WebDB*, 2007.

[58] M. Kamp, C. Kopp, M. Mock, M. Boley, and M. May, "Privacy-preserving mobility monitoring using sketches of stationary sensor readings," in *ECML/PKDD*, 2013, pp. 370–386.

[59] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *ESORICS*, 2008, pp. 192–206.

[60] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen, "Asynchronous multiparty computation: Theory and implementation," in *PKC*, 2009, pp. 160–179.

[61] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *CRYPTO 2012*, 2012, pp. 643–662.

[62] J. Launchbury, I. S. Diatchki, T. DuBuisson, and A. Adams-Moran, "Efficient lookup-table protocol in secure multiparty computation," in *ICFP*, 2012, pp. 189–200.

[63] R. Canetti, "Security and composition of multiparty cryptographic protocols," *J. Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

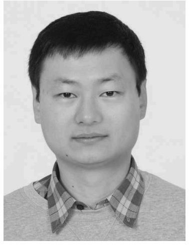
[64] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *PKC*, 2007, pp. 343–360.

[65] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *CRYPTO*, 1991, pp. 420–432.

[66] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management - part 1: General (revision 3)," NIST, Tech. Rep. SP 800-57, July 2012.

[67] K. Orr, "Data quality and system theory," *Commun. ACM*, vol. 41, no. 2, pp. 66–71, 1998.

[68] J. I. Maletic and A. Marcus, "Data cleansing: Beyond integrity analysis," in *IQ*, 2000, pp. 200–209.



Changyu Dong holds a PhD from Imperial College London. He is currently a senior lecturer at the School of Computing Science, Newcastle University. His research interests include applied cryptography, trust management, data privacy and security policies. His recent work focuses mostly on designing practical secure computation protocols. The application domains include for example privacy preserving medical data analytics, secure cloud computing, privacy preserving data mining and private data processing in mobile and wireless networks. He has over 30 publications in international journals and conferences.

Grigorios Loukides is an Assistant Professor (Lecturer) in the Department of Informatics, King's College London. Prior to that, he was a Royal Academy of Engineering Research Fellow (2011-2016). His research interests are in data mining, with a focus on privacy-preserving data mining and social network optimization. He has published over 60 papers on these topics, including in top venues (KDD, ICDM, SDM).