Large-Scale Multi-Party Counting Set Intersection using a Space Efficient Global Synopsis*

Dimitrios Karapiperis,[1] Dinusha Vatsalan,[2]
Vassilios S. Verykios,[1] and Peter Christen[2]

[1] School of Science and Technology, Hellenic Open University, Patras, Greece
dkarapiperis@eap.gr, verykios@eap.gr
[2] Research School of Computer Science, The Australian National University, Canberra ACT 0200, Australia
dinusha.vatsalan@anu.edu.au, peter.christen@anu.edu.au

Abstract. Privacy-preserving set intersection (PPSI) of very large data sets is increasingly being required in many real application areas including health-care, national security, and law enforcement. Various techniques have been developed to address this problem, where the majority of them rely on computationally expensive cryptographic techniques. Moreover, conventional data structures cannot be used efficiently for providing count estimates of the elements of the intersection of very large data sets. We consider the problem of efficient PPSI by integrating sets from multiple (three or more) sources in order to create a global synopsis which is the result of the intersection of efficient data structures, known as Count-Min sketches. This global synopsis furthermore provides count estimates of the intersected elements. We propose two protocols for the creation of this global synopsis which are based on homomorphic computations, a secure distributed summation scheme, and a symmetric noise addition technique. Experiments conducted on large synthetic and real data sets show the efficiency and accuracy of our protocols, while at the same time privacy under the Honest-but-Curious model is preserved.

1 Introduction

Computing set operations, such as intersection, union, equi-join and disjointness, efficiently and privately among different parties is an important task in privacy-preserving data mining [3,9]. In this paper, we study the problem of privacy-preserving set intersection (PPSI), which is also known as private data matching [15], of multi-sets of an arbitrary large number of distinct elements held by three or more parties. Growing privacy concerns and government laws preclude the exchange of sensitive and private values stored in databases across different organizations for calculating the intersection of those private values.

* This research was partially funded by the Australian Research Council under Discovery Project DP130101801.

This has led to an active research area, known as PPSI, in the field of privacy-preserving data mining [3] and specifically in private data matching [19,35].

PPSI is useful in many real-world applications, ranging from health-care, crime detection, national security, to finance and business. An example motivating application would be a health surveillance system, where by monitoring drug consumption at pharmacies or hospitals located at different places, alerts could be issued whenever consumption of certain drugs exceeds a threshold at all or some of these hospitals. A crime detection or national security application could be the monitoring of the number of times certain on-line services are accessed, by applying the intersection operation on requests made to these on-line services from different Internet Service Providers (ISPs). These examples illustrate that often large sets of sensitive elements held by different parties (or organizations) need to be intersected so that a set of common elements, accompanied by their counts of occurrences, can be identified. However, privacy and confidentiality concerns, as well as other business regulations, commonly prevent the sharing and exchange of such private values across several parties.

There have been several solutions proposed in the literature addressing the problem of PPSI. Most of them are either based on Secure Multi-Party computation (SMC) [27] techniques that are computationally expensive and thus are not scalable to large sizes of sets and larger number of parties, or they only perform intersection of two sets (from two parties). Moreover, in the applications described above, we are not only interested in learning the set of intersection of elements but also their number of occurrences. This problem cannot be efficiently solved by conventional data structures, such as hash tables or vectors, due to the large number of distinct elements that need to be monitored.

In this paper, we propose the creation of a privacy-preserving global synopsis by integrating data from many sources and attaining the common elements. More specifically, we tackle the challenge of identifying the counts of these elements from a potentially very large multi-set as they occur. Each party independently summarizes its elements in a local synopsis, which is implemented by a Count-Min sketch [12], and then these local synopses are intersected in order to create the global synopsis. This global synopsis (a) provides collective count estimates for the common elements attained and (b) hides the contribution of each party to these estimates. We propose and evaluate two protocols for the creation of this global synopsis:

1. the first protocol, which relies on homomorphic operations, exhibits accurate and reliable results but adds high communication cost. The number of homomorphic operations has a logarithmic relation to the total number of occurrences of elements in the sets to be intersected.
2. the second protocol relies completely on simple secure computations, exhibiting high performance and simultaneously highly accurate results.

The remainder of this paper is structured as follows. We next review related work in Sect. 2. In Sect. 3, we formulate the problem to be addressed, and in Sect. 4 we describe the building blocks used for creating a privacy-preserving global synopsis. Then, we present our protocols for the creation of the global

synopsis in detail in Sect. 5, while in Sect. 6 we empirically evaluate our protocols using both synthetic and real data sets. Finally, in Sect. 7 we summarize our work and discuss directions for future work.

2 Related Work

Various techniques have been developed addressing the PPSI problem over the past decades. Most of the solutions proposed so far rely on general SMC-based cryptographic techniques. General two-party secure computation was introduced by Yao [36] and extended to multi-parties by Goldreich et al. [18].

Agrawal et al. [4] developed two-party protocols based on SMC commutative encryption schemes for three set operations: intersection, intersection size, and equi-join. The protocols allow for information integration with minimal data sharing. However, they are expensive in terms of computation and communication complexity. Freedman et al. [15] proposed two-party PPSI protocols based on homomorphic encryption and balanced hashing for both the semi-honest and the malicious adversary models. In their work, the sets are represented as roots of polynomials. This work was extended by Kissner et al. [24], who utilize the power of polynomial representation of multi-sets for PPSI.

Hazay and Lindell [20] adopted a pseudo-random-function-based solution for the two-party PPSI problem, which can be used either for one malicious and one semi-honest party or for two covert parties [5]. Dachman-Soled et al. [13] addressed the problem of PPSI for two malicious parties using homomorphic encryption and polynomial functions. In addition, several approaches have been proposed on variants of the PPSI problem, such as privacy-preserving union [16], privacy-preserving equality test [30], or privacy-preserving disjointness [23]. These works also employ SMC-based privacy techniques, which makes the solutions not efficient and scalable to large sets held by multiple parties.

In order to overcome the drawback of high computational overhead with SMC-based techniques, privacy-preserving set operations, which rely on efficient privacy techniques such as Bloom filter-based encoding, have recently being investigated in the areas of privacy-preserving record linkage [35] and privacy-preserving data mining [3]. Lai et al. [32] proposed an efficient PPSI on multiple sets using Bloom filters. The parties distributively compute a conjuncted Bloom filter by applying the logical AND operation on their partitions and then each party checks its elements with this conjuncted Bloom filter in order to determine if they are in the intersection set. A similar approach using Counting Bloom filters was proposed by Many et al. [28]. Dong et al. [14] introduced an efficient PPSI protocol between two sets using Garbled Bloom filters (GBfs) and an oblivious transfer (OT) protocol. First, the participating parties encode their sets as GBFs and then run the OT protocol in order to obtain the intersection set. A private equi-join approach on multiple databases was presented by Kantarcioglu et al. [22], where a secure equi-join is applied on k-anonymized databases.

Roughan and Zhang [33] proposed an efficient private set union solution by using Count-Min sketches [12] in order to collect Internet-wide statistics.

Charikar et al. [8] proposed the Count-Sketch data structure, which was adjusted for self-join size estimation by Cormode and Garofalakis [11]. However, space requirements of Count sketches are far higher than those of Count-Min sketches [12], making them less suitable for large-scale applications.

3 Problem Formulation

Let us suppose a distributed environment, which consists of m parties, each denoted by p_i, where $i = 1, \ldots, m$, and a global authority G, which plays the role of a central public semi-trusted regulatory agency. Each p_i should monitor the number of occurrences of an element e_j (like an IP address, a drug, or the registration plate of a car), where $j = 1, \ldots, n$, and n is a large number possible in the tens or even hundreds of millions. By doing so, a local summary S is built by each p_i, which includes the total number of occurrences of each e_j for a certain time period, or for a specified total number of e_js monitored, denoted by N, where each distinct e_j might appear several times. Therefore, N is equal to $\sum_{j=1}^{n} V(e_j)$, where $V(\cdot)$ returns the exact number of occurrences of an e_j. An update operation is required when an e_j should be monitored by a p_i, which should increase the number of occurrences of this e_j in the local summary. A query operation is also needed, which should return the current number of occurrences of an e_j. Authority G, by exploiting the local summaries, should answer collective queries regarding the number of occurrences of each e_j above a specified threshold θ, as monitored globally by every p_i. In essence, these e_js constitute the intersection set $\mathcal{S}$, defined formally as $\mathcal{S} = \{e_j | e_j \in p_1 \wedge \ldots \wedge e_j \in p_m\}$.

Such a query could be "How many times ($> \theta$) has a certain web site been accessed by all p_is?". All summaries should be collected by G on a frequent basis, so that G can produce an almost real-time global summary. Moreover, each p_i should maintain its summary in main memory to allow fast updating, when an e_j is monitored. Therefore, the size of the data structure used to realize each summary should be as small as possible, although the number of the e_js can be large. The reluctance of some p_is to disseminate their summaries due to privacy concerns is an additional problem. For instance, a hospital might be a reluctant p_i not willing to share any medical information, such as the drugs consumed which are the e_js in this case, in order to protect the privacy of its patients.

The solution of building each summary, by utilizing a vector, where each position represents a distinct element and holds the number of its occurrences, is prohibitive due to the size of the summary which will grow linearly with the number of elements represented. Also, by using a vector, the size and the update time for each element is $\mathcal{O}(n)$. By utilizing a hash table, we can achieve $\mathcal{O}(1)$ update time but the size required remains the same as by using a vector. For this reason, each p_i creates a local synopsis, denoted by S_i, which is a specialized sketching data structure [12] that consumes a fixed amount of space in main memory regardless of the number of e_js represented, at the cost of an allowable configurable error. After all the S_is are created, a global synopsis, denoted by

GS, should be generated by performing the intersection operation among the S_is. The GS produced should provide collective count estimates for all the elements of the intersection set $\mathcal{S}$. Privacy of each p_i should be protected, so that it is not possible to infer the contribution of each p_i to these collective count estimates. Formally, given a collective count estimate for an e_j ($e_j \in \mathcal{S}$) one cannot make any inferences or estimates regarding each $V_{p_i}(e_j)$, where $V_{p_i}(\cdot)$ denotes the exact number of occurrences for an e_j at a certain party p_i.

4 Background

In this section, we give a brief outline of the building blocks utilized in order to create the privacy-preserving global synopsis.

4.1 Creating a Local Synopsis

An efficient way for creating a local synopsis is by using a Count-Min sketch [12] (sketch). The main feature of a sketch is that it utilizes space that is sublinear with the number of e_js represented by it. A sketch is an array that consists of D rows and W cells in each row, initialized to 0. Both D and W are specified later. In order to update an e_j in a sketch, D hash operations are performed by randomly chosen, pairwise independent hash functions of the form $h_d(e_j) = [(a_d e_j + b_d) \mod P] \mod W$, where $d = 1, \ldots, D$, P is a large prime number (e.g., $2^{31} - 1$) greater than n, and each a_d, b_d are randomly chosen integers from $(0, P)$. Thus, there are D hash results, each corresponding to a cell in each row, where its value is incremented by 1, namely $S[d][h_d(e_j)] = S[d][h_d(e_j)] + 1$, for each of the $d \in D$ rows. The query operation $query(S, e_j)$ returns the count estimate of an e_j by hashing this e_j using the same hash functions as in the update operation and then picking the minimum hash value.

An interesting property is the linearity of sketches; the sketch produced by adding cell-wise two sketches (both built by using the same hash functions) is the union of these two sketches. This property makes sketches particularly useful because collective count estimates can be provided in distributed environments. By using sketches, we can detect frequent elements, such as IPs flooding in a network or drugs consumption above a certain frequency of appearance, denoted by ϕ. These frequent elements, the so-called heavy hitters [12], may indicate a certain anomaly, which may require an immediate course of proactive actions. For example, over-consumption of certain drugs by all pharmacies state-wide may highlight an outbreak of an infectious disease. By setting $D = \lceil \ln(N/\delta) \rceil$ and $W = \lceil 1/\epsilon \rceil$ [12], heavy hitters which exhibit a number of occurrences more than a specified threshold[3] θ, are identified, where $\theta = \lceil \phi N \rceil$ with $0 < \phi < 1$. Simultaneously, any e_j occurring less than $\lceil (\phi - \epsilon)N \rceil$ times, where $\epsilon \ll \phi$ e.g., $\epsilon = \phi/10$, is ignored, with confidence $1 - \delta$.

[3] In the literature, the term threshold can also be found as *support*.

Algorithm 1 Secure distributed summation scheme.

Input: $x_1, \ldots, x_m, r, F$
Output: s // the summation returned
1: $q_1 \leftarrow (x_1 + r) \mod F$ // p_1 produces q_1
2: **for** $(i = 2, \ldots, m)$ **do**
3: $q_i \leftarrow (q_{i-1} + x_i) \mod F$ // p_i produces q_i by using q_{i-1}
4: **end for**
5: $s \leftarrow (q_m - r) \mod F$ // p_1 produces s which is the summation

4.2 Homomorphic Computations for Preserving Privacy

A reliable Secure Multi-Party (SMC) technique of performing a joint computation among several parties is the partially homomorphic Paillier cryptosystem [31]. A joint computation could be the addition of some values, where these values should remain secret due to privacy concerns. Successive encryption of the same value generates different cipher texts with high probability. A trusted authority is required in order to issue a public/private key pair, needed for the encryption and decryption operations respectively. Given two values (messages), x_1 and x_2, encryption is performed by using the public key and the produced cipher texts are denoted by $\widetilde{x}_1$ and $\widetilde{x}_2$ respectively. Given the cipher texts, we can perform either homomorphic addition ($\widetilde{x}_1 \oplus \widetilde{x}_2$) or multiplication with a constant c ($c \odot \widetilde{x}_1$). The cipher texts can be decrypted by the trusted authority by using its private key.

4.3 Secure Distributed Summation

A simple summation scheme, as introduced in [9], can be applied by the p_is in order to perform a joint summation. This scheme is much more efficient than the homomorphic approach described above. By using this scheme, each p_i masks the corresponding values in its S_i, such that if p_{i+1} obtains S_i it cannot reproduce the actual values of S_i. We illustrate the scheme by using a simple example. Each p_i has monitored e_1 x_i times. Then, the p_is should sum up jointly each x_i such that none of the x_is are disclosed to any p_i. First, p_1 generates a random number r, which lies in the interval $(0, F)$, where F is a large integer greater than the summation calculated. The steps of the scheme are illustrated in Algorithm 1. The number of parties m must be more than two, because otherwise x_2 will be revealed to p_1.

5 Protocols for Creating a Privacy-Preserving Intersection Global Synopsis

In the simple non privacy-preserving scenario, G performs the intersection operation, among the S_is, following a naive protocol where each p_i submits to G its corresponding S_i and then G merges them. This approach overwhelms G with

the S_is, imposing high computational overhead and network traffic. Additionally, since the hash functions are common for both updating an S_i and querying GS, G can easily perform an iterative query process to an S_i, for each possible e_j, and consequently find out sensitive information regarding a single p_i. In order to mitigate these concerns, the GS can also be generated by the p_is jointly, without the participation of G, as we will describe later.

The basic idea of the proposed protocols is the secure generation of the global synopsis GS', which holds the common e_js without the count estimates. More specifically, if a cell in a S_i contains a value which is below a specified threshold θ, then the GS' should indicate this and the corresponding cell in the GS should be set to 0. Thus, if we query GS for an e_j which is hashed to a cell of a row in GS holding 0, this will result in returning 0 as the global estimate. This happens because during the query operation of the GS, we choose the minimum value of the D estimates. By doing so, we can use GS' to easily exclude those cells in GS which correspond to elements not included in the intersection set.

The communication complexity of both protocols is linear in the number of parties, namely $\mathcal{O}(m)$. We make the assumption that the participating parties (p_is and G) follow the Honest-but-Curious (HBC) model [19,27], in that they follow the protocol steps while being curious to learn about other party's data. Furthermore, we assume that there is no collusion among them. Our protocols are secure in that the contribution of each p_i to the count estimates is hidden from both the rest of the p_is and from G. We utilize the secure distributed summation scheme combined with homomorphic operations in order to both deliver accurate results and to protect the privacy of the p_is in an efficient manner.

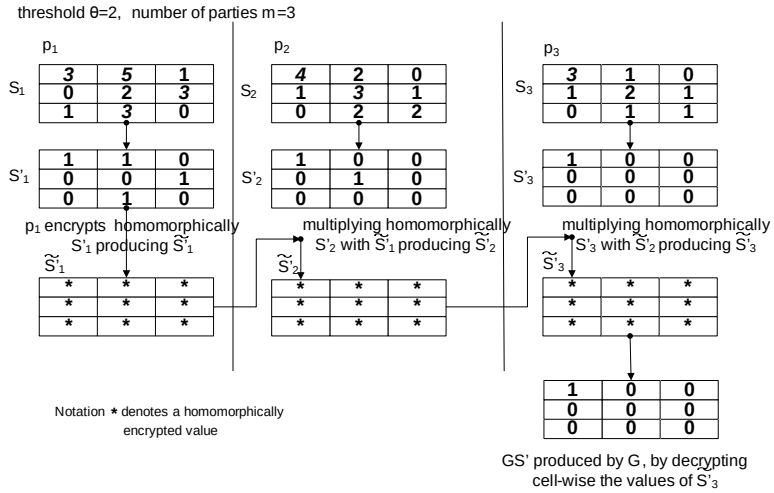


Fig. 1: Each $\widetilde{S}'_{i-1}$ is multiplied cell-wise by each S'_i. In the end, G decrypts $\widetilde{S}'_m$ and attains the common elements ($i = 2, \ldots, m$).

Table 1: The steps of the homomorphic protocol for creating a privacy-preserving intersection global synopsis.

step 1	G issues the pair of public/private keys needed for the homomorphic operations and sends out the public key to each p_i.
step 2	Each p_i by using its S_i, produces S'_i, by replacing the values above θ with 1 and those below θ with 0.
step 3a	p_1 encrypts homomorphically S'_1, producing $\widetilde{S}'_1$, which is submitted to p_2.
step 3b	p_1 produces R (one random value per cell) and then Q_1, which are $D \times W$ arrays and correspond to r and q_1 respectively depicted in Algorithm 1, needed for the secure cell-wise summation of each S_i. Q_1 is submitted to p_2.
step 4a	Each p_i $(i = 2, \ldots, m)$ performs cell-wise homomorphic multiplications between $\widetilde{S}'_{i-1}$ and S'_i, producing $\widetilde{S}'_i$, which is submitted to p_{i+1}.
step 4b	Each p_i $(i = 2, \ldots, m)$, by using Q_{i-1}, produces Q_i $(Q_i[d][w] = (Q_{i-1}[d][w] + S_i[d][w]) \mod F)$, which is submitted to p_{i+1}.
step 5a	p_m produces $\widetilde{S}'_m$ and then submits it to G.
step 5b	p_m produces Q_m and then submits it to p_1.
step 6	p_1 produces GU, by subtracting R from Q_m, which is submitted to G.
step 7a	G decrypts $\widetilde{S}'_m$ in order to produce GS'.
step 7b	G multiplies cell-wise GU with GS' in order to obtain GS.

5.1 The Homomorphic Protocol

The homomorphic protocol (HP) is two-fold; it identifies the common e_js and calculates the union of the S_is, by exploiting the linearity of sketches as explained in Sect. 4.1. Any cells which contain values above θ in all S_is are securely identified by performing homomorphic operations using encrypted data, as if we are using the initial plain values. First, each p_i produces a new synopsis, denoted by S'_i, by replacing the cells of the S_i which hold values above threshold θ with 1 and below it with 0. Then, p_1 encrypts S'_1 homomorphically, producing $\widetilde{S}'_1$, which is submitted to p_2. Following the protocol, each p_i $(i = 2, \ldots, m)$ performs cell-wise homomorphic multiplications between $\widetilde{S}'_{i-1}$ and S'_i, producing $\widetilde{S}'_i$, which is submitted to p_{i+1}, as shown in Fig. 1. This is necessary because if a cell in any S'_i contains 0, then GS' in this cell should contain 0, regardless whether there are other S'_is that contain 1 in this particular cell. By doing so, we identify the common e_js, as monitored by every p_i in a secure manner since each p_i cannot infer anything by inspecting $\widetilde{S}'_{i-1}$. If instead of multiplying we added cell-wise $\widetilde{S}'_{i-1}$ to $\widetilde{S}'_i$, we would create a GS' where each cell would contain the exact number of parties, which exceed θ. In this case, we can apply the rule of the minimum number of parties, where an $S'_i[d][w]$ should exceed θ in order to be included in the GS'. Synopsis GS' though cannot be used to provide count estimates because the values of its cells are equal to either 1 or 0. For this reason, we also obtain GU, by using the secure distributed summation scheme, which is the result of the union operation among the S_is. Finally, by multiplying cell-wise

Algorithm 2 Multiplying $\widetilde{S}'_{i-1}$ with S'_i, where $i = 2, \ldots, m$ (step 4a of the HP).

Input: $\widetilde{S}'_{i-1}$, S'_i
Output: $\widetilde{S}'_i$
1: **for** $(d = 1, \ldots, D)$ **do**
2: **for** $(w = 1, \ldots, W)$ **do**
3: $\widetilde{S}'_i[d][w] \leftarrow \widetilde{S}'_{i-1}[d][w] \odot S'_i[d][w]$
4: **end for**
5: **end for**

Algorithm 3 p_i produces Q_i by using Q_{i-1}, where $i = 2, \ldots, m$ (steps 4b and 2b of the HP and NBP, respectively.)

Input: S_i, Q_{i-1}, F
Output: Q_i
1: **for** $(d = 1, \ldots, D)$ **do**
2: **for** $(w = 1, \ldots, W)$ **do**
3: $Q_i[d][w] \leftarrow (Q_{i-1}[d][w] + S_i[d][w]) \mod F$
4: **end for**
5: **end for**

GU with GS', we obtain GS, which can be used to provide count estimates of the common e_js, realizing the result of the intersection operation among the underlying S_is. This protocol, as illustrated in Table 1, prevents G, or any p_i, from inferring any information from the intermediate synopses circulated. Algorithm 2 illustrates step 4a of the protocol where each p_i, by performing homomorphic multiplications between $\widetilde{S}'_{i-1}$ and S'_i, produces $\widetilde{S}'_i$ which is submitted to p_{i+1}. At each party, the number of homomorphic operations is $\mathcal{O}(D \times W)$ where D has a logarithmic dependency on N and W depends on ϵ regardless of N or n.

In Algorithm 3, it is shown how each p_i performs the secure distributed summation scheme cell-wise, by exploiting the linearity of the S_is (step 4b of the protocol). The main computational overhead of this protocol is the encryption of the S_1 in step 3a. Also, each $\widetilde{S}'_i$ adds high communication cost due to its size, which is proportional to the size of S'_i (or S_i), multiplied by a constant factor (e.g., 2), which depends on the implementation of the Paillier cryptosystem [31].

5.2 The Noise-Based Protocol

In the Noise-Based Protocol (NBP), instead of producing each $\widetilde{S}_i$ in order to generate the GS', we apply cell-wise the secure distributed summation scheme (see Algorithm 1). As shown in Algorithm 4, if an $S_i[d][w]$ contains a value above θ, then $Q'_i[d][w]$ becomes $Q'_{i-1}[d][w]$ plus 1. Otherwise, p_i assigns to $Q'_i[d][w]$ the value of $Q'_{i-1}[d][w]$ plus some symmetric noise (eg., a random value drawn from a Laplace or a Gaussian distribution, where the location and scale parameters are set to 0 and 1, respectively).

Algorithm 4 p_i produces Q'_i by using Q'_{i-1}, where $i = 2, \ldots, m$ (step 2a of the NBP).

Input: S_i, Q'_{i-1}, F, θ
Output: Q'_i

```
1: for (d = 1, ..., D) do
2:     for (w = 1, ..., W) do
3:         if S_i[d][w] > θ then
4:             Q'_i[d][w] ← (Q'_{i-1}[d][w] + 1) mod F
5:         else
6:             Q'_i[d][w] ← (Q'_{i-1}[d][w] + Lap(0,1)) mod F        // adding Laplace noise
7:         end if
8:     end for
9: end for
```

We add symmetric noise in order to sanitize $Q'_i[d][w]$ with respect to m. By doing so, p_1 is prevented from making any inferences, such as the exact number of parties which exceed θ, by inspecting the cells of the Q'_m after subtracting R'. By revealing the value of m for a cell, which is the case where all parties exceed θ in that cell, neither p_1 nor G learns anything that can breach the privacy of any party. Fig. 2 illustrates how three parties create the GS' by following the steps of this protocol. Finally, G receives two global synopses, namely GU and GS'. Synopsis GU is the result of the union operation among the S_is while GS' holds for each cell either m or a sanitized value, which is the result of the noise applied in the case where at least one cell in the same coordinates of any S_i contains an unacceptable value (below θ). Authority G, by checking cell-wise GS', identifies accurately which cells should be discarded from GU and sets them to 0. This protocol is illustrated in Table 2.

6 Evaluation

We evaluate our protocols in terms of the accuracy of the count estimates, the execution time, the space required, the precision, and the recall of the results, by using both synthetic and real data sets. For measuring accuracy, we specify the completeness measure C as:

$$C = 1 - \frac{\sum_{\forall e_j \in \mathcal{S}} \mid query(GS, e_j) - V(e_j) \mid}{\sum_{\forall e_j \in \mathcal{S}} V(e_j)}, \tag{1}$$

where $V(\cdot)$ returns the exact global number of occurrences of an e_j. The completeness measure shows the overall accuracy of the estimates, as compared with the exact global number of occurrences of the e_js. A value for completeness near 1 denotes high accuracy for the estimates provided by the corresponding GS. Recall is the number of the correct elements found as a percentage of the number of the truly correct elements. On the other hand, precision is the number of the

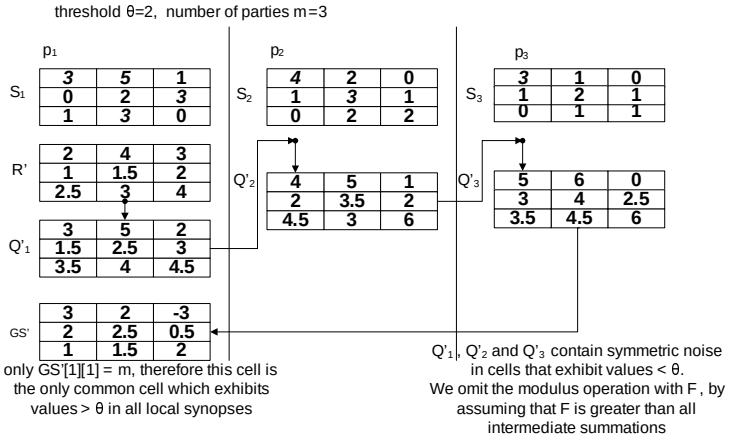


Fig. 2: In each cell of each Q'_i, we add either 1 or symmetric noise, depending on the corresponding value in each S_i $(i = 2, \ldots, m)$.

correct elements found as a percentage of the entire output. For our experiments, we have chosen as the application domain the monitoring and the identification of common web resources appearing at five local parties. All experiments were conducted on a Pentium Dual Core at 2GHz with 4GB RAM. The software components are developed using the Java programming language version 1.7, and are available from the authors.

We compare our protocols with the intersection operation included in the Sepia library presented in [7] and [28], where Counting Bloom filters (*CBf*s) are used, as initially introduced in [10]. A *CBf* is an integer array, where each e_j is hashed K times, by using the HMAC-MD5 hash function [26]. For each cell that an e_j is hashed to, we increment this cell's value by 1. In order to derive the count estimate of an e_j, we hash it and then take the minimum of the values retrieved. More specifically, we use the weighted intersection (I-SEPIA) of the Sepia library, where an e_j should be represented by the global *CBf* (*gCBf*) only if it has been monitored by every p_i. Each p_i submits to G two *CBf*s: A flag-based *CBf* that actually states if an e_j has been monitored by this p_i, and another *CBf* which holds the number of occurrences of each e_j. The flag-based *CBf*s are cell-wise multiplied producing $gCBf_1$, which includes the common e_js that should be represented by the *gCBf*. The *CBf*s holding the number of occurrences are cell-wise added producing $gCBf_2$, which holds the global summations. A straightforward cell-wise multiplication between these two *gCBf*s produces the final *gCBf*. The number of hash functions K and the size L of each *CBf* depend on the specified false-positive probability, denoted by *FPR*. This rate can be considered as the acceptable error rate, when a *CBf* provides a count estimate for an e_j which it has never been hashed to it. The specified error rate is achieved by setting $K = \ln(2)L/n$ and $L = cn$, where c is

Table 2: The steps of the noise-based protocol for creating a privacy-preserving intersection global synopsis.

step 1a	p_1 produces R' and Q'_1, which are $D \times W$ arrays and correspond to r and q_1 respectively depicted in Algorithm 1, but each cell of Q'_1 instead of holding the real values of S_1 holds 1 if $S_1[d][w] > \theta$ and some symmetric noise if not (plus the random values of R').
step 1b	p_1 produces R and then Q_1, both needed for the secure cell-wise summation of each S_i. Both Q_1 and Q'_1 are submitted to p_2.
step 2a	Each p_i $(i = 2, \ldots, m)$, by checking cell-wise each S_i, adds $Q'_{i-1}[d][w]+1$ to $Q'_i[d][w]$, only if $S_i[d][w]$ is above θ. Otherwise, $Q'_i[d][w]$ becomes $Q'_{i-1}[d][w]$ plus some symmetric noise.
step 2b	Each p_i $(i = 2, \ldots, m)$, by using Q_{i-1}, produces Q_i $(Q_i[d][w] = (Q_{i-1}[d][w] + S_i[d][w]) \mod F)$. Both Q_i and Q'_i are submitted to p_{i+1}.
step 3a	p_m produces Q'_m.
step 3b	p_m produces Q_m and then submits it to p_1 along with Q'_m.
step 4a	p_1 produces GS' by subtracting R' from Q'_m.
step 4b	p_1 produces GU, by subtracting R from Q_m. Both GU and GS' are submitted to G.
step 5	G checks cell-wise GS' and if $GS'[d][w] \neq m$, then $GU[d][w] = 0$. Finally, $GS = GU$.

a small constant in order to minimize the formula $(0.6185)^{L/n}$. An analysis of the derivation of these optimal values is given in [6].

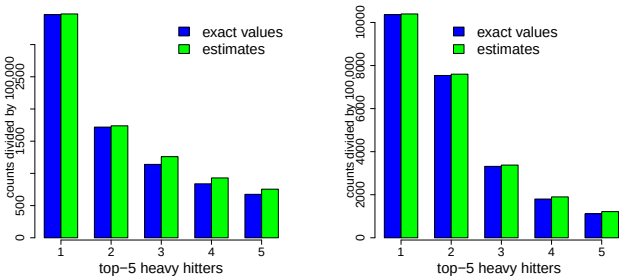
6.1 Evaluation using Synthetic Data Sets

Each party uses a synthetic data set where we generate $N = 10^9$ occurrences from an alphabet of $n = 10^6$ distinct elements. The parameters used for building the synopses are depicted in Table 3.

Table 3: The parameters used for building the synopses.

ϕ	ϵ	D	W	δ
0.1	0.01	26	100	0.01
0.01	0.001	26	1000	0.01
0.001	0.0001	26	10,000	0.01

Elements generated for each party follow the Zipf distribution, since there are numerous studies reporting that web resource popularity obeys power-law long-tailed distributions [2,17,21,25]. We set the skew parameter z to 1 (Fig. 3a) and to 2 (Fig. 3b) for higher skew. As shown in these figures, by setting $z = 2$, the differences between the exact values and the corresponding estimates



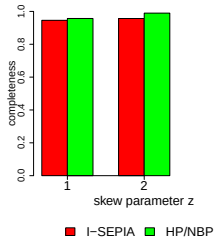
(a) $z = 1$ (b) $z = 2$

Fig. 3: Absolute estimates and exact values, by generating e_js, following the Zipf distribution.

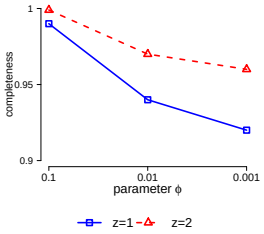
are almost eliminated, since the accuracy of Count-Min sketches is increased by using highly skewed data distributions [34].

For I-SEPIA, we set $FPR = 0.1$, which yields nearly 5×10^6 cells for the corresponding CBFs and the $gCBF$. In Fig. 4a, we illustrate the completeness rates, where our protocols outperform I-SEPIA. By setting higher skew ($z = 2$), the completeness rate of our protocols is even higher than the one of I-SEPIA. In Fig. 4b, we show the completeness rates for both protocols, where we observe that these rates are constantly above 0.9. Also, by setting skew to 2, the completeness rates, as expected, exhibit higher rates exceeding 0.95. We also observe that, as ϕ is set to lower values, the completeness rates fall, exhibiting the lowest value by setting $\phi = 0.001$ ($\theta = 0.001 \times 10^9 = 10^6$ and $\epsilon = 0.0001$). This happens because exact counts of the e_js near $\theta = 10^6$ become more and more uniform and this uniformity of values, as reported in [34], results in reducing the accuracy of the Count-Min sketches. Recall rates for both I-SEPIA and our protocols are constantly at 1.0. In terms of precision rates (Figs. 4c and 4d), our protocols perform slightly lower than I-SEPIA. Especially, by setting $\phi = 0.001$ with skew $z = 1$, precision falls below 0.95, which means that there is an amount of elements returned where their exact number of occurrences is below θ (false positives). When we increase skew ($z = 2$), precision rates increase accordingly reaching almost 1.0. As expected, the number of occurrences for some false positives returned by our protocols lie within the interval $(\lceil(\phi - \epsilon)N\rceil, \theta)$.

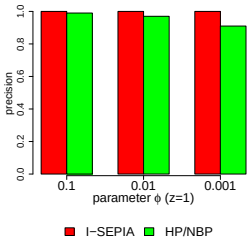
The higher precision rates achieved by I-SEPIA are outweighed by the cost of exceptionally large space required, whereas our protocols utilize orders of magnitude smaller data structures, as shown in Fig. 4e. The space utilized by our protocols depends on the specified parameters ϵ, δ, and on N. By setting lower values for ϕ, in cases where we need a lower threshold, we consequently decrease ϵ, which eventually results in using more space, as it is clearly shown in Fig. 4e by setting ϕ to 0.001. We illustrate the space requirements by assuming the RAM model [29], where a plain integer is represented by a machine word while in HP a homomorphically encrypted integer is represented by two machine



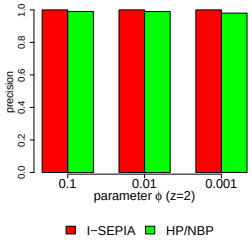
(a) By using skewed data, HP/NBP exhibit high completeness rates ($\phi = 0.01$).



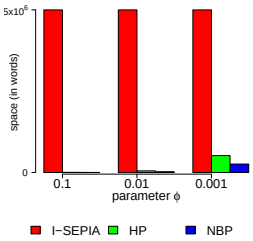
(b) The completeness rates of our protocols.



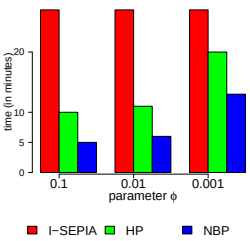
(c) I-SEPIA has slightly better precision rates.



(d) The precision rates of HP/NBP are almost 1.0, by using highly skewed data.



(e) Space requirements in words.



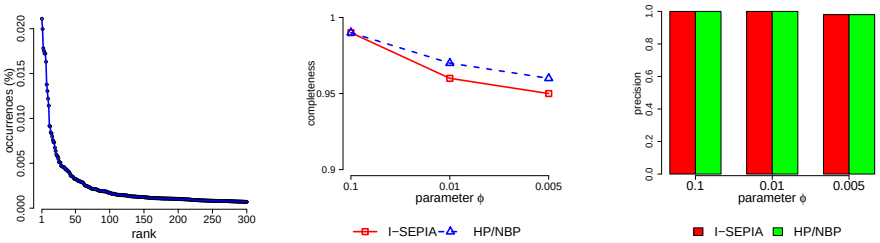
(f) Time performance in minutes.

Fig. 4: Comparing to I-SEPIA by using skewed data sets.

words respectively. Obviously, the space utilized affects execution time as well, since the number of operations required is proportional to the number of cells of each synopsis (S'_i, $\widetilde{S}_i$, Q_i and Q'_i), which depends on ϵ and consequently on ϕ, as illustrated in Fig. 4f. The extra time required for HP is due to the initial encryption of the cells of S'_1 in step 3a, while the homomorphic multiplications in step 4a add negligible overhead.

6.2 Evaluation using Real Data Sets

We received from OTS SA [1], which is a big Greek IT company, an anonymized list of the top 1,000-ranked web sites, as determined by the visits paid by their employees in January 2014. In Fig. 5a, the distribution of the occurrences of the e_js, where an e_j is a hit to a web site, is illustrated which indicates a stretched-exponential distribution with the stretching parameter set between 0.8 and 0.9. The total number of hits $N = 10^8$ and the number of distinct e_js $n = 10^3$. We set the frequency ϕ to 0.1, 0.01 and 0.005. If we set ϕ to 0.001 the corresponding threshold would be considerably reduced, namely it would be equal



(a) Distribution of the occurrences of the e_js according to their rank.

(b) The completeness rates of our protocols are constantly above 0.95.

(c) The precision rates for both our protocols and I-SEPIA are almost the same, very close to 1.0.

Fig. 5: Intersecting web logs provided by OTS SA.

to 10^5, and the synopses would return too many e_js as heavy hitters. We distributed randomly the e_js to five parties except for the heavy hitters, for each value of frequency ϕ, which appear universally in order to allow for evaluating the performance of both our protocols and I-SEPIA. In terms of completeness, our protocols, as expected due to the skewness of the data set, exhibit slightly better rates, as depicted in Fig. 5b. The recall rates are consistently 1.0 while the precision rates are almost the same for both our protocols and I-SEPIA, as shown in Fig. 5c.

7 Conclusions

We have proposed two protocols for efficient and privacy-preserving set intersection (PPSI) by using Count-Min sketches. The aim of these protocols is to allow an authority to create a global synopsis by performing private intersection of the sets that are individually summarized as local synopses by three or more parties. This global synopsis also provides count estimates of the intersected elements. Our protocols use a combination of privacy techniques, namely homomorphic computations, a secure distributed summation scheme, and noise addition. An empirical study conducted on large synthetic and real data sets validates the efficiency and accuracy of our protocols as compared to an existing PPSI protocol. In the future, we aim to (a) improve further the accuracy and scalability of our protocols, (b) apply time widows for monitoring the e_js, and (c) study other private set operations and techniques using efficient data structures.

References

1. OTS SA. `http://www.ots.gr/` (2014)
2. Adamic, L., Huberman, B.: Zipf's law and the internet. Glottonmetrics (11), 143–150 (2002)
3. Aggarwal, C., Yu, P.: A general survey of privacy-preserving data mining models and algorithms. Adv. Datab. Sys. 34, 11–52 (2008)
4. Agrawal, R., Evfimievski, A., Srikant, R.: Information sharing across private databases. In: SIGMOD. pp. 86–97. San Diego, California, USA (2003)
5. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. J. of Cryptol. 23(2), 281–343 (2010)
6. Broder, A., Mitzenmacher, M.: Network applications of Bloom filters: A survey. Internet Math. 1(4), 485–509 (2002)
7. Burkhart, M., Dimitropoulos, X.: Privacy-preserving distributed network troubleshooting - bridging the gap between theory and practice. ACM Trans. Inf. Sys. Sec. 14(4) (2011)
8. Charikar, M., Chen, K., Colton, M.F.: Finding frequent items in data streams. In: ICALP. pp. 693–703. Malaga, Spain (2002)
9. Clifton, C., Kantarcioglou, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explor. Newsl. 4(2), 28 – 34 (2002)
10. Cohen, S., Matias, Y.: Spectral Bloom filters. In: SIGMOD. pp. 241 – 252. San Diego, California (2003)
11. Cormode, G., Garofalakis, M.: Sketching streams through the net distributed approximate query tracking. In: VLDB. pp. 13–24. Trondheim, Norway (2005)
12. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the Count-Min sketch and its applications. J. of Algor. 55(1), 58 – 75 (2005)
13. Dachman-Soled, D., Malkin, T., Raykova, M., Yung, M.: Efficient robust private set intersection. Appl. Cryptog. 2(4), 289–303 (2012)
14. Dong, C., Chen, L., Wan, Z.: When private set intersection meets big data: an efficient and scalable protocol. In: SIGSAC. pp. 789 – 800. Berlin, Germany (2013)
15. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: EUROCRYPT. pp. 1–19. Interlaken, Switzerland (2004)
16. Frikken, K.: Privacy-preserving set union. Appl. Cryptog. Network Sec. 4521, 237–252 (2007)
17. Glassman, S.: A caching relay for the world wide web. Comput. Netw. ISDN Syst. 27(2), 165–173 (1994)
18. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: STOC. pp. 218–229. New York, USA (1987)
19. Hall, R., Fienberg, S.: Privacy-preserving record linkage. In: PSD. pp. 269–283. Corfu, Greece (2010)
20. Hazay, C., Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In: TCC, pp. 155–175. Springer, New York, USA (2008)
21. Jauhari, M., Saxena, A., Gautam, J.: Zipf's law and the number of hits on the world wide web. Annals of Lib. and Inf. Studies 54, 81–84 (2007)
22. Kantarcioglu, M., Jiang, W., Malin, B.: A privacy-preserving framework for integrating person-specific databases. In: PSD. pp. 298–314. Springer, Istanbul, Turkey (2008)

23. Kiayias, A., Mitrofanova, A.: Testing disjointness of private datasets. In: FC, pp. 109–124. Springer, Roseau, The Commonwealth Of Dominica (2005)
24. Kissner, L., Song, D.: Privacy-preserving set operations. In: CRYPTO. pp. 241–257. Springer, Santa Barbara, California, USA (2005)
25. Krashakov, S., Teslyuk, A., Shchur, L.: On the universality of rank distributions of website popularity. Comp. Netw. 50(11), 1769–1780 (2006)
26. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: keyed-hashing for message authentication, Internet RFC 2104 (1997), `http://tools.ietf.org/html/rfc2104`
27. Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. J. Priv. Conf. 1(1) (2009)
28. Many, D., Burkhart, M., Dimitropoulos, X.: Fast private set operations with sepia. Tech. Rep. no. 345, ETH Zurich (2012)
29. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)
30. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: STOC. pp. 245–254. Atlanta, Georgia, USA (1999)
31. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. pp. 223 – 238. Prague, Czech Republic (1999)
32. Pierre, K., Lai, S., Yiu, K., Chow, C., Chong, L., Hui, C.: An efficient Bloom filter based solution for multiparty private matching. In: SAM (2006)
33. Roughan, M., Zhang, Y.: Secure distributed data-mining and its application to large-scale network measurements. SIGCOMM Comput. Commun. Rev. 36(1), 7–14 (2006)
34. Rusu, F., Dobra, A.: Statistical analysis of sketch estimators. In: SIGMOD. pp. 187 – 198. Beijing, China (2007)
35. Vatsalan, D., Christen, P., Verykios, V.S.: A taxonomy of privacy-preserving record linkage techniques. J. Inf. Sys. 38(6), 946–969 (2013)
36. Yao, A.: How to generate and exchange secrets. In: SFCS. pp. 162–167. Toronto, Canada (1986)