Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Faster fog-aided private set intersection with integrity preserving*

Qiang WANG[1], Fu-cai ZHOU‡[1], Tie-min MA[2], Zi-feng XU[1]

[1]*Software College, Northeastern University, Shenyang 100819, China*

[2]*School of Computer Science and Engineering, Northeastern University, Shenyang 100819, China*

E-mail: wangq3635@126.com; fczhou@mail.neu.edu.cn; mtmwx@163.com; dk@tnimdk.com

Received Aug. 30, 2018; Revision accepted Nov. 11, 2018; Crosschecked Dec. 17, 2018

Abstract: Private set intersection (PSI) allows two parties to compute the intersection of their private sets while revealing nothing except the intersection. With the development of fog computing, the need has arisen to delegate PSI on outsourced datasets to the fog. However, the existing PSI schemes are based on either fully homomorphic encryption (FHE) or pairing computation. To the best of our knowledge, FHE and pairing operations consume a huge amount of computational resource. It is therefore an untenable scenario for resource-limited clients to carry out these operations. Furthermore, these PSI schemes cannot be applied to fog computing due to some inherent problems such as unacceptable latency and lack of mobility support. To resolve this problem, we first propose a novel primitive called "faster fog-aided private set intersection with integrity preserving", where the fog conducts delegated intersection operations over encrypted data without the decryption capacity. One of our technical highlights is to reduce the computation cost greatly by eliminating the FHE and pairing computation. Then we present a concrete construction and prove its security required under some cryptographic assumptions. Finally, we make a detailed theoretical analysis and simulation, and compare the results with those of the state-of-the-art schemes in two respects: communication overhead and computation overhead. The theoretical analysis and simulation show that our scheme is more efficient and practical.

Key words: Private set intersection; Fog computing; Verifiable; Data privacy
https://doi.org/10.1631/FITEE.1800518 **CLC number:** TP309

1 Introduction

Private set intersection (PSI) enables two parties to compute the intersection of their private sets while revealing nothing except the intersection. It was first introduced by Freedman et al. (2004) based on polynomial oblivious evaluation, which required two parties, Alice and Bob, to interact directly with each other and jointly perform the set intersection operation among locally available datasets. Over the past years, PSI has become an essential building-block in many applications such as common friend discovery (Nagy et al., 2013), fully-sequenced genome test (Baldi et al., 2011), and location sharing (Narayanan et al., 2011). Many variants of PSI have been proposed with the aims of hiding input sizes (Ateniese et al., 2011), supporting multiple clients (Kolesnikov et al., 2017), learning the number of users in common and the sum of the associated values (Ion et al., 2017), and preventing malicious cloud from deviating from the protocol (Kamara et al., 2014; Zheng and Xu, 2015; Abadi et al., 2016; Rindal and Rosulek, 2017a, 2017b; Zhang et al., 2017). Moreover, many researchers have focused on reducing the computation and communication overheads

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61872069, 61772127, 61703088, and 61472184)

ORCID: Qiang WANG, http://orcid.org/0000-0003-4480-9314

© Zhejiang University and Springer-Verlag GmbH Germany, part of Springer Nature 2018

(Pinkas et al., 2014, 2015, 2018; Kolesnikov et al., 2016; Chen et al., 2017; Orrù et al., 2017; Falk et al., 2018).

Due to the lower cost, higher reliability, better performance, and faster deployment of cloud computing, delegating PSI to the cloud is becoming a trend. However, there are still some challenges to be solved (Wu et al., 2018). The most important one is that the cloud cannot be fully trusted and it may misbehave by forging the computation or disclosing the sensitive data (Guan et al., 2017). Hence, it is essential to not only guarantee the correctness of the computation but also preserve the confidentiality of their outsourced datasets. As a result, verifiable delegated set intersection (VDSI) has been proposed. To the best of our knowledge, there are only two schemes (Zheng and Xu, 2015; Abadi et al., 2016) that satisfy the above requirements simultaneously. Abadi et al. (2016) used fully homomorphic encryption (FHE) to compute outsourced computation. Zheng and Xu (2015) used pairing operations to check the correctness of the outsourced computation. However, FHE and pairing operations consume a huge amount of computational resource. Therefore, such resource consumption is untenable for resource-limited clients. Furthermore, some applications and services cannot benefit from cloud computing due to its inherent problems such as unacceptable latency or lack of mobility support. To tackle these problems, a new computing paradigm called "fog computing" was introduced, which provides elastic resources and services to end users at the edge of the network. From these points, there is a pressing need for an faster private set intersection scheme in fog computing.

In this study, we introduce a novel primitive called faster fog-aided private set intersection with integrity preserving (FFPSI), which greatly reduces the computation cost for the client by eliminating FHE and pairing operations. Meanwhile, it not only protects the confidentiality of their outsourced data but also checks the integrity of outsourced PSI computation. Apart from these, our proposed scheme ensures that the outsourced data cannot be used without the owner's permission. There are three main contributions in theoretical aspect: (1) We introduce the concept of FFPSI and give its system model and security model. (2) Taking advantage of the Rivest-Shamir-Adleman (RSA) cryptosystem, we design an efficient FFPSI protocol and prove its security against the untrusted fog. (3) We assess the performance of the proposed scheme through theoretical analysis and simulation, and compare the results with those of the state-of-the-art schemes in terms of computation overhead and communication overhead. Theoretical analysis and simulation show that our proposed scheme is more efficient and practical.

There are also three main contributions in practical aspect: (1) high checkability of the correctness of computation results; (2) strong privacy for the data owner in the process of outsourcing and delegated computation; (3) low computational burden for the data owner by eliminating FHE and pairing operations either when the fog identifies the common dataset or when the data owner checks the integrity of delegated computation.

2 Preliminaries

2.1 RSA cryptosystem

1. (RSA.PK, RSA.SK) $\leftarrow$ RSA.KeyGen(1^λ)

To generate public and private keys, the algorithm works as follows: (1) Choose two distinct large primes p and q randomly and compute $n = pq$. (2) Compute $\varphi(n) = (p-1)(q-1)$ and choose a random e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$. (3) Compute an integer d such that $de = 1 \bmod \varphi(n)$. (4) Set public key RSA.PK $= (e, n)$ and private key RSA.SK $= (d, n)$.

2. RSA.C $\leftarrow$ RSA.Enc(RSA.PK, m)

Given a plaintext m, the algorithm encrypts it and generates the resulting ciphertext RSA.C $= m^e \bmod n$.

3. $m \leftarrow$ RSA.Dec(RSA.SK, RSA.C)

Given a ciphertext RSA.C, the algorithm decrypts it and recovers $m =$ RSA.C$^d \bmod n$.

2.2 Unforgeable digital signature

Let Sig = (Sig.KeyGen, Sig.Sign, Sig.Verify) be a secure digest scheme, where Sig.KeyGen generates a pair of keys (Sig.PK, Sig.SK), Sig.Sign generates a digest for a message, and Sig.Verify determines if a message matches a digest. Any digest scheme satisfying the standard definition of unforgeability under adaptive chosen-message attacks is sufficient for the purpose of this study.

2.3 Discrete logarithm problem

This problem was proposed by Nyberg and Rueppel (1994); that is, given $g, h \in G$, it is difficult to find an $x \in G$ such that $g^x = h$.

2.4 Integer factorization problem

Given a positive integer N, it is difficult to compute its decomposition into prime numbers, i.e., $N = \prod {p_i}^{e_i}$, where p_i is a prime.

2.5 Polynomial extend Euclidean algorithm

It is an extension to the Euclidean algorithm for polynomial. It computes $g(x) = \gcd(a(x), b(x))$ and the Bézout cofactors $r(x)$, $s(x)$ such that $g(x) = r(x)a(x) + s(x)b(x)$. The algorithm is shown in Algorithm 1.

Algorithm 1 Polynomial extend Euclidean algorithm

$(r_0(x), r_1(x), s_0(x), s_1(x), t_0(x), t_1(x)) := (a(x), b(x), 1, 0, 0, 1)$
$i := 1$
while $r_i(x) \neq 0$ **do**
  $q_i(x) :=$ quotient of $r_{i-1}(x)$ on division by $r_i(x)$
  $(r_{i+1}(x), s_{i+1}(x), t_{i+1}(x)) := (r_{i-1}(x), s_{i-1}(x), t_{i-1}(x)) - q_i(r_i(x), s_i(x), t_i(x))$
  $i := i + 1$
end while
$(g(x), s(x), t(x)) = (r_{i-1}(x), s_{i-1}(x), t_{i-1}(x))$
Return $g(x)$

2.6 Characteristic polynomial

Given a set $S = \{s_1, s_2, \ldots, s_n\}$, where elements $s_i \in Z_p$ can be represented by a polynomial (Ghosh et al., 2015). The polynomial $P(\alpha) = \prod_{i=1}^{n} (\alpha + x_i) = \sum_{i=0}^{n} c_i \alpha^i$ from $Z_p[\alpha]$, where α is a formal variable, is called the characteristic polynomial of set S.

2.7 Set intersection

Let $I = S_1 \cap S_2 \cap \ldots \cap S_t$. We express the correctness of intersection I by means of the following two conditions (Papamanthou et al., 2011):

Subset condition: $I \subseteq S_1 \wedge I \subseteq S_2 \wedge \ldots \wedge I \subseteq S_t$. This condition ensures that the intersection is a subset of all set indices.

Completeness condition: $(S_1 - I) \cap (S_2 - I) \cap \ldots \cap (S_t - I) = \varnothing$. This condition ensures that none of the elements has been omitted from the intersection. From the completeness condition, Lemma 1 can be deduced. For details, please refer to Ghosh et al. (2015).

Lemma 1 Set I is the intersection of sets $S_1, S_2, \ldots, S_t$, if and only if there exist polynomials $q_1(\alpha), q_2(\alpha), \ldots, q_t(\alpha)$ such that $q_1(\alpha)P_{S_1-I}(\alpha) + q_2(\alpha)P_{S_2-I}(\alpha) + \ldots + q_t(\alpha)P_{S_t-I}(\alpha) = 1$.

3 Definition

In this section, we first give the problem statement formally, and then present the system model and its security required in our proposed scheme.

3.1 Problem statement

In this subsection, we consider the problem of verifiable set intersection over encrypted data. This problem involves three parties: Alice with a dataset D_a, Bob with a dataset D_b, and a fog. Alice and Bob want to calculate $\mathrm{DI} = D_a \cap D_b$, but they do not want to share their datasets with each other because they do not trust each other. Due to the limited computational resource and storage capacity, Alice and Bob wish the fog to help them conduct intersection operation in a privacy-preserving way. To reveal nothing to the fog, Alice and Bob first encode their datasets (denoted by C_a and C_b) using their public keys, respectively. Thereafter, they migrate C_a and C_b to the remote fog and delegate the intersection operation to the fog. When they become interested in the intersection of D_a and D_b, the fog obtains their permission and conducts the intersection over their encrypted data C_a and C_b without the decryption capability. In the end, the fog learns nothing from the process of this protocol. Alice and Bob learn nothing about each other's data beyond what can be deduced from the intersection result DI. Furthermore, Alice and Bob can check the correctness of the intersection result returned by the fog at a low cost.

To the best of our knowledge, all existing solutions are based on either pairing or homomorphic encryption. The existing protocols do not scale up well since computation complexity is dominated mainly by either pairing or homomorphic encryption. To reduce the computation cost greatly, we design an efficient protocol without pairing and homomorphic encryption in this study.

3.2 System model

The FFPSI protocol contains three different kinds of entities, which are illustrated in Fig. 1. Their roles are identified as follows:

1. Trusted third party (TTP): a fully trusted entity, who is responsible for bootstrapping the public parameter for the system.

2. Data owners, Alice and Bob: the honest entities with limited computation capability, who can reside a huge number of datasets and delegate some computations to the fog. Meanwhile, they can check the integrity of outsourced computation at a low cost.

3. Fog: an untrusted entity with powerful computation resource, who can perform the corresponding computation outsourced by the data owner and return the computation result along with a witness.

Before presenting the definition of our scheme, we define some notations (Table 1).

Definition 1 (Faster fog-aided private set intersection with integrity preserving) The FFPSI scheme contains seven procedures:

1. params $\leftarrow$ Setup(1^λ): The procedure is run by TTP. It takes the security parameter λ as the input and outputs the public parameters params.

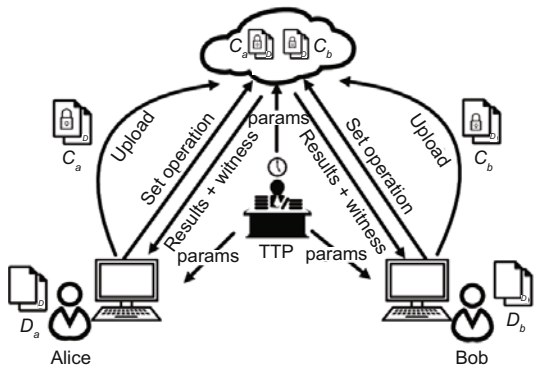

Fig. 1 Architecture of the faster fog-aided private set intersection with the integrity preserving protocol

Table 1 Notations

Notation	Description
λ	Security parameter
params	Public parameters
PK	Public key
SK	Secret key
$\mathrm{AK}_a(\mathrm{AK}_b)$	Alice's (Bob's) authorized key
$D_a(D_b)$	Alice's (Bob's) plaintext dataset
$C_a(C_b)$	Alice's (Bob's) encrypted dataset
$\xi_a(\xi_b)$	Alice's (Bob's) signature
DI	Intersection result, i.e., $D_a \cap D_b$
$I_a(I_b)$	Intersection result encrypted under Alice's (Bob's) public key
$\zeta_a(\zeta_b)$	Alice's (Bob's) witness returned by the fog

2. (PK, SK) $\leftarrow$ KeyGen(params): The procedure is run by any entity except TTP in the system. It takes public parameters params as the input and generates a key pair (PK, SK) for the public key cryptosystem. The public key PK is made public and the secret key SK is kept confidential by himself/herself. We denote Alice's key pair by (PK_a, SK_a), Bob's key pair by (PK_b, SK_b), and fog's key pair by (PK_c, SK_c).

3. $(C_a, \xi_a) \leftarrow \mathrm{Enc}(\mathrm{PK}_a, \mathrm{PK}_c, \mathrm{params}, D_a)$: The procedure is run by Alice. It takes public keys PK_a and PK_c, public parameters params, and dataset D_a in message space M as the inputs, to generate the encrypted dataset C_a and digest ξ_a. In a similar way, Bob can produce his encrypted dataset C_b and digest ξ_b, where $(C_b, \xi_b) \leftarrow \mathrm{Enc}(\mathrm{PK}_b, \mathrm{PK}_c, \mathrm{params}, D_b)$.

4. $\mathrm{AK}_a \leftarrow \mathrm{AuGen}(\mathrm{SK}_a, \mathrm{PK}_b, \xi_a)$: The procedure is run by Alice. It takes her private key SK_a, Bob's public key PK_b, and digest ξ_a as the inputs, to output the authorized key AK_a, which will be sent to the fog. Similarly, Bob can generate and send AK_b to the fog, where $\mathrm{AK}_b \leftarrow \mathrm{AuGen}(\mathrm{SK}_b, \mathrm{PK}_a, \xi_b)$.

5. $((I_a, \zeta_a), (I_b, \zeta_b)) \leftarrow \mathrm{SetI}(C_a, \mathrm{AK}_a, C_b, \mathrm{AK}_b)$: The procedure is run by the fog. It takes encrypted datasets C_a, C_b, and authorized keys AK_a, AK_b as the inputs, to produce set intersection result I_a (I_b) along with a witness ζ_a (ζ_b) for Alice (Bob). The witnesses ζ_a and ζ_b can check whether the fog conducts the delegated set intersection operation faithfully.

6. {"0" or "1"} $\leftarrow \mathrm{Verify}(\mathrm{SK}_a, \xi_a, I_a, \zeta_a, \mathrm{params})$: The procedure is run by Alice. It takes her secret key SK_a, digest ξ_a, encrypted intersection result I_a, witness ζ_a, and public parameters params as the inputs, to verify that I_a is indeed the result of set intersection. It outputs "1" or "0", denoting the result is valid or not, respectively. Similarly, Bob can check the correctness of computation.

7. $\{\mathrm{DI}_a \text{ or } \perp\} \leftarrow \mathrm{Dec}(\mathrm{SK}_a, I_a)$: The procedure is run by Alice. It takes Alice's secret key SK_a and encrypted intersection result I_a as the inputs, to decrypt I_a. If the decryption succeeds, it outputs the resulting intersection DI_a; otherwise, it outputs $\perp$. Note that this procedure can also be used to decrypt C_a. In a similar way, Bob can decrypt the ciphertext I_b and C_b.

3.3 Correctness and security definition

The FFPSI scheme should be correct, unforgeable, data owner zero-knowledge, and fog zero-knowledge. Intuitively, the FFPSI scheme is correct if whenever its algorithms are executed honestly, a correct result will never be rejected. More formally, we give the following definition:

Definition 2 (FFPSI correctness) The FFPSI scheme is correct if the following holds: First, it runs algorithm Setup to bootstrap the public parameters params. Then Alice, Bob, and the fog conduct algorithm KeyGen to generate their key pairs $(\mathrm{PK}_a, \mathrm{SK}_a)$, $(\mathrm{PK}_b, \mathrm{SK}_b)$, and $(\mathrm{PK}_c, \mathrm{SK}_c)$, respectively. For two original datasets D_a and D_b, execute $(C_a, \xi_a) \leftarrow \mathrm{Enc}(\mathrm{PK}_a, \mathrm{PK}_c, \mathrm{params}, D_a)$ and $(C_b, \xi_b) \leftarrow \mathrm{Enc}(\mathrm{PK}_b, \mathrm{PK}_c, \mathrm{params}, D_b)$, respectively. The algorithm AuGen is involved twice to generate two authorized keys AK_a and AK_b such that: (1) the resulting ciphertext sets C_a, I_a, C_b, and I_b can be successfully transformed into the original plaintext sets by performing algorithm Dec; (2) the outputs of algorithm SetI can be successfully verified by Verify.

Intuitively, the FFPSI scheme is unforgeable if an arbitrary adversary cannot convince a verifier to accept a wrong result with an overwhelming probability. More formally, we give the following definition:

Definition 3 (FFPSI unforgeability) Let FFPSI be a faster fog-aided private set intersection with the integrity preserving protocol and let $A(\cdot) = (A_0, A_1)$ be a two-tuple of probabilistic polynomial time (PPT) machine. We define security via the following experiment:

$$
\begin{aligned}
&\mathrm{EXP}_A^{\mathrm{UFG}}[\mathrm{FFPSI}, \lambda] : \mathrm{params} \leftarrow \mathrm{Setup}(1^\lambda),\\
&(\mathrm{PK}_a, \mathrm{SK}_a) \leftarrow \mathrm{KeyGen}(\mathrm{params}),\\
&(\mathrm{PK}_b, \mathrm{SK}_b) \leftarrow \mathrm{KeyGen}(\mathrm{params}),\\
&(\mathrm{PK}_c, \mathrm{SK}_c) \leftarrow \mathrm{KeyGen}(\mathrm{params}),\\
&(D_a, D_b) \leftarrow A_0\left(\mathrm{PK}_a, \mathrm{PK}_b, \mathrm{PK}_c, \mathrm{SK}_c\right),\\
&(C_a, \xi_a) \leftarrow \mathrm{Enc}(\mathrm{PK}_a, \mathrm{PK}_c, \mathrm{params}, D_a),\\
&(C_b, \xi_b) \leftarrow \mathrm{Enc}(\mathrm{PK}_b, \mathrm{PK}_c, \mathrm{params}, D_b),\\
&\mathrm{AK}_a \leftarrow \mathrm{AuGen}(\mathrm{SK}_a, \mathrm{PK}_b, \xi_a),\\
&\mathrm{AK}_b \leftarrow \mathrm{AuGen}(\mathrm{SK}_b, \mathrm{PK}_a, \xi_b),\\
&\{(I'_a, \zeta'_a), (I'_b, \zeta'_b)\} \leftarrow A_1(\mathrm{PK}_a, \mathrm{PK}_b, \mathrm{PK}_c,\\
&\qquad \mathrm{SK}_c, D_a, D_b, C_a, C_b, \xi_a, \xi_b, \mathrm{AK}_a, \mathrm{AK}_b),\\
&\beta_a \leftarrow \mathrm{Verify}(\mathrm{SK}_a, \xi_a, I'_a, \zeta', \mathrm{params}),\\
&\beta_b \leftarrow \mathrm{Verify}(\mathrm{SK}_b, \xi_b, I'_b, \zeta'_b, \mathrm{params}).
\end{aligned}
$$

If (β_a or $\beta_b = 1$) and ($\mathrm{Dec}(\mathrm{SK}_a, I'_a)$ or $\mathrm{Dec}(\mathrm{SK}_b, I'_b)$) $\neq D_a \cap D_b$, output "1"; otherwise, output "0".

For any $\lambda \in N$, we define the advantage of arbitrary adversary A in the above experiment against FFPSI as $\mathrm{Adv}_A^{\mathrm{UFG}}(\mathrm{FFPSI}, \lambda) = \Pr[\mathrm{EXP}_A^{\mathrm{UFG}}[\mathrm{FFPSI}, \lambda] = 1]$. We say that FFPSI achieves unforgeability if $\mathrm{Adv}_A^{\mathrm{UFG}}(\mathrm{FFPSI}, \lambda) \leq \mathrm{neg}(\lambda)$.

Intuitively, the FFPSI scheme is data owner zero-knowledge if the data owner, either Alice or Bob, cannot obtain any knowledge of others' dataset beyond what can be deduced from intersection result DI. More formally, we give the following definition:

Definition 4 (Data owner zero-knowledge) Let FFPSI be a faster fog-aided private set intersection with the integrity preserving protocol, and let $A(\cdot)$ be a PPT machine. We define security via the following experiment:

$$
\begin{aligned}
&\mathrm{EXP}_A^{\mathrm{DOZK}}[\mathrm{FFPSI}, \lambda] : \mathrm{params} \leftarrow \mathrm{Setup}(1^\lambda),\\
&(\mathrm{PK}_a, \mathrm{SK}_a) \leftarrow \mathrm{KeyGen}(\mathrm{params}),\\
&(\mathrm{PK}_b, \mathrm{SK}_b) \leftarrow \mathrm{KeyGen}(\mathrm{params}),\\
&(\mathrm{PK}_c, \mathrm{SK}_c) \leftarrow \mathrm{KeyGen}(\mathrm{params}),\\
&(C_a, \xi_a) \leftarrow \mathrm{Enc}(\mathrm{PK}_a, \mathrm{PK}_c, \mathrm{params}, D_a),\\
&(C_b, \xi_b) \leftarrow \mathrm{Enc}(\mathrm{PK}_b, \mathrm{PK}_c, \mathrm{params}, D_b),\\
&\mathrm{AK}_a \leftarrow \mathrm{AuGen}(\mathrm{SK}_a, \mathrm{PK}_b, \xi_a),\\
&\mathrm{AK}_b \leftarrow \mathrm{AuGen}(\mathrm{SK}_b, \mathrm{PK}_a, \xi_b),\\
&\{D_a{}' \cup D_b{}'\} \leftarrow A(\mathrm{PK}_a, \mathrm{PK}_b, \mathrm{PK}_c, \mathrm{SK}_c,\\
&\qquad C_a, C_b, \xi_a, \xi_b, \mathrm{AK}_a, \mathrm{AK}_b).
\end{aligned}
$$

If $\exists d \in \{D_a{}' \cup D_b{}'\}$ and $d \in \{D_a \cup D_b\}$, output "1"; otherwise, output "0".

For any $\lambda \in N$, D_a, and D_b, we define the advantage of arbitrary adversary A in the above experiment against FFPSI as $\mathrm{Adv}_A^{\mathrm{DOZK}}(\mathrm{FFPSI}, \lambda) = \Pr[\mathrm{EXP}_A^{\mathrm{DOZK}}[\mathrm{FFPSI}, \lambda] = 1]$. We say that FFPSI achieves the data owner zero-knowledge property if $\mathrm{Adv}_A^{\mathrm{DOZK}}(\mathrm{FFPSI}, \lambda) \leq \mathrm{neg}(\lambda)$.

Intuitively, the FFPSI scheme is fog zero-knowledge if the fog cannot obtain any knowledge of the outsourced dataset beyond its cardinality. More formally, we give the following definition:

Definition 5 (Fog zero-knowledge) Let FFPSI be a faster fog-aided private set intersection with the integrity preserving protocol, and let $A(\cdot)=(A_0,A_1)$ be a two-tuple of PPT machine. We define security via the following experiment:

$\mathrm{EXP}_A^{\mathrm{FZK}}[\mathrm{FFPSI},\lambda]:\mathrm{params}\leftarrow\mathrm{Setup}(1^\lambda)$,
$(\mathrm{PK}_a,\mathrm{SK}_a)\leftarrow\mathrm{KeyGen}(\mathrm{params})$,
$(\mathrm{PK}_b,\mathrm{SK}_b)\leftarrow\mathrm{KeyGen}(\mathrm{params})$,
$(\mathrm{PK}_c,\mathrm{SK}_c)\leftarrow\mathrm{KeyGen}(\mathrm{params})$,
$(D_0,D_1)\leftarrow A_0(\mathrm{PK}_a,\mathrm{PK}_b,\mathrm{PK}_c,\mathrm{SK}_c)$ s.t. $|D_0|=|D_1|$,
$\beta\in_R\{0,1\}$,
$(C_{a,\beta},\xi_{a,\beta})\leftarrow\mathrm{Enc}(\mathrm{PK}_a,\mathrm{PK}_c,\mathrm{params},D_\beta)$,
$(C_{b,\beta},\xi_{b,\beta})\leftarrow\mathrm{Enc}(\mathrm{PK}_b,\mathrm{PK}_c,\mathrm{params},D_\beta)$,
$\beta'\leftarrow A_1(\mathrm{PK}_a,\mathrm{PK}_b,\mathrm{PK}_c,\mathrm{SK}_c,D_0,D_1,C_{a,\beta},\xi_{a,\beta},C_{b,\beta},\xi_{b,\beta})$.

If $\beta'=\beta$, output "1"; otherwise, output "0".

For any $\lambda\in N$, we define the advantage of arbitrary adversary A in the above experiment against FFPSI as $\mathrm{Adv}_A^{\mathrm{FZK}}(\mathrm{FFPSI},\lambda)=\left|\Pr[\mathrm{EXP}_A^{\mathrm{FZK}}[\mathrm{FFPSI},\lambda]=1]-1/2\right|$. We say that the FFPSI scheme achieves the fog zero-knowledge property if $\mathrm{Adv}_A^{\mathrm{FZK}}(\mathrm{FFPSI},\lambda)\leq\mathrm{neg}(\lambda)$.

4 Construction

To briefly describe the construction, we first assume that there exists a probabilistic polynomial time algorithm GGen with input security parameter λ to generate a group G of prime order p with λ bits. The detailed construction is presented as follows:

Setup: To initialize the system public parameters, TTP works as follows: (1) Conduct algorithm GGen, using the security parameter λ as the input, to generate group parameter $\mathrm{pm}=(g,p,G)$, where $G=Z_p$. (2) Choose a random number $\alpha\in Z_p^*$ and pick a hash function $H:\{0,1\}^*\rightarrow Z_p^*$. (3) Set the public parameters $\mathrm{params}=(\mathrm{pm},\alpha,\mathrm{H})$.

KeyGen: To establish a public/private key pair, Alice conducts algorithm RSA.KeyGen, using the security parameter λ as the input, to generate $\mathrm{RSA.PK}_a$ as her public key PK_a and $\mathrm{RSA.SK}_a$ as her private key SK_a. Similarly, Bob and the fog can produce their key pairs, denoted by $(\mathrm{SK}_b,\mathrm{PK}_b)$ and $(\mathrm{SK}_c,\mathrm{PK}_c)$, respectively.

Enc: Let $D_a=\{d_{a,1},d_{a,2},\ldots,d_{a,n}\}$ be a dataset with size n. To generate the encrypted dataset and digest, Alice works as follows: (1) Pick a random $d_{a,0}\in Z_p^*$ and initialize digest ξ_a with 1. (2) For $0\leq i\leq n$: (a) conduct algorithm $\mathrm{RSA.Enc}(\mathrm{PK}_a,d_{a,i})$ to generate the resulting ciphertext $\mathrm{RSA.C}_{a,i}$, (b) compute $g^{d_{a,i}}$ and conduct algorithm $\mathrm{RSA.Enc}(\mathrm{PK}_c,g^{d_{a,i}})$ to generate the resulting ciphertext $\mathrm{RSA.C}_{a,i}^c$, (c) let $T_{a,i}=H(g^{d_{a,i}})$ and update ξ_a with $\xi_a(\alpha-T_{a,i})$, and (d) set the resulting ciphertext $C_{a,i}=\{\mathrm{RSA.C}_{a,i},\mathrm{RSA.C}_{a,i}^c\}$. (3) Set $C_a=\{C_{a,0},C_{a,1},\ldots,C_{a,n}\}$ and return C_a, ξ_a. In a similar way, Bob, with dataset $D_b=\{d_{b,1},d_{b,2},\ldots,d_{b,m}\}$ of size m, can produce the encrypted datasets $C_b=\{C_{b,1},C_{b,2},\ldots,C_{b,m}\}$ and digest ξ_b.

AuGen: To generate the authorized key AK_a, Alice works as follows: (1) Conduct algorithm RSA.Enc to encrypt ξ_b under Bob's public key PK_b and set the resulting ciphertext as CS_b. (2) Conduct algorithm $\mathrm{Sig.Sign}(\mathrm{CS}_b,\mathrm{SK}_a)$ and set the resulting signature as σ_a. (3) Set $\mathrm{AK}_a=\{\mathrm{CS}_b,\sigma_a\}$ and output AK_a. Likewise, Bob can generate the authorized key AK_b.

SetI: Given C_a, C_b, AK_a, and AK_b, the fog conducts the set intersection as follows: (1) Initialize two empty sets T_a and T_b. (2) For each item in C_a: (a) conduct algorithm $\mathrm{RSA.Dec}(\mathrm{RSA.C}_{a,i}^c,\mathrm{SK}_c)$ to decode $g^{d_{a,i}}$, and (b) compute $T_{a,i}=H(g^{d_{a,i}})$ and add the element $T_{a,i}$ to set T_a. (3) For each item in C_b: (a) conduct algorithm $\mathrm{RSA.Dec}(\mathrm{RSA.C}_{b,i}^c,\mathrm{SK}_c)$ to decode $g^{d_{b,i}}$, and (b) compute $T_{b,i}=H(g^{d_{b,i}})$ and add the element $T_{b,i}$ to set T_b. (4) Generate the set intersection $I=T_a\cap T_b$. (5) Generate the intersection result I_a with respect to C_a as $I_a=\{\mathrm{RSA.C}_{a,i}|T_{a,i}\in I\}$. (6) Generate the intersection result I_b with respect to C_b as $I_b=\{\mathrm{RSA.C}_{b,i}|T_{b,i}\in I\}$. (7) Generate the witnesses ζ_a and ζ_b as follows: (a) construct three polynomials $U(x)$, $V(x)$, $I(x)$ satisfying $U(x)=\prod_{t\in(T_a-I)}(x-t)$, $V(x)=\prod_{t\in(T_b-I)}(x-t)$, $I(x)=\prod_{t\in I}(x-t)$, (b) conduct the extended Euclidean algorithm to find two polynomials $U'(x)$, $V'(x)$ satisfying $U'(x)U(x)+V'(x)V(x)=1$, (c) compute $U(\alpha)$, $U'(\alpha)$, $V(\alpha)$, $V'(\alpha)$, $I(\alpha)$, and (d) set witnesses $\zeta_a=\{U(\alpha),U'(\alpha),V(\alpha),V'(\alpha),I(\alpha),\mathrm{CS}_b,\sigma_a\}$ and $\zeta_b=\{U(\alpha),U'(\alpha),V(\alpha),V'(\alpha),I(\alpha),\mathrm{CS}_a,\sigma_b\}$, and output $\{(I_a,\zeta_a),(I_b,\zeta_b)\}$.

Verify: To verify that I_a is indeed the intersection result, Alice does the following: (1)

Conduct algorithm Sig.verify to verify whether the signature σ_b is valid. (2) If not, return "0" and abort; otherwise, proceed to the next step. (3) Conduct algorithm Dec to obtain the plaintext intersection $\mathrm{DI}_a = \{d_{a,i}|\mathrm{RSA.Enc}(\mathrm{PK}_a, d_{a,i}) \in I_a\}$ and compute $\mathrm{RI}_a = \{H(g^{d_{a,i}})|\mathrm{RSA.C}_{a,i} \in I_a\}$. (4) Conduct algorithm Dec to decrypt CS_a and recover the digest ξ_b. (5) Parse ζ_a as $U(\alpha)$, $U'(\alpha)$, $V(\alpha)$, $V'(\alpha)$, $I(\alpha)$, CS_b, σ_a. (6) Construct the polynomial $I'(x) = \prod_{t\in \mathrm{RI}_a} (x-t)$ and check whether the following formula holds: $I'(\alpha) = I(\alpha)$. (7) If not, return "0" and abort; otherwise, proceed to the next step. (8) Check whether the following equation holds:

$$U'(\alpha)U(\alpha)+V'(\alpha)V(\alpha) = 1. \tag{1}$$

(9) If not, return "0" and abort; otherwise, proceed to the next step. (10) Check whether the following equations hold:

$$U(\alpha)I'(\alpha) = \xi_a, \tag{2}$$

$$V(\alpha)I'(\alpha) = \xi_b. \tag{3}$$

(11) If not, return "0" and abort; otherwise, return "1". Similarly, Bob can run this algorithm to verify whether the fog performs delegated intersection operation faithfully.

Dec: Alice/Bob can conduct algorithm RSA.Dec with her/his secret key and encrypted dataset to obtain the resulting plaintext set.

Remark 1 In the first step of Algorithm Enc, $d_{a,0}$ is to assure that no useful information is leaked. For details, please refer to Zheng and Xu (2015).

5 Analysis

5.1 Security analysis

It is not difficult to examine the correctness of the FFPSI scheme. In what follows, we focus on its security properties.

Theorem 1 If the subset condition and completeness condition hold, the FFPSI scheme achieves the unforgeability property as defined in Definition 3.

Proof There are two cases for breaking the unforgeability property. The first case is that an element is common but the fog does not return it as the result. The second case is that an element does not belong to the intersection but the fog returns it as the result.

For the first case, the unforgeability property relies on the completeness condition. In other words, if the completeness condition holds, the unforgeability property holds for our proposed scheme. According to the protocol construction, the data owner verifies the correctness of the result by checking Eq. (1) (i.e., Lemma 1).

For the second case, the unforgeability property relies on the subset condition. Similarly, if the subset condition holds, the unforgeability property holds for our proposed scheme. According to the protocol construction, the data owner verifies the integrity of the result by checking Eqs. (2) and (3).

As a result, if the subset condition and completeness condition hold, an arbitrary wrong result will never be accepted by the data owner. Since the subset condition and completeness condition have been proved in Ghosh et al. (2015), the unforgeability property holds for our proposed scheme.

Theorem 2 If the hardness of integer factorization holds, the FFPSI scheme achieves the data owner zero-knowledge property as defined in Definition 4.

Proof The proof of data owner zero-knowledge is trivial. During the protocol, there is just one part where Bob (Alice) obtains information about Alice's (Bob's) dataset beyond what can be deduced from the intersection result. The part is during Algorithm Enc at steps 2(a) and 2(b). For this part, the data owner zero-knowledge relies on the semantic security of the RSA public-key cryptosystem. In other words, if the RSA public-key cryptosystem is semantically secure, the data owners cannot learn any information about each other's dataset besides the intersection result. Since the RSA public-key cryptosystem is proved to have the semantic security property based on the hardness of integer factorization, the data owner zero-knowledge property holds for our proposed scheme.

Theorem 3 If the hardness of integer factorization and the discrete logarithm assumption hold, the FFPSI scheme achieves the fog zero-knowledge property as defined in Definition 5.

Proof There are two parts where the fog receives information about data owner's datasets. The first part is during Algorithm Enc at step 2(a), and the second part is during Algorithm Enc at step 2(b).

For the first part, similar to the proof of Theorem 2, fog zero-knowledge relies on the semantic security of the RSA public-key cryptosystem. In

other words, if the RSA public-key cryptosystem is semantically secure, the fog cannot learn any information about data owner's set. Since the RSA public-key cryptosystem is proved to hold the semantic security property based on the hardness of integer factorization (Falk et al., 2018), the data owner zero-knowledge property holds for our proposed scheme.

For the second part, we need to show that the information that the fog receives during Algorithm Enc at step 2(b) does not reveal any information about the data owner's set, beyond the cardinality of the outsourced set. According to the protocol construction, the fog receives encrypted data $\mathrm{RSA.C}_{a,i}^{c}$ during algorithm Enc at step 2(b). Note that $\mathrm{RSA.C}_{a,i}^{c}$ is encrypted under fog's public key, i.e., $\mathrm{RSA.C}_{a,i}^{c} = \mathrm{RSA.Enc}(\mathrm{PK}_c, g^{d_{a,i}})$. The fog could recover $g^{d_{a,i}}$ by conducting algorithm $\mathrm{RSA.Dec}(\mathrm{SK}_c, \mathrm{Enc}(\mathrm{PK}_c, g^{d_{a,i}}))$. Therefore, if the fog cannot recover $d_{a,i}$ from $g^{d_{a,i}}$, the zero-knowledge for the fog will hold. Since the hardness of the discrete logarithm (Papamanthou et al., 2011) holds, the fog cannot solve the discrete logarithm problem to recover $d_{a,i}$ from $g^{d_{a,i}}$.

From all of the above, the fog zero-knowledge property holds for the proposed scheme, since the hardness of integer factorization and the discrete logarithm assumption hold.

5.2 Performance analysis

Our proposed scheme is a pairing and FHE-free scheme. In the following, we first make a comparison between our proposed scheme and the state-of-the-art schemes (Zheng and Xu, 2015; Abadi et al., 2016) in two folds: communication overhead and computation overhead. Zheng and Xu (2015) took advantage of the pairing operation to check the correctness of outsourced computation. Abadi et al. (2016) used FHE and blind technology to check the integrity of outsourced computation. Then we give a prototypal implementation of our proposed scheme. The simulation confirms our theoretical analysis. For convenience, we denote Alice's dataset size by n, Bob's dataset size by m, intersection size by k, and the upper bound of set cardinality by d, where $k \leq n \leq d$, $k \leq m \leq d$.

5.2.1 Communication

In our proposed scheme, the communication overhead comes mainly from data outsourcing and set intersection. The communication comparison can be summarized in Table 2. Bob's communication cost is similar to Alice's. Due to limited space, Bob's communication cost is omitted. Compared with state-of-the-art schemes (Zheng and Xu, 2015; Abadi et al., 2016), our scheme has a better overhead performance.

5.2.2 Computation

Since computation complexity is dominated by bilinear pairing, exponentiation, homomorphic encryption, and homomorphic decryption, we evaluate computation cost by counting the number of these operations. The computation comparison is summarized in Table 3. Compared with state-of-the-art schemes (Zheng and Xu, 2015; Abadi et al., 2016), our scheme has a better performance since it is a pairing and FHE-free construction.

5.2.3 Simulation

We simulated our proposed scheme, VDSI, and VDPSI, and compared the computation overhead. All protocols were implemented using C and C++ programming languages. The development environment is summarized in Table 4. Based on the recommendations of NIST (Kerry et al., 2013), Paillier and RSA cryptosystems with 1024-bit security parameter and elliptic curve cryptography with 160-bit security parameter provide the same level of security. Therefore, we implemented our scheme and VDPSI

Table 2 Comparison of communication cost between our proposed scheme and state-of-the-art schemes

Phase	Communication cost					
	VDSI (Zheng and Xu, 2015)		VDPSI (Abadi et al., 2016)		Our scheme	
	Alice	Cloud	Alice	Cloud	Alice	Fog
Outsourcing	$(3n+1)Z_p^*+1G$	$(3n+3m+2)Z_p^*+2G$	$(10d+20)Z_p^*$	$(8d+12)Z_p^*$	$(2n+2)Z_p^*$	$(2n+2m+4)Z_p^*$
Intersection	$3kZ_p^*+4G$	$3kZ_p^*+4G$	$(2d+3)Z_p^*$	$(2d+3)Z_p^*$	$(2k+5)Z_p^*$	$(2k+5)Z_p^*$

$1G$ denotes one element of bilinear group G and $1Z_p^*$ denotes one item of Z_p^*

(Abadi et al., 2016) with 1024-bit security parameter. We implemented VDSI (Zheng and Xu, 2015) with 160-bit security parameter, where we instantiated the bilinear map with Type A curve with 160 bits in PBC Library v.0.5.14 (Stanford University, 2013).

In our simulation, we set the same size for datasets owned by Alice and Bob, i.e., $m = n$, and varied them from 2^{10} to 2^{15}. We also set the upper bound of the set cardinality $d = n$. In the following, we compared our scheme with state-of-the-art schemes. Fig. 2 depicts the comparison of the

Table 3 Comparison of computation cost between our proposed scheme and state-of-the-art schemes

Algorithm	Computation cost		
	VDSI (Zheng and Xu, 2015)	VDPSI (Abadi et al., 2016)	Our scheme
Enc	$3n$Exp+nPairing	N/A	$3n$Exp
Dec	$2n$Exp	N/A	nExp
AuGen	4Exp	$(2d+3)$FHE.E + $(6d+9)$Exp	3Exp
SetI	$3(n+m)$Pairing	$(2d+3)$FHE.E + $(2d+3)$Exp	$(n+m)$Exp
Verify	$(3k+2)$Exp + $(k+7)$Pairing	$(2d+3)$FHE.D	$2k$Exp

Exp: cost time of exponentiation; Pairing: cost time of bilinear pairing; FHE.E: cost time of fully homomorphic encryption; FHE.D: cost time of fully homomorphic decryption

Table 4 Development environment

Scheme	Security parameter size (bit)	Library	Hardware environment
VDSI (Zheng and Xu, 2015)	160	OpenSSL, GMP, PBC, NTL	CPU: Intel Core i7
VDPSI (Abadi et al., 2016)	1024	OpenSSL, GMP, NTL	Physical memory: 8 GB
Our scheme	1024	OpenSSL, NTL	OS: Ubuntu 18.04 LTS

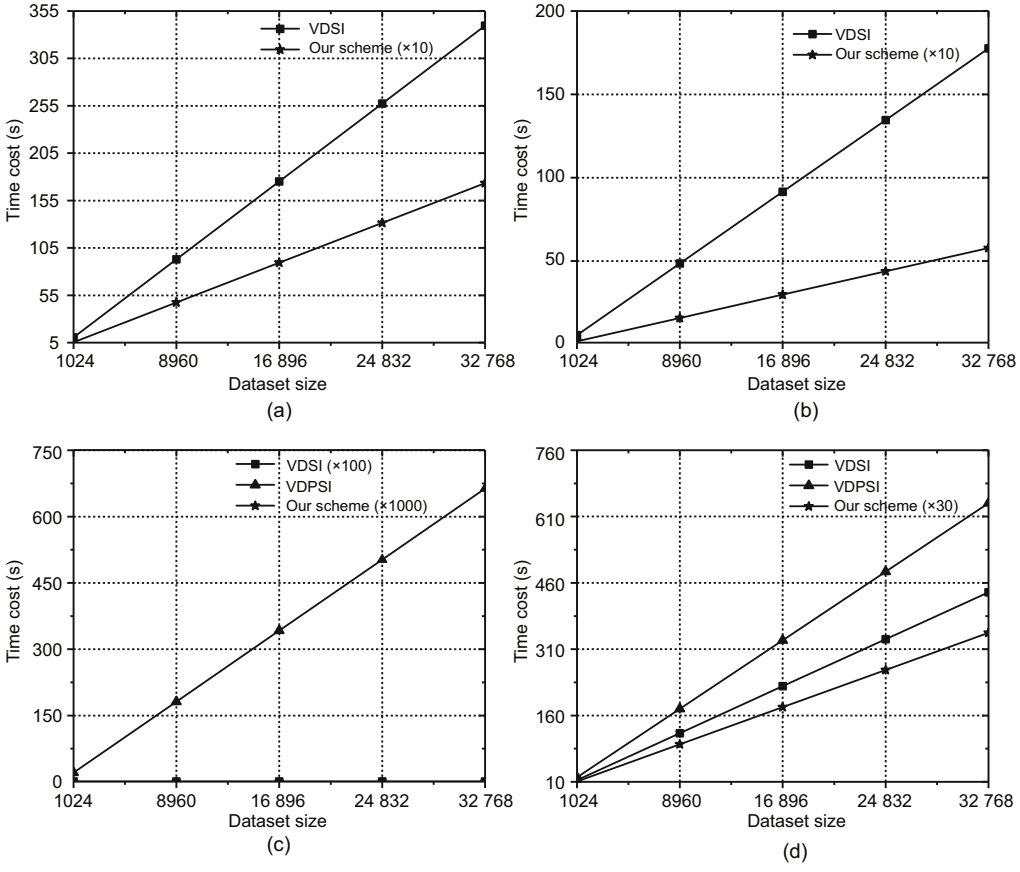

Fig. 2 Computation cost comparison among common algorithms: (a) Enc; (b) Dec; (c) AuGen; (d) SetI

execution time of algorithms Enc, Dec, AuGen, and SetI. To facilitate observation, the execution time of our scheme has been magnified, but the execution time is still far less than that of VDPSI and VDSI. It is easy to find that all algorithms in our scheme are more effective. These algorithms in each scheme scale linearly with the data size except Algorithm AuGen in our scheme. In our protocol, the execution time of Algorithm AuGen is constant. The execution time of AuGen and SetI in VDPSI is much more than that of the algorithms in our scheme and VDSI, respectively. Fig. 3 plots the execution time of algorithm Verify affected by the intersection size ratio. From all the above, our proposed scheme is more efficient and practical.

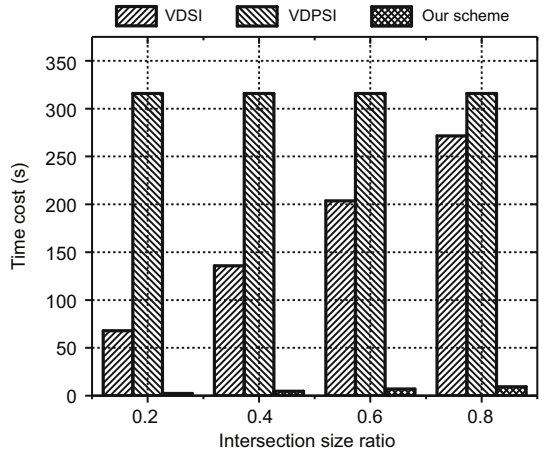

Fig. 3 Computation cost at dataset size $n = 32\,768$

6 Conclusions

In this paper, to outsource private set intersection computation to the fog, we proposed a new primitive called faster fog-aided private set intersection with integrity preserving (FFPSI), where the fog conducts delegated intersection over encrypted data without the decryption capacity. In addition, one of our technical highlights is to reduce the computation cost greatly by eliminating the pairing computation. The security of our proposed scheme is based on the discrete logarithm, integer factorization problem, and the correctness of set intersection. We made a detailed theoretical analysis and simulation between our scheme and several state-of-the-art schemes in two directions: communication overhead and computation overhead. Theoretical analysis and simulation showed that our scheme is more efficient and practical.

References

Abadi A, Terzis S, Dong CY, 2016. VD-PSI: verifiable delegated private set intersection on outsourced private datasets. Proc Int Conf on Financial Cryptography and Data Security, p.149-168. https://doi.org/10.1007/978-3-662-54970-4_9

Ateniese G, de Cristofaro E, Tsudik G, 2011. (If) size matters: size-hiding private set intersection. Proc Int Workshop on Public Key Cryptography, p.156-173. https://doi.org/10.1007/978-3-642-19379-8_10

Baldi P, Baronio R, de Cristofaro E, et al., 2011. Countering GATTACA: efficient and secure testing of fully-sequenced human genomes. Proc 18th ACM Conf on Computer and Communications Security, p.691-702. https://doi.org/10.1145/2046707.2046785

Chen H, Laine K, Rindal P, 2017. Fast private set intersection from homomorphic encryption. Proc ACM SIGSAC Conf on Computer and Communications Security, p.1243-1255. https://doi.org/10.1145/3133956.3134061

Falk BH, Noble D, Ostrovsky R, 2018. Private Set Intersection with Linear Communication from General Assumptions. Cryptology ePrint Archive: Report 2018/238.

Freedman MJ, Nissim K, Pinkas B, 2004. Efficient private matching and set intersection. Proc Int Conf on the Theory and Applications of Cryptographic Techniques, p.1-19. https://doi.org/10.1007/978-3-540-24676-3_1

Ghosh E, Ohrimenko O, Papadopoulos D, et al., 2015. Zero-Knowledge Accumulators and Set Operations. Cryptology ePrint Archive: 2015/404.

Guan ZT, Li J, Wu LF, et al., 2017. Achieving efficient and secure data acquisition for cloud-supported Internet of Things in smart grid. *IEEE Internet Things J*, 4(6):1934-1944. https://doi.org/10.1109/JIOT.2017.2690522

Ion M, Kreuter B, Nergiz E, et al., 2017. Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions. Cryptology ePrint Archive: Report 2017/738.

Kamara S, Mohassel P, Raykova M, et al., 2014. Scaling private set intersection to billion-element sets. Proc Int Conf on Financial Cryptography and Data Security, p.195-215. https://doi.org/10.1007/978-3-662-45472-5_13

Kerry CF, Secretary A, Director CR, 2013. Federal Information Processing Standards Publication Digital Signature Standard (DSS), FIPS PUB 186-4. National Institute of Standards and Technology of America.

Kolesnikov V, Kumaresan R, Rosulek M, et al., 2016. Efficient batched oblivious PRF with applications to private set intersection. Proc ACM SIGSAC Conf on Computer and Communications Security, p.818-829. https://doi.org/10.1145/2976749.2978381

Kolesnikov V, Matania N, Pinkas B, et al., 2017. Practical multi-party private set intersection from symmetric-key techniques. Proc ACM SIGSAC Conf on Computer and Communications Security, p.1257-1272. https://doi.org/10.1145/3133956.3134065

Nagy M, de Cristofaro E, Dmitrienko A, et al., 2013. Do I know you?: efficient and privacy-preserving common friend-finder protocols and applications. Proc 29th Annual Computer Security Applications Conf, p.159-168. https://doi.org/10.1145/2523649.2523668

Narayanan A, Thiagarajan N, Lakhani M, et al., 2011. Location privacy via private proximity testing. Network and Distributed System Security Symp.

Nyberg K, Rueppel RA, 1994. Message recovery for signature schemes based on the discrete logarithm problem. Proc Workshop on the Theory and Application of Cryptographic Techniques, p.182-193. https://doi.org/10.1007/BFb0053434

Orrù M, Orsini E, Scholl P, 2017. Actively secure 1-out-of-N OT extension with application to private set intersection. Proc Cryptographers' Track at the RSA Conf, p.381-396. https://doi.org/10.1007/978-3-319-52153-4_22

Papamanthou C, Tamassia R, Triandopoulos N, 2011. Optimal verification of operations on dynamic sets. 31st Annual Cryptology Conf, p.91-110. https://doi.org/10.1007/978-3-642-22792-9_6

Pinkas B, Schneider T, Zohner M, 2014. Faster private set intersection based on OT extension. Proc 23rd USENIX Conf on Security Symp, p.797-812.

Pinkas B, Schneider T, Segev G, et al., 2015. Phasing: private set intersection using permutation-based hashing. Proc 24th USENIX Conf on Security Symp, p.515-530.

Pinkas B, Schneider T, Zohner M, 2018. Scalable private set intersection based on OT extension. *ACM Trans Priv Secur*, 21(2):7.1-7.35. https://doi.org/10.1145/3154794

Rindal P, Rosulek M, 2017a. Improved private set intersection against malicious adversaries. Proc 36th Annual Int Conf on the Theory and Applications of Cryptographic Techniques, p.235-259. https://doi.org/10.1007/978-3-319-56620-7_9

Rindal P, Rosulek M, 2017b. Malicious-secure private set intersection via dual execution. Proc ACM SIGSAC Conf on Computer and Communications Security, p.1229-1242. https://doi.org/10.1145/3133956.3134044

Stanford University, 2013. The Pairing-Based Cryptography Library. https://crypto.stanford.edu/pbc

Wu J, Dong MX, Ota K, et al., 2018. Big data analysis-based secure cluster management for optimized control plane in software-defined networks. *IEEE Trans Netw Serv Manag*, 15(1):27-38. https://doi.org/10.1109/TNSM.2018.2799000

Zhang E, Li FH, Niu B, et al., 2017. Server-aided private set intersection based on reputation. *Inform Sci*, 387:180-194. https://doi.org/10.1016/j.ins.2016.09.056

Zheng QJ, Xu SH, 2015. Verifiable delegated set intersection operations on outsourced encrypted data. Proc IEEE Int Conf on Cloud Engineering, p.175-184. https://doi.org/10.1109/IC2E.2015.38