Are you The One to Share? Secret Transfer with Access Structure

Yongjun Zhao and Sherman S.M. Chow*

Department of Information Engineering
The Chinese University of Hong Kong, Hong Kong
{zy113, sherman}@ie.cuhk.edu.hk

Abstract. Sharing information to others is common nowadays, but the question is with whom to share. To address this problem, we propose the notion of secret transfer with access structure (STAS). STAS is a two-party computation protocol that enables the server to transfer a secret to a client who satisfies the prescribed access structure.
In this paper, we focus on the case of STAS for threshold access structure, i.e. threshold secret transfer (TST). We also discuss how to replace it with linear secret sharing to make the access structure more expressive. Our proposed TST scheme enables a number of applications including a simple construction of oblivious transfer with threshold access control, and (a variant of) threshold private set intersection (t-PSI), which are the first of their kinds in the literature to the best of our knowledge. Moreover, we show that TST is useful a number of applications such as privacy-preserving matchmaking with interesting features.
The underlying primitive of STAS is a variant of oblivious transfer (OT) which we call OT for sparse array. We provide two constructions which are inspired from state-of-the-art PSI techniques including oblivious polynomial evaluation and garbled Bloom filter (GBF). We implemented the more efficient construction and provide its performance evaluation.

Keywords: oblivious transfer, access structure, secret sharing, private set-intersection, garbled Bloom filter, oblivious polynomial evaluation, privacy

1 Introduction

Many people are disseminating information everyday, ranging from short tweet to video. While sharing about oneself is certainly a trend, information can be personal or sensitive. Deciding whom to share becomes an important question, especially when the counter-party may make abusive use of the information. For example, location based services in which sharing information makes sense only when two users are in proximity, or dating apps in which users may only want to share information based on (mutual) interests of the profiled attributes. Yet, the criteria of whom to share may be sensitive as well. Users may not want to reveal their whereabouts or their attributes. Moreover, one may not expect an *exact* match in many cases, *e.g.*, it may be difficult to find someone via dating apps who matches every single desired attributes.

In this paper, we propose the notion of *secret transfer with access structure* (STAS), such that a secret from one user will only be transferred to another user who satisfies the prescribed access structure, with the aim of revealing as little information about the access structure as possible.

To illustrate, STAS can be considered as a general case of threshold private set intersection (PSI), where the secret to be shared is the intersection of two sets of the respective parties, and the access structure is a threshold policy on the two sets, which only reveals the intersection set if the size of the intersection is larger than a pre-defined threshold, Privacy guarantee offered by a typical PSI protocol is that, all the elements outside the intersection remain unknown to the counter-party.

* Sherman Chow is supported by the Early Career Scheme and the Early Career Award of the Research Grants Council, Hong Kong SAR (CUHK 439713), and Direct Grant (4055018) of the Chinese University of Hong Kong.

Computing the intersection of two sets is useful in many scenarios, *e.g.*, two Facebook users may check who are their common friends before accepting add-friend request; two companies want to find the number of common customers before launching a joint promotion; or apps like tworlds which share photos to a random stranger across the globe simply based on the supplied hashtags. A straightforward approach requires two parties to reveal their sets and then compute the result locally. However, the sets might contain valuable information that should not be disclosed for economic reasons or too sensitive to reveal. Revealing only the intersection, while protecting the confidentiality of elements which are not in the intersection, is an important task.

PSI is proposed to solve the above problem. It involves two parties: a client and a server, each holding a private set C and S respectively. At the end of the protocol, the client learns the intersection $C \cap S$ of their sets. The usual treatment in the literature is that the server learns nothing. If both parties need to learn $C \cap S$, they could switch roles and engage in a second PSI instance[1]. In a variant called PSI-CA, only the cardinality $|C \cap S|$ is revealed (*e.g.* [12, 20]). Very efficient PSI protocols have been proposed (*e.g.* [18, 38, 39]).

1.1 PSI with Access Structure

This paper considers a general PSI which only reveals the sets when the intersection satisfies a certain structure. A notable example is threshold PSI (t-PSI), which only reveals the intersection set $C \cap S$ when the *size* of the intersection $|C \cap S|$ is larger than a pre-agreed threshold t. When $t = 0$, t-PSI reduces to PSI. When $t = |C|$, t-PSI reduces to PSI-CA. We consider t-PSI as a natural and useful extension to PSI.

Designing an efficient t-PSI protocol (without resorting to *generic* secure multiparty computation) is not an easy task, as observed by Pinkas *et al.* [39]. Using a generic secure computation protocol is probably not efficient especially when the datasets are large. Constructing t-PSI based on other techniques with better efficiency is non-trivial. To the best of authors' knowledge, there is no such protocol in the literature, not to say access structure more general than threshold policy, *e.g.*, weighted threshold, in which different elements carry different weights counting towards the final threshold. Apart from the technical challenge, we believe that t-PSI is an interesting primitive that deserves further investigation. Other important cryptographic primitives also have their threshold version, *e.g.*, signature [7, 44], password-based authenticated key exchange [35, 40], and attribute-based encryption. Existing threshold cryptosystems have shown their applicability in many scenarios. We foresee the same will hold for t-PSI.

1.2 Technical Overview of Our Results

We focus on threshold secret transfer (TST), as a special case of STAS. TST allows a server to transfer a secret κ to a client if and only if their respective private sets have more than t common elements. We define appropriate security definitions for this new primitive, and provide two constructions, one based on oblivious polynomial evaluation (OPE) [20], and the other based on a variant of garbled Bloom filter (GBF) [18], both were used to construct state-of-art (vanilla) PSI protocols. For building TST, we introduce *oblivious transfer for a sparse array* (OTSA), in which the selection strings are from a large domain. This helps us to achieve privacy in threshold matching. In contrast, typical OT only works on an array indexed by polynomially many numbers.

We first show how to solve a variant of t-PSI problem which we call t-PSI-CA. t-PSI-CA differs from t-PSI as it allows the client to always learn the *size* of the intersection $|C \cap S|$. This may suffice for some application scenarios and actually can also be regarded as a *feature* (see Sec. 8).

With TST, t-PSI-CA is readily achievable. The server and the client engage in a TST with their respective sets. The client will obtain a secret κ only if there are more than t overlappings. Conceptually κ can be considered as a "proof token" to show that the client indeed holds a set containing at least t common elements. We stress that which elements belong to the set remains

[1] In a malicious model, one may require both parties to first commit to the sets and carry out a zero-knowledge proof asserting that both instances are using the same sets as input.

hidden at this point. Then the server appends κ to every element in its set S, and executes another PSI with the client using this new set.

Our construction blueprint is readily extensible to other access structures by replacing the polynomial-based threshold secret sharing to other schemes. For example, we can obtain secret transfer for weighted threshold and weighted t-PSI-CA. We also discuss how to construct STAS by replacing threshold secret sharing with linear secret sharing, which supports more expressive access control policy.

All the constructions in this paper are proven to be secure in the semi-honest model (sometimes called honest-but-curious), in which the adversary is assumed to follow the protocol exactly as specified, but may try to learn as much as possible about the input of the other party. We focus on semi-honest constructions because they are often more efficient than their fully secure counterparts. We leave it as an open question to propose efficient constructions in stronger security model.

1.3 Roadmap

The next section discusses primitives related to TST including PSI. Sec. 3 introduces notations and important building blocks of our protocol. In Sec. 4, we introduce OTSA, which may be of independent interests. We then define TST and construct TST from OTSA in Sec. 5. Our generic construction of t-PSI-CA is presented in Sec. 6 and we provide evaluation results in Sec. 7. Finally, we conclude our paper with further applications and future work.

OPPRF	oblivious permuted pseudorandom function
OTSA	oblivious transfer for a sparse array
TST	threshold secret transfer
$(\mathrm{ST})^2$	strong threshold secret transfer
STAS	secret transfer with access structure
t-PSI	threshold private set-intersection
t-PSI-CA	threshold private set-intersection cardinality

OPE	oblivious polynomial evaluation
OPRF	oblivious pseudorandom function
OT	oblivious transfer
BF	Bloom filter
GBF	garbled Bloom filter
LSSS	linear secret sharing scheme
PSI	private set-intersection
PSI-CA	private set-intersection cardinality

Table 1: List of major acronyms: the upper table contains acronyms introduced in this paper; the lower one contains existing acronyms in the literature.

2 Related Work

2.1 Private Set Intersections

As far as we know, the first rigorous treatment for PSI was done by Freedman *et al.* [20], who proposed a protocol based on *oblivious polynomial evaluation* (OPE). The key idea is that the client uses additive homomorphic encryption to encrypt the coefficients of a polynomial $p(x)$ whose roots are the elements in the set. The server obliviously evaluates the polynomial $rp(x) + x$ for each element $s_i \in S$. The evaluation results are then sent to the client, who could decrypt and

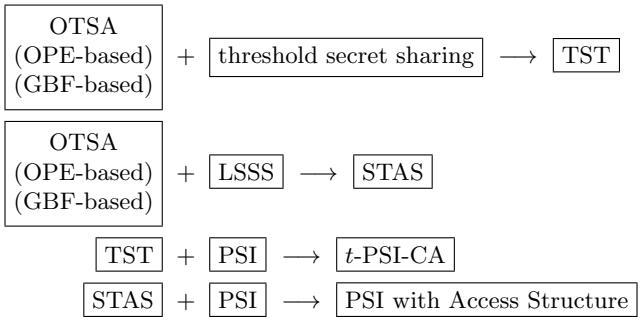


Fig. 1: Roadmap of Our Constructions

compare with its own set. Using similar technique, Kissner and Song [30] constructed multi-party PSI protocol. More efficient PSI protocols in the malicious model are also proposed [15, 23].

Another common approach of constructing PSI is using *oblivious pseudorandom function* (OPRF) [19, 22, 27]. The idea is very simple: the server sends to the client $f_k(s_i)$ for each element $s_i \in S$, where $f(\cdot)$ is a pseudorandom function and k is a random key. The client and the server then engage in an OPRF protocol such that the client learns $f_k(c_i)$ for each $c_i \in C$ while the server learns nothing except the size of C. PSI protocols based on OPRF can achieve linear computational and bandwidth complexity.

Very recently, Dong *et al.* [18] introduced new techniques for PSI. The core component is a variant of Bloom Filter, which they called *garbled Bloom filter* (GBF). We will give a more detailed description of GBF in the next section. Combining GBF and *oblivious transfer* (OT), Dong *et al.* constructed a very efficient PSI in both semi-honest and malicious models. Inspired by their paper, we construct PSI with access structure with linear complexity in the semi-honest model. We remark that their technique of constructing malicious model PSI might be useful in constructing PSI with access structure in the malicious model. Their PSI protocol is further improved [38, 39].

Researchers have considered variants of PSI. Some reveal only the size (cardinality) of intersection but not the set itself [2, 12, 20, 25, 30, 45], which we call *PSI-CA*. Some achieve more features. For example, Camenisch and Zaverucha [11] and De Cristofaro *et al.* [13, 14] considered *authorized PSI* (APSI) which requires the input sets to be signed by a trusted party. Ateniese *et al.* [3] and D'Arco *et al.* [16] proposed *size-hiding PSI* which hides the size of the client's input set. Outsourcing the computation of PSI to an oblivious cloud is considered by Abadi *et al.* [1] and Kerschbaum [29]. Kissner and Song [30] considered threshold set union and its variants in multi-user setting: all n players learn which elements appear more than t times in the union (which is a multiset) of their private input set.

2.2 Fuzzy Vault

A fuzzy vault, introduced by Juels and Sudan [28], allows a server to "lock" a small secret value κ using a set S, such that a client holding another set C which is similar to S could recover κ efficiently. This primitive looks very much like TST, but we would like to highlight the differences here.

We first briefly recall the existing construction [28]: the elements in the server set S are encoded as distinct x-coordinate values. The server selects a random polynomial $p(\cdot)$ that encodes κ in some way (say $p(0) = \kappa$), and evaluates $p(\cdot)$ on these coordinates. To hide κ, the server adds a number of random "noisy" points that do not lie on $p(\cdot)$, and publishes the set of both real and noisy points *in clear*.

If the client holds a set C that substantially overlaps with S, it can identify and use the common x-coordinates to reconstruct (the polynomial and) κ. If the intersection $|C \cap S|$ is not

large enough, then the client is not able to identify enough "correct" x-coordinates to perform polynomial interpolation.

Even if the polynomial (say, of degree k) cannot be reconstructed when the size of the intersection $|C \cap S|$ is k (*i.e.* one more point is needed), an adversary can still launch the following attack to reveal the rest of the set S. Consider the server set is of size n_s, and the number of noisy points is N. The client could simply try reconstructing κ by choosing one of the $(n_s + N - k)$ remaining points. The probability of success is $\frac{n_s - k}{n_s + N - k}$.

One might consider this as a feature: the more similar the client set is, the more likely to recover κ. However, in TST, we aim to impose a *sharp distinction* between "over threshold" and "below threshold". Again, the client only knows that $|C \cap S|$ is not greater than the threshold, but not exactly which elements in C lie in the intersection. To achieve the same guarantee using fuzzy vault, an exponential number of noisy points must be added, which is impractical.

2.3 Attribute-Based Encryption

Attribute-based encryption (ABE) [41] is a generalization of public key encryption, which allows sharing of encrypted content to people according to some prescribed access control policy, without the need of knowing all public keys of those people. Anyone in possession of attributes satisfying the access policy can use their secret keys for decryption. Early ABE schemes suffer from the weakness of revealing the access policy of ciphertexts to everyone, which might be sensitive. Anonymous ABE [31, 36] is then proposed.

One might attempt to implement STAS as follows: encrypt the secret value using anonymous ABE, then send the ciphertext to the receiver. This solution seems plausible. Yet, ABE requires a trusted authority to set up the whole system, and issue secret keys to participants according to their attributes. Such requirement somewhat trivializes the two-party computation since both parties need to trust the authority not to reveal the attributes to others. ABE constructions also typically make extensive use of rather expensive pairing operations.

Furthermore, the anonymity offered by practical anonymous ABE schemes nowadays are not ideal. Having a closer look of existing works, the size of public key and ciphertext in the first scheme [36] are both linear in the size of the attribute universe $\mathcal{U}$. Li *et al.* [34] reduced the public key size to $O(1)$ and the ciphertext size to $O(\log |\mathcal{U}|)$, but unfortunately their scheme is insecure [33]. These two constructions only support limited form of policies. Lai *et al.* [31] constructed an anonymous ABE which supports more expressive policy in the form of LSSS (see Sec. 3.2), yet sacrifices anonymity to some extent. In more details, each attribute is a category-value (*e.g.* Title: Professor, Department: CS) pair in their construction. The LSSS matrix is defined over the categories, and this matrix is public. Only the attribute values are anonymized. That is to say, a policy "(Title: Professor) AND (Department: CS)" is anonymized as "(Title: $*$) AND (Department: $*$)". In contrast, such an access policy is anonymized as "`A` AND `B`" in our STAS construction where `A` and `B` are predicates. To conclude, anonymous ABE is not a good fit for our problem.

3 Preliminary

3.1 Notations

For a binary string x, $|x|$ denotes its length. The i^{th} bit of x is $x[i]$ and $x[i, j]$ denotes $x[i] \dots x[j]$ for $1 \leq i \leq j \leq |x|$. For a finite set S, $|S|$ denotes its size and $s \xleftarrow{\$} S$ denotes picking an element uniformly at random from the set S. For $i \in \mathbb{N}$, we let $[1, i] = \{1, \dots, i\}$. We write $\{s_i\}_n$ as a shorthand for the set $S = \{s_1, \dots, s_n\}$ of n elements. We drop the subscript n if it is clear from context. We denote the security parameter by $\lambda \in \mathbb{N}$ and its unary representation by 1^λ.

Algorithms are polynomial time (PT) and randomized unless otherwise indicated. By $y \xleftarrow{\$} A(x_1, \dots; R)$ we denote running algorithm A on input $x_1, \dots$ using randomness R, and assigning the output to y. We may omit R for brevity.

A probability ensemble indexed by I is a sequence of random variables indexed by a countable index set I. Namely, $X = \{X_i\}_{i\in I}$ where X_i is a random variable. Two distribution ensembles $X = \{X_n\}$ and $Y = \{Y_n\}$ are computationally indistinguishable, denoted by $X \stackrel{c}{\equiv} Y$, if for every probabilistic polynomial-time (PPT) algorithm D, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $n \in \mathbb{N}$,

$$|\Pr[D(X_n, 1^n) = 1] - \Pr[D(Y_n, 1^n) = 1]| \leq \mathsf{negl}(n).$$

We let $\mathbb{G} = \langle g \rangle$ be a group of order p, a λ bit-prime number.

3.2 Secret Sharing

Threshold secret sharing is a fundamental cryptographic primitive and it could be considered as the most basic tool for threshold cryptography. It allows a dealer to split a secret κ into n shares, such that κ can be recovered efficiently with any subset of t or more shares. Any subset of size less than t reveals no information about the secret value. Shamir's secret-sharing scheme [43], which is based on polynomial interpolation, is such a (t, n)-secret sharing scheme. We denote $\mathsf{SecretSharing}_{(t,n)}(\kappa)$ and $\mathsf{Reconstruct}_{(t,n)}(\{\kappa_i\}_{n'})$ as its sharing algorithm and reconstruction algorithm.

When $t = n$, an efficient (n, n)-secret sharing scheme can be obtained from $\oplus$ (XOR) operations [42]. It works by picking random $|\kappa|$-bit strings $\kappa_1, \kappa_2, \ldots, \kappa_{n-1}$ as the first $n-1$ shares. The last share is computed by $\kappa_n = \left(\bigoplus_{i=1}^{n-1} \kappa_i\right) \oplus \kappa$. We will use this simple scheme extensively.

We can extend threshold secret sharing scheme to support more general access structure. Let $\{\kappa_1, \ldots, \kappa_n\}$ be a set of secret shares. An access structure [5] is a collection $\mathbb{A}$ of non-empty subsets of $\{\kappa_1, \ldots, \kappa_n\}$, *i.e.*, $\mathbb{A} \subseteq 2^{\{\kappa_1,\ldots,\kappa_n\}} \setminus \{\varnothing\}$. We denote $\mathsf{SecretSharing}_{\mathbb{A}}(\kappa)$ and $\mathsf{Reconstruct}_{\mathbb{A}}(\{\kappa_i\}_{n'})$ as the sharing algorithm and reconstruction algorithm in a secret sharing scheme implementing access structure $\mathbb{A}$. Any monotone access structure can be realized by a linear secret sharing scheme defined below [5].

Definition 1 (Linear Secret-sharing Schemes (LSSS)) *A secret sharing scheme Π over a set of parties $\mathcal{P}$ is called linear over $\mathbb{Z}_p$ if*

1. *The shares of each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix M called the share-generating matrix for Π. The matrix M has ℓ rows and n columns. For a column vector $v = (\kappa, r_2, \ldots, r_n)$, where $\kappa \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret κ according to Π. The share $(Mv)_x$, the x-th row of Mv, belongs to party $\rho(x)$, where ρ maps $\{1, \ldots, \ell\}$ to $\mathcal{P}$.*

Any LSSS defined as above enjoys the linear reconstruction property as follows. Suppose that Π is an LSSS for access structure $\mathcal{A}$. Let $S \in \mathcal{A}$ be an authorized set, and $I \subseteq \{1, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. There exist constants $\{w_i \in \mathbb{Z}_p\}_{i\in I}$ satisfying $\sum_{i\in I} w_i M_i = (1, 0, \ldots, 0)$, so that if $\{\lambda_i\}$ are valid shares of any secret κ according to Π, then $\sum_{i\in I} w_i \lambda_i = \kappa$. Furthermore, these constants $\{w_i\}$ can be found in time polynomial in the size of the share-generating matrix M. For any unauthorized set, no such constants exist. The LSSS is denoted by (M, ρ), and its size is the number of rows of M.

Access structures can also be represented by monotonic boolean formulas. The techniques of transforming any monotonic boolean formula to LSSS are well known in the literature [5]. One can also convert the boolean formula into an access tree. An access tree of ℓ nodes results in an LSSS matrix of ℓ rows. Readers can refer to [32, appendix] for the conversion algorithm.

3.3 Oblivious Transfer (OT)

Oblivious transfer (OT) is another basic cryptographic building block. It allows the receiver to get only part of the sender's input, while the sender remains oblivious about what the receiver obtains.

Formally, in an OT_ℓ^m protocol, the sender inputs m pairs ℓ-bit strings $(x_{i,0}, x_{i,1})$ $(1 \le i \le m)$ and the receiver inputs an m-bit selection string $b = (b_1, \ldots, b_m)$. At the end of the protocol, the sender learns nothing about b, while the receiver only gets x_{i,b_i} for $1 \le i \le m$. OT protocols in the random oracle model can be very efficient[2].

Basically the above formulation only allows the receiver to choose *1-out-of-2* according to each *bit* in the selection string. Researchers have also considered more general k-out-of-n OT, where the receiver prepares n strings, and the receiver only gets k of them by specifying a set of distinct indexes (in range $[1, n]$ instead of $\{0, 1\}$) of size k.

Looking ahead, we use the notion (which slightly abuses the one defined above) $\mathrm{OT}_{I_s}^{|I_s \cap I_r|}$ for *oblivious transfer for a sparse array* (OTSA). Note that the superscript is a number while the subscript is a set, thus differentiates OTSA from OT_ℓ^m. Roughly speaking, $\mathrm{OT}_{I_s}^{|I_s \cap I_r|}$ can be considered as an $|I_s \cap I_r|$-out-of-$|I_s|$ OT. The meaning behind such superscript and subscript will be more clear after we explained the meaning of I_s and I_r in Sec. 4.1.

3.4 Additive Homomorphic Encryption

We will also use semantically-secure additive homomorphic encryption scheme. One well known example is Paillier's cryptosystem [37]. It supports addition, and multiplication by a constant, without private key sk. Specifically, given two ciphertexts $\mathsf{Enc}_{pk}(m_0)$ and $\mathsf{Enc}_{pk}(m_1)$, there is an efficient operation that can compute $\mathsf{Enc}_{pk}(m_0 + m_1)$; given one ciphertext $\mathsf{Enc}_{pk}(m)$ and a constant c, there is an efficient operation that can produce $\mathsf{Enc}_{pk}(c \cdot m)$. A corollary of these two properties is: given encryptions of the coefficients $a_0, \ldots, a_k$ of a polynomial $p(x)$ of degree k, and a plaintext s, it is possible to compute an encryption of $p(s)$.

3.5 Oblivious Pseudorandom Function

An oblivious pseudorandom function [19] is a two-party protocol between a server S and a client C for securely computing a pseudorandom function $f_k(\cdot)$ under key k known by S only, while the input x is known by C. The client learns $f_k(x)$ while the server learns nothing after their interaction. In this work, we consider the construction given by Jarecki and Liu [27]. It is secure under parallel composition [27] and our second construction relies on this special property. This protocol uses Camenisch-Shoup version [10] of Paillier encryption [37] $(\mathsf{Enc}_{pk}(\cdot), \mathsf{Dec}_{sk}(\cdot))$, which is additive homomorphic, to compute Dodis-Yampolskiy PRF [17] $f_k(x) = g^{\frac{1}{k+x}}$. We sketch the main procedures as follows:

1. The server sends $\mathsf{Enc}_{pk_s}(k)$ to the client.
2. The client chooses a random number r_c, computes $\mathsf{Enc}_{pk_s}(r_c(k + x))$ by the homomorphic property, and then replies with $(c_1, c_2) = (\mathsf{Enc}_{pk_s}(r_c(k + x)), \mathsf{Enc}_{pk_c}(r_c))$.
3. The server decrypts c_1 and computes its inverse $\frac{1}{r_c(k+x)}$. It also chooses a random number r_s and uses the homomorphic property to compute $s_1 = c_2^{\frac{1}{r_c(k+x)}} \cdot \mathsf{Enc}_{pk_c}(-r_s)$. Then it replies with $(s_1, s_2) = (\mathsf{Enc}_{pk_c}(\frac{1}{k+x} - r_s), g^{r_s})$.
4. The client decrypts s_1 to get $\frac{1}{k+x} - r_s$, and computes the final output as $g^{\frac{1}{k+x}} = g^{\frac{1}{k+x} - r_s} \cdot s_2$.

The parallel version of this OPRF can be easily obtained by replacing $r_c, r_s, (c_1, c_2)$ and (s_1, s_2) with $\{r_c^{(i)}\}_{|C|}, \{r_s^{(i)}\}_{|C|}, \{(c_1^{(i)}, c_2^{(i)})\}_{|C|}$ and $\{(s_1^{(i)}, s_2^{(i)})\}_{|C|}$ respectively in Steps 2 to 4. Also note that if the server applies a random permutation Π on $\{(s_1^{(i)}, s_2^{(i)})\}_{|S|}$, the client will still get the same set $\left\{g^{\frac{1}{k+x_i}}\right\}_{|C|}$, but it does not know which $g^{\frac{1}{k+x_i}}$ corresponds to which x_i due to property of PRF. We denote such parallel OPRF with additional permutation step as *oblivious permuted pseudorandom function* (OPPRF).

[2] In the random oracle model, one can reduce expensive public key operations for OT_ℓ^m to that of $\mathrm{OT}_\lambda^\lambda$, where λ is the security parameter [26]. This technique is used to improve efficiency of PSI protocol of Dong *et al.* [18]. Readers are referred to [18, 26] for details.

3.6 Bloom Filters and Garbled Bloom Filters

An (m, n, k, H)-Bloom filter [6] is a compact array of m bits that can represent a set S of at most n elements for efficient set membership testing. It consists of a set of k independent hash functions $H = (h_1, h_2, \ldots, h_k)$ where each h_i maps elements to index numbers in the set $[1, m]$ uniformly.

Initially, all bits in the array are set to 0. To insert an element $x \in S$ into a Bloom filter, the element is hashed using the k hash functions to get k index numbers. The bits at all these indexes in the bit array are set to 1, regardless of its original value. To check if an item y is in S, y is hashed by the k hash functions to get k indexes. If any of the bits at these indexes is 0, we conclude that y is certainly not in S. Otherwise, y is probably in S. So, it never yields a false negative, but there is a small fraction of false positives. The upper bound of the false positive probability [8] is: $\epsilon = p^k \times \left(1 + O\left(\frac{k}{p}\sqrt{\frac{\ln m - k \ln p}{m}}\right)\right)$ where $p = 1 - (1 - 1/m)^{kn}$.

The false positive rate should set to be less than a certain threshold ϵ. It can be shown that the length of the bit array m should be at least $m \geq n \log_2 e \cdot \log_2 1/\epsilon$, and the number of hash functions $k = (m/n) \cdot \ln 2 = \log_2 1/\epsilon$, where e is the base of the natural logarithm. In the rest of this paper, we will stick with these optimal values when we use (garbled) Bloom filter. Specifically, we set the false positive probability $\epsilon = 2^{-\lambda}$ where λ is the security parameter. As a result, $m = \lambda n \log_2 \epsilon$ and $k = \lambda$ in all cases. So we represent a Bloom filter with optimal parameters as an (n, H, λ)-BF.

An (m, n, k, H, λ)-garbled Bloom filter [18] is a variant of Bloom filter introduced by Dong *et al.* that supports efficient private set-intersection. Roughly speaking, a garbled Bloom filter uses an array of λ-bit strings instead of an array of bits in a normal Bloom filter. Initially all m strings in the garbled Bloom filter are set to NULL. To insert an element $x \in \{0,1\}^\lambda$, x is first split into k shares by XOR-based (k, k)-secret sharing. The k^{th} share is placed at location $h_i(x)$. If the location $h_i(x)$ is already occupied due to previous insertion, we *reuse* the string at that location, and adjust the value of subsequent shares accordingly, subject to the constraint that $\bigoplus_{i=1}^{k} GBF[h_i(x)] = x$. Such adjustment is always possible unless all locations $\{h_i(x)\}_{1 \leq i \leq k}$ are all occupied, which corresponds to a false positive. The probability of this happening can be negligible if the parameters are set properly. After inserting all elements in S to the garbled Bloom filter, the undefined slots in the vector are filled with random strings. To check if an item y is in S, y is hashed by the k hash functions, and the strings in those locations are retrieved. If y can be reconstructed from these shares, we conclude that y is surely in S. For brevity, GBF with optimal parameters will be denoted by (n, H, λ)-GBF.

3.7 Semi-honest Secure Computation

We use the following simulation-based definition for security. We consider *static semi-honest* adversaries [21], which can control one of the two parties and assumed to follow the protocol specification exactly. However, it may try to learn more information about the other party's input.

A two-party protocol π computes a function that maps a pair of inputs to a pair of outputs $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. For every pair of inputs $x, y \in \{0,1\}^*$, the output-pair is a random variable $(f_1(x, y), f_2(x, y))$. The first party obtains $f_1(x, y)$ and the second party obtains $f_2(x, y)$.

In the semi-honest model, a protocol π is secure if whatever can be computed by a party in the protocol can be obtained from its input and output only. This is formalized by the simulation paradigm. We require a party's *view* in a protocol execution to be simulatable given only its input and output. The view of the party i during an execution of π on input (x, y) is denoted by $\mathsf{View}_i^\pi(x, y) = (w, r^i, m_1^i, \ldots, m_t^i)$ where $w \in (x, y)$ is the input of i, r^i is i's internal random coin tosses and m_j^i denotes the j^{th} message that it received.

Definition 2 (Semi-honest Model) *Let $f = (f_1, f_2)$ be a deterministic function. Protocol π is said to be securely computes f in the presence of static semi-honest adversaries if there exists*

probabilistic polynomial-time algorithms Sim_1 *and* Sim_2 *such that*

$$\{\mathsf{Sim}_1(x, f_1(x,y))\}_{x,y} \stackrel{c}{\equiv} \{\mathsf{View}_1^{\pi}(x,y)\}_{x,y}$$
$$\{\mathsf{Sim}_2(y, f_2(x,y))\}_{x,y} \stackrel{c}{\equiv} \{\mathsf{View}_2^{\pi}(x,y)\}_{x,y}$$

4 OT for a Sparse Array

We propose oblivious transfer for a sparse array (OTSA) as a building block for secret transfer with access structure (STAS). We first provide a formal definition for OTSA and then sketch the main design idea before presenting two concrete constructions.

Although in an abstract level, the two constructions both make use of the idea of *PSI with data transfer*, we believe the first OPE-based construction is conceptually simpler and easier to understand than the second GBF-based one for readers who do not have prior knowledge in recent advances of PSI, in particular, (garbled) Bloom filter. As a consequence, the OPE-based one is worth mentioning, even though its computation complexity is quite high (quadratic in the number of elements). The second construction achieves linear complexity using more recent techniques (*i.e.*, combining garbled Bloom filter and oblivious transfer) in the area of PSI. In Sec. 7, we implement the second (more practical) construction and evaluate its performance.

4.1 Definitions

OTSA is a new variant of the original OT concept. The sender holds an array of n_s elements $\{e_i\}_{1\le i\le n_s}$ (from a certain domain), which are associated with n_s distinct indices $I_s = \{s_1, \ldots, s_{n_s}\}$, where each s_j for $1 \le j \le n_s$ is an element from the domain $\mathcal{D}$, not necessarily $[1, n_s]$. The receiver specifies a set of indices $I_r = \{r_1, \ldots, r_{n_r}\}$, not necessarily a subset of I_s, and asks for retrieving the associated elements. We assume the domain $\mathcal{D}$ is large, *i.e.*, $n_s \ll |\mathcal{D}|$ and $n_r \ll |\mathcal{D}|$, so not every index $r_j \in I_r$ specified by the receiver indeed has an element from the server associated with it. Our OTSA satisfies the following properties:

- **Correctness:** The receiver retrieves $E' = \{e'_j\}_{1\le j\le n_r}$, where $e'_j = e_j$ if $s_j \in I_s \cap I_r$, or e'_j is an element randomly picked from a pre-defined domain if $s_j \notin I_s \cap I_r$.
- **Receiver privacy:** The sender learns nothing about I_r.
- **Sender array privacy:** The receiver learns nothing about e_j's whose index $s_j \notin I_s \cap I_r$.
- **Sender indices privacy:** The receiver learns nothing about I_s except $|I_s \cap I_r|$.

Distinction from Normal OT Normal OT typically represents the server indices by $[1, n_s]$. A possible way to use OT to realize the functionality we want to achieve is to require the server to publish a 1-to-1 mapping between $[1, n_s]$ and $\{s_1, \ldots, s_{n_s}\}$, but this breaks the sender indices privacy. Another possibility is to simply use normal OT in which the server holds $|\mathcal{D}|$ elements. This will incur $O(|\mathcal{D}|)$ communication complexity. Ideally, we shoot for $O(\max(n_r, n_s))$. We remark that our primitive implies the normal OT since one can simply set $I_s = [1, n_s]$.

More on Correctness Requirement For OTSA to be useful, the client should have the ability to differentiate a correct data element from a random string. In case the application (*e.g.*, in TST) may not make it apparent, it can be achieved by asking the server to append a special symbol to e_i for recognition, or simply publish $\{H(e_i)\}$ where $H(\cdot)$ is a cryptographic (one-way) hash function.

On Sender Indices Privacy When a correct data element is distinguishable from a random string, the receiver will learn from the protocol the cardinality of the intersection, *i.e.*, $|I_s \cap I_r|$, according to the functionality requirement; and hence we cannot afford to protect it. Yet, we note that the receiver will not know what exactly is the intersection set $I_s \cap I_r$. We will show that this level of privacy suffices for our applications.

Security Using the language of secure two-party computation, we define OT for a sparse array as the following functionality:

Definition 3 (Oblivious Transfer for a Sparse Array (OTSA)) *OTSA is a two-party computation protocol that implements the following functionality*

$$f(x, y) = (\perp, E')$$

where the server's input $x = (E, I_s)$ consists of two sets of the same size n_s, one being the (multi-)set of data elements $E = \{e_1, \ldots, e_{n_s}\}$, another being the sender's index set $I_s = \{s_1, \ldots, s_{n_s}\}$. The receiver's input $y = I_r$ is a set of indices of size n_r. The output of the receiver E' is a subset of E, such that $e_j \in E'$ if and only if its index $s_j \in I_s \cap I_r$.

We assume that the size of index sets, namely n_s and n_r, are publicly known by both parties.

We say that a protocol π is an OTSA in the semi-honest model if it securely implements the above function f in the semi-honest model. As discussed in Sec.3.3, we denote such a protocol by $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$.

4.2 Construction Idea

Our two constructions borrow ideas from the area of private set-intersection. One is based on oblivious polynomial evaluation (OPE); the other one is based on a variant of garbled Bloom filter. It has been observed that the PSI protocol from OPE can actually allow the transfer of auxiliary information. Hence, we exploit this storage capacity to store the data elements for our OTSA. Corresponding, we make the same observation for the GBF-based PSI. We note that while both PSI protocols share the same property, their construction ideas are quite different.

4.3 OPE-based $\mathbf{OT}_{\boldsymbol{I_r}}^{|\boldsymbol{I_s} \cap \boldsymbol{I_r}|}$

Now we describe our first construction, which is based on oblivious polynomial evaluation, in Fig. 2.

Protocol: OPE-based $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$

Input: The receiver's input is an index set $I_r = \{r_1, \ldots, r_{n_r}\}$. The sender's input is an index set $I_s = \{s_1, \ldots, s_{n_s}\}$ and a data set $E = \{e_1, \ldots, e_{n_s}\}$.

1. The receiver chooses a key pair (pk, sk) for a semantic secure additive homomorphic encryption scheme $(\mathsf{Enc}_{pk}, \mathsf{Dec}_{sk})$, and publishes pk.
2. The receiver computes the coefficients of the polynomial $p(x) = \sum_{i=0}^{n_r} a_i x^i$ of degree n_r with roots being elements in the *selection strings* set I_r.
3. The receiver encrypts each of the $(n_r + 1)$ coefficients by the additive homomorphic encryption scheme and gives the sender the resulting set of ciphertexts, $\{\mathsf{Enc}_{pk}(a_i)\}$.
4. For each index $s_i \in I_s$, the sender:
   (a) Uses the homomorphic property to evaluate the encrypted polynomial at s_i, namely computes $\mathsf{Enc}_{pk}(p(s_i))$.
   (b) Chooses a random value r and computes $\mathsf{Enc}_{pk}(rp(s_i) + e_i)$.
5. The sender sends a permutation of these n_s ciphertexts to the client.
6. The receiver decrypts all n_s ciphertexts received, picks up the set of valid elements.

Fig. 2: Protocol: OPE-based $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$

The correctness of this OPE-based $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ is straightforward: if $s_i \in I_r \cap I_s$, then $rp(s_i)+e_i = e_i$, meaning that the receiver successfully received one element; otherwise $rp(s_i) + e_i$ will be a random string containing no useful information about e_i. More formally, we assert the security of the above protocol in the following theorem:

Theorem 1 *The protocol in Fig. 2 securely implements the function f in Sec. 4 in the semi-honest model.*

Proof. Constructing an ideal world sender SIM_s will be easy, since the sender's view only contains pk and $\{\mathsf{Enc}_{pk}(a_i)\}$. The first one is simply a valid public key, while the second one can be simulated by encryption of random strings due to semantic security of the encryption scheme.

Constructing an ideal world receiver SIM_r from a malicious receiver R^* in the real world can be done in the following way: SIM_r first gets the public key pk and the set $\Gamma = \{\gamma_i\}$ where $|\Gamma| = |I_r| + 1$. On getting I_r from the receiver R^*, it sends $((E, I_s), I_r)$ to the ideal functionality $\mathcal{F}_{OTAS}$, which outputs $E' = \{e_j\}_{s_j \in I_s \cap I_r}$ to the receiver, on the ideal world sender's input (E, I_s) and SIM_r's input I_r. Then SIM_r first gives the sender the set of ciphertexts by encrypting the coefficient of a polynomial constructed from I_r as in the real world, then sends to R^* a permutation of the following set of ciphertext: (1) among $|I_s \cap I_r|$ of them, each encrypts e_j, (2) the rest of them just encrypt random strings.

4.4 GBF*-based $\mathbf{OT}_{I_r}^{|I_s \cap I_r|}$

Our second construction is based on a variant of garbled Bloom filter (GBF), which we call it GBF*. An (n, H, λ)-$\mathrm{GBF}^*_{X,I}$ stores a secret set X using another set I of the same size n. In the original GBF, each element $x_j \in X$ is first split into $k = \lambda$ shares and then these shares are placed at locations $h_1(x_j), \ldots, h_k(x_j)$. While in our $\mathrm{GBF}^*_{X,I}$, each $x_j \in X$ is split and placed at locations defined by $h_1(i_j), \ldots, h_k(i_j)$, where $i_j \in I$. Namely, the set I "indexes" the locations to place X. When we query $\mathrm{GBF}^*_{X,I}$ using some element $i' \in I$, then $\mathrm{GBF}^*_{X,I}$ returns the corresponding element x'. If $i' \notin I$, then $\mathrm{GBF}^*_{X,I}$ returns a uniformly random string.

The $\mathsf{BuildGBF}^*$, $\mathsf{QueryGBF}^*$, and $\mathsf{GBF}^*\mathsf{Intersection}$ algorithms constituting our GBF* are listed as follows:

It is easy to see that Algorithm 1 fails only when emptySlot remains unchanged before line 22. The false positive of GBF* means when querying $\mathrm{GBF}^*_{X,I}$ with some index $i' \notin I$, Algorithm 2 returns some element $x \in X$. Following the existing analysis [18], we have with the following theorems:

Theorem 2 *Algorithm 1 fails with probability* $\mathsf{negl}(\lambda)$.

Theorem 3 *The false positive probability is* $\mathsf{negl}(\lambda)$.

The underlying idea of Algorithm 3 is very similar to the $\mathsf{GBFIntersection}$ algorithm [18]. Thus Algorithm 3 inherits the corresponding theorems for $\mathsf{GBFIntersection}$ algorithm. We omit the largely repeated proofs for the page limit.

Theorem 4 *For $GBF^*_{\tilde{S},I\cap I'}$ produced by Algorithm 3. Let a_ℓ be the event that $GBF^*_{\tilde{S},I\cap I'}[h_\ell(i_j)]$ equals the ℓ-th share of s_j, $1 \le \ell \le k$; then (i) $\forall i_j \in I \cap I' : \Pr[a_1 \wedge \ldots \wedge a_\ell] = 1$ (ii) $\forall i_j \notin I \cap I' : \Pr[a_1 \wedge \ldots \wedge a_k]$ is* $\mathsf{negl}(k)$.

Theorem 5 *Given sets I, I' and their intersection $I \cap I'$, let $\tilde{S} \subseteq S$ be a set such that $s_j \in \tilde{S}$ if and only if $i_j \in I \cap I'$. Let $GBF^*_{\tilde{S},I\cap I'}$ be the output of the Algorithm 3 from $GBF^*_{S,I}$ and $BF_{I'}$, let $GBF^{*\prime}_{\tilde{S},I\cap I'}$ be another GBF* produced by Algorithm 1 using $\tilde{S}$ and $I \cap I'$, then $GBF^*_{\tilde{S},I\cap I'} \overset{c}{\equiv} GBF^{*\prime}_{\tilde{S},I\cap I'}$.*

Algorithm 1 $\mathsf{BuildGBF}^*(n, H, \lambda)$

Input: A secret set X, an indexing set I, n, λ, and λ uniform hash functions $H = \{h_1, \ldots, h_\lambda\}$
Output: An (n, H, λ)-$GBF^*_{X,I}$

procedure
  Set $m = \lambda n \log_2 e$
  $GBF^*_{X,I}$ = new m-element array of λ-bit strings
  for $j \leftarrow 1, m$ **do**
    $GBF^*_{X,I}[j]$ = NULL
  end for
  for each $x_j \in X$ **do**
    emptySlot $= -1$, finShare $= x_j$
    for $\ell \leftarrow 1, \lambda$ **do**
      $index = h_\ell(i_j)$;
      if $GBF^*_{X,I}[index] ==$ NULL **then**
        if emptySlot$== -1$ **then**
          emptySlot $= index$
        else
          $GBF^*_{X,I}[index] \xleftarrow{\$} \{0,1\}^\lambda$
          finShare=finShare$\oplus$ $GBF^*_{X,I}[index]$
        end if
      else
        finShare=finShare $\oplus$ $GBF^*_{X,I}[index]$
      end if
    end for
    $GBF^*_{X,I}$[emptySlot] = finShare
  end for
  for $j \leftarrow 1, m$ **do**
    if $GBF^*_{X,I}[j] ==$ NULL **then**
      $GBF^*_{X,I}[j] \xleftarrow{\$} \{0,1\}^\lambda$
    end if
  end for
  return $GBF^*_{X,I}$
end procedure

Now we are ready to detail the procedure of our second $\mathrm{OT}^{|I_s \cap I_r|}_{I_r}$ construction in Fig. 3. Note that in the second step, the receiver's set I_r is transformed into a pseudorandom set I'_r. We implement this transformation using an OPPRF protocol (Sec. 3.5) instead of a normal OPRF because we need to hide the one-to-one correspondence between r_j and r'_j from the receiver. Note that our scheme is also modular since any secure OPPRF suffices.

Theorem 6 *The protocol in Fig. 3 securely implements the function f in Sec. 4 in the semi-honest model.*

Proof. (sketch). The proof mostly follows that of [18, Theorem 7], except that the simulator needs to randomly generate a set of indices $\tilde{I}$ of size $|I_s \cap I_r|$, and uses it as the additional input to construct $GBF^{*\pi}_{\tilde{E},\tilde{I}}$, which is computationally indistinguishable from $GBF^*_{\tilde{E},I'_r \cap I'_s}$ by the security of PRF, and thus indistinguishable from $GBF^{*\pi}_{\tilde{E},I'_s \cap I'_r}$ by Theorem 5 above.

5 (Threshold) Secret Transfer

We now discuss how to make use of OTSA to construct *secret transfer with access structure* (STAS). STAS is a two-party computation protocol that allows the client to receive a secret transferred from the server, if the client satisfies the prescribed access structure. For illustration purpose, below we first define STAS with a simple threshold access structure. We call it threshold

Algorithm 2 $\mathsf{QueryGBF}^*(GBF^*_{X,I}, i', k, H)$

Input: A (n, H, λ)-secret embedding garbled Bloom filter $GBF^*_{X,I}$, $\lambda, H = \{h_1, \ldots, h_\lambda\}$
Output: An element $x \in X$ if $i' \in I$, a random string otherwise

procedure
  $\tilde{x} = \{0\}^\lambda$
  for $\ell \leftarrow 1, \lambda$ **do**
    $index = h_\ell(i')$
    $\tilde{x} = \tilde{x} \oplus GBF^*_{X,I}[index]$
  end for
  return $\tilde{x}$
end procedure

Algorithm 3 $\mathsf{GBF}^*\mathsf{Intersection}(GBF^*_{X,I}, BF_{I'}, m)$

Input: An (n, H, λ)-secret embedding garbled Bloom filter $GBF^*_{X,I}$, an (n, H, λ)-Bloom filter $BF_{I'}$
Output: (n, H, λ)-$GBF^*_{\tilde{X},I\cap I'}$

procedure
  Set $m = \lambda n \log_2 e$
  $GBF^*_{\tilde{X},I\cap I'}$ = new m-element array of λ-bit strings
  for $j \leftarrow 1, m$ **do**
    if $BF_{I'}[j] == 1$ **then**
      $GBF^*_{\tilde{X},I\cap I'}[j] = GBF^*_{X,I}[j]$
    else
      $GBF^*_{\tilde{X},I\cap I'}[j] \xleftarrow{\$} \{0,1\}^\lambda$
    end if
  end for
  return $GBF^*_{\tilde{X},I\cap I'}$
end procedure

secret transfer (TST). Next we demonstrate how to easily extend TST to support general access structure, namely STAS.

5.1 Definition

Using the language of secure two-party computation, we define TST as the following functionality:

Definition 4 (Threshold secret transfer (TST)) *TST is a two-party computation protocol that implements the following functionality*

$$f(x,y) = \begin{cases} (\perp, \kappa \text{ and } |C \cap S|) & \text{if } |C \cap S| \geq t \\ (\perp, |C \cap S|) & \text{otherwise} \end{cases}$$

where the server's input $x = (\kappa, S)$ and the client's input $y = C$.

Above definition always leaks the intersection size to the client. This is due to the technical difficulty for simulation to the client without this knowledge. More discussion will be given later. For the sake of completeness, we give the stronger definition.

Definition 5 (Strong threshold secret transfer ($(\mathbf{ST})^2$)) *$(ST)^2$ is a two-party computation protocol that implements the following functionality:*

$$f'(x,y) = \begin{cases} (\perp, \kappa) & \text{if } |C \cap S| \geq t \\ (\perp, \perp) & \text{otherwise} \end{cases}$$

where the server's input $x = (\kappa, S)$ and the client's input $y = C$.

Protocol: GBF* based $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$

Input: The receiver's input is an index set $I_r = \{r_1, \ldots, r_{n_r}\}$, the sender's input is an index set $I_s = \{s_1, \ldots, s_{n_s}\}$ and a data set $E = \{e_1, \ldots, e_{n_s}\}$.

1. The sender chooses a random key k_{PRF} for Dodis-Yampolskiy pseudorandom function $f_{k_{\mathsf{PRF}}}(\cdot)$.
2. The receiver and the sender engage in an OPPRF protocol using I_r and k_{PRF} as input respectively, such that the sender learns nothing while the receiver obtains $I'_r = \{r'_j\}$ where $r'_j = f_{k_{\mathsf{PRF}}}(r_j)$ for all j. The sender also converts its own private set I_s into I'_s in similar manner.
3. The receiver generates an (n, H, λ)-$BF_{I'_r}$ that encodes its private set I'_r, the sender invokes Algorithm 1 to generate a (n, H, λ)-GBF^*_{E,I'_s}.
4. The receiver uses $BF_{I'_r}$ as the *selection strings* and acts as the receiver in an OT_λ^m protocol.
5. The sender sends m pairs of λ-bit strings $(x_{i,0}, x_{i,1})(1 \leq i \leq m)$ in OT_λ^m, where $x_{i,0}$ is a uniformly random string and $x_{i,1}$ is $GBF^*_{E,I'_s}[i]$. For $1 \leq i \leq m$, if $BF_{I'_r}[i]$ is 0, then the receiver receives a random string; if $BF_{I'_r}[i]$ is 1 it receives $GBF^*_{E,I'_s}[i]$. The result obtained by the client is denoted by $GBF^{*\pi}_{\tilde{E},I'_s \cap I'_r}$.
6. The receiver queries $GBF^{*\pi}_{E,I'_s \cap I'_r}$ using its own set I'_r, and picks up the set of valid element $\{\tilde{E}\}$.

Fig. 3: Protocol: GBF*-based $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$

In $(\mathrm{ST})^2$, the server remains oblivious about the client's input set, but the client only learns if $|C \cap S| \geq t$ or not.

5.2 TST Construction from $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$

The basic idea behind the construction is to split κ into $K = \{\kappa_i\}$ using $(t, |S|)$-secret sharing scheme. The server and the client then engage in an $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ protocol with the server acting as sender using secret input set K and indexing set S, and the client acting as the receiver using C as input set. By the security of $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$, the client receives only a subset of secret shares of κ corresponding to elements in $|C \cap S|$. The security of TST naturally follows from that of $(t, |S|)$-secret sharing scheme.

The detailed construction is given in Fig. 4.

Protocol: TST from $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$

Input: The client's input is a set $C = \{c_i\}_{n_c}$, the server's input is a set $S = \{s_i\}_{n_s}$ and a secret value κ.

1. The server computes the set of shares $K = \{\kappa_i\} \xleftarrow{\$} \mathsf{SecretSharing}_{(t,n_s)}(\kappa)$.
2. The server and the client engage in a $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ protocol. The server acts as the sender with input $x = (K, S)$. The client acts as the receiver with input C.
3. Let $\tilde{K}$ denote the set of the client's collected secret shares from step 1. If $|\tilde{K}| \geq t$, then outputs $\kappa = \mathsf{Reconstruct}_{(t,n_s)}(\tilde{K})$ and $|\tilde{K}|$ locally. Otherwise output $|K|$ only.

Fig. 4: TST Construction from $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$

Theorem 7 *If the underlying secret sharing scheme and the $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ protocols are secure, then the protocol defined in Fig. 4 securely evaluates the TST functionality.*

Proof. (sketch). If the $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ is secure, the simulators for both sides exist. We can use them as subroutines to construct the simulator for the whole protocol.

Server's view: We construct a simulator Sim_S, when given the server's private input and output, simulates the server's view View_S of a real protocol execution. View_S contains the input set S, the secret value κ, the random coins, and the view of $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$. The simulator Sim_S selects random coins r^s uniformly random, and also computes $K' = \{\kappa'_i\} \xleftarrow{\$} \mathsf{SecretSharing}_{(t,n_s)}(\kappa)$. Then Sim_S invokes the simulator for $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ to obtain $\mathsf{Sim}_{\mathtt{sx}}^{OT}((K', S), \perp)$. Finally, Sim_S outputs $(S, K', r^s, \mathsf{Sim}_{\mathtt{sx}}^{OT}((K', S), \perp))$ as the simulated view. Because K' is generated in the same way as in the real protocol, it follows that $\mathsf{Sim}_{\mathtt{sx}}^{OT}((K', S), \perp) \stackrel{c}{\equiv} \mathsf{Sim}_{\mathtt{sx}}^{OT}((K, S), \perp)$. Thus by the security of $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$, the simulated view should be indistinguishable from View_S.

Client's view: We construct a simulator Sim_C, when given the client's private input and output, simulates the client's view View_C of a real protocol execution. View_C contains the input set C, the random coins, and the view of $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$.

- If $|C \cap S| \geq t$, the simulator Sim_C is given $C, \kappa, |K| = |C \cap S|$ as input. Sim_C picks coins r^c uniformly random, and also computes $K' \xleftarrow{\$} \mathsf{SecretSharing}_{(t,n_s)}(\kappa)$. Sim_C then selects a random subset $\tilde{K'} \subseteq K'$ of size $|K|$ uniformly at random. Then Sim_C invokes the simulator for $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ to obtain $\mathsf{Sim}_{\mathtt{rx}}^{OT}(C, (\tilde{K'}, |K|))$. Sim_C outputs $(C, r^c, \mathsf{Sim}_{\mathtt{rx}}^{OT}(C, (\tilde{K'}, |K|)))$ as the simulated view. Since K' and $\tilde{K'}$ are generated in the same way as K and $\tilde{K}$ in the real protocol, $\mathsf{Sim}_{\mathtt{rx}}^{OT}(C, (\tilde{K'}, |K|))$ is identically distributed as $\mathsf{Sim}_{\mathtt{rx}}^{OT}(C, (\tilde{K}, |K|))$. Thus by the security of $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$, the simulated view should be indistinguishable from View_C.
- If $|C \cap S| < t$, Sim_C is given only $C, |K| = |C \cap S|$ as input. This time Sim_C selects a random κ', and computes $K'' \xleftarrow{\$} \mathsf{SecretSharing}_{(t,n_s)}(\kappa')$. Sim_C then selects a random subset $\tilde{K''} \subseteq K''$ of size $|K| < t$ uniformly at random. Sim_C outputs $(C, r^c, \mathsf{Sim}_{\mathtt{rx}}^{OT}(C, (\tilde{K''}, |K|)))$ as the simulated view. To see that above simulated view works well, notice that by the security of (t, n_s)-secret sharing scheme, both $\tilde{K''}$ generated by Sim_C and $\tilde{K}$ received in the real protocol execution leak no information about the original value κ' and κ respectively. Thus $\tilde{K''}$ and $\tilde{K}$ are computationally indistinguishable. Therefore $\mathsf{Sim}_{\mathtt{rx}}^{OT}(C, (\tilde{K''}, |K|))$ simulates the view of $\mathrm{OT}_{I_r}^{|I_s \cap I_r|}$ perfectly.

In both cases, the client's view can be simulated. Combining the result for the server's view, we conclude that the protocol in Fig. 4 is secure in the semi-honest model.

5.3 Transferring Multiple Secrets

Our exposition only considers transferring the secret shares of a single secret. Yet, it is possible to store multiple shares of the same secret in the same slot for supporting weighted TST. One step further, it is also possible to store multiple share of *different* secrets in the same slots. However, since the capacity of each slot is limited. One may need to resort to a hybrid approach where symmetric keys are stored in the TST, while these symmetric keys can in turn unlock the corresponding ciphertext encrypting multiple shares.

5.4 Extending to General Access Structure

It is not hard to see that the threshold access structure of TST comes directly from the underlying threshold secret sharing scheme $(\mathsf{SecretSharing}_{(t,n_s)}, \mathsf{Reconstruct}_{(t,n)})$. If we replace it with other secret sharing scheme with different access structure $(\mathsf{SecretSharing}_{\mathbb{A}}, \mathsf{Reconstruct}_{\mathbb{A}})$, the TST construction will be readily transformed into STAS with access structure $\mathbb{A}$. The proof strategy remains mostly unchanged.

In particular, linear secret sharing [5] fits with our design well. We do not need the usual mapping ρ from attributes to row number of the matrix, since our underlying OTSA supports a sparse array with indices from a large domain.

Recall that in LSSS (see Def. 1), the share generating matrix M is public. The secret value κ is embedded in a column vector $v = (\kappa, r_2, \ldots, r_n)$, and party $\rho(x)$ gets the share $(Mv)_x$. In STAS, the server also needs to publish M. Moreover, it explicitly appends an index x to each share $(Mv)_x$, so that the client knows how to calculate constants $\{w_i \in \mathbb{Z}_p\}_{i \in |I_s \cap I_r|}$ according to M in the reconstruction phase. Exposing M in clear reveals some information about the access structure, *i.e.*, the shape of the access tree. However, we would like to stress that by the security property (sender's indices privacy) of the underlying OTSA, the client does not know the correspondence between the elements in its secret set C and the leave nodes of the access tree.

5.5 Discussion on Leaking Intersection Size

From a theoretical point of view, knowledge of both C and $|C \cap S|$ allows one to infer some information about S, especially when $|C| \approx |C \cap S|$. For instance, if C is of size 100 and the client learns that $|C \cap S| = 70$. The client is able to conclude that many elements in C are also in S. Moreover, if the client can interact with the server multiple times, it can change C by one element each time and monitor how $|C \cap S|$ changes accordingly, which will eventually lead to S.

From a practical standpoint, such leakage is acceptable because the aforementioned probing attack can be mitigated by limiting the number of interactions. Moreover, in the next section we will see an immediate application of TST (resp. STAS), *i.e.* generic t-PSI-CA (resp. PSI with access structure) construction from TST (resp. STAS). The fact that TST always leaks the cardinality of intersection hinders us from obtaining t-PSI. Nevertheless, we believe t-PSI-CA is still a useful primitive and it is the first of its kind in the literature. Finally, we remark that, when the cardinality of the intersection is not exceeding the threshold, t-PSI-CA is the same as PSI-CA; otherwise t-PSI-CA works as normal PSI.

6 Threshold-PSI-CA Protocol

With all necessary building blocks at hand, we are now ready to formally introduce t-PSI(-CA). Again, we first formally define functionality of t-PSI-CA and t-PSI, then describe how to use TST to realize t-PSI-CA.

6.1 Definitions

Definition 6 (Threshold Private Set-intersection (t-PSI)) *t-PSI is a two-party computation protocol that implements the following functionality*

$$f(x, y) = \begin{cases} (\perp, C \cap S) & \text{if } |C \cap S| \geq t \\ (\perp, \perp) & \text{otherwise} \end{cases}$$

where the server's input $x = S$ and the client's input $y = C$.

Definition 7 (Threshold Private Set-intersection (t-PSI-CA)) *t-PSI-CA is a two-party computation protocol that implements the following functionality*

$$f(x, y) = \begin{cases} (\perp, C \cap S) & \text{if } |C \cap S| \geq t \\ (\perp, |C \cap S|) & \text{otherwise} \end{cases}$$

where the server's input $x = S$ and the client's input $y = C$.

We assume threshold t is known by the client beforehand.

6.2 Generic t-PSI-CA Construction from TST

The idea is simple: the client and the server first engage in a TST protocol, such that the client learns κ if $|C \cap S| \geq t$; then they engage in a normal PSI protocol, in which the server and the client uses $S^\kappa = \{s_i||\kappa\}_{n_s}$ and $C^\kappa = \{c_i||\kappa\}_{n_c}$ as input respectively. In case $|C \cap S| < t$, the client chooses a random κ' uniformly at random and use $C^\kappa = \{c_i||\kappa'\}_{n_c}$ instead. The correctness of the above idea is straightforward. What we need to prove is its security. To this end, we first formally describe the above construction in Fig. 5.

Protocol: t-PSI-CA

Input: The client's input is a set $C = \{c_i\}_{n_c}$, the server's input is a set $S = \{s_i\}_{n_s}$.

1. The server selects κ uniformly at random from $\{0,1\}^\lambda$.
2. The client and the server engage in a secure TST protocol π^{TST} with the server's input being (κ, S) and the client's input being C.
3. The server computes $S^\kappa = \{s_i||\kappa\}_{n_s}$.
4. If the client obtains κ from π^{TST}, then it computes $C^\kappa = \{c_i||\kappa\}_{n_c}$. Otherwise it samples a uniformly random κ', and computes C^κ as $\{c_i||\kappa'\}_{n_c}$.
5. The client and the server engage in a secure PSI protocol π^{PSI} with the server's input being S^κ and the client's input being C^κ.

Fig. 5: Protocol: t-PSI-CA

Theorem 8 *Let π^{TST} be a secure two-party computation protocol that securely implements the function f defined in Sec. 5 in the semi-honest model. Let π^{PSI} be a secure PSI protocol in the semi-honest model. Then the protocol defined in Fig. 5 securely evaluates the t-PSI-CA functionality.*

Proof. (sketch) If π^{TST} and π^{PSI} are secure, there exist the simulators for the participants in both protocols. We can use them as subroutines to construct our simulators.

Server's view: The view of a real protocol execution contains the input set S, the random coins, the view of π^{TST}, the view of π^{PSI} ($\mathsf{View}_S = (S, r^s, \mathsf{View}^{\pi^{\mathsf{TST}}}_{\mathtt{svr}}, \mathsf{View}^{\pi^{\mathsf{PSI}}}_{\mathtt{svr}})$). Given S, the simulator Sim_S pick coins r^s uniformly at random, chooses κ uniformly at random from $\{0,1\}^\lambda$, computes $S^\kappa = \{s_i||\kappa\}_{n_s}$. Sim_S then invokes the simulator for the underlying protocols to obtain $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{svr}}((\kappa, S), \perp)$ and $\mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{svr}}(S^\kappa, \perp)$. Sim_S outputs $(S, r^s, \mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{svr}}((\kappa, S), \perp), \mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{svr}}(S^\kappa, \perp))$ as the simulated view. Because κ is identically distributed as in the real execution, so will S^κ. Thus by the security of π^{TST} and π^{PSI}, $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{svr}}((\kappa, S), \perp), \mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{svr}}(S^\kappa, \perp)$ are computationally indistinguishable from $\mathsf{View}^{\pi^{\mathsf{TST}}}_{\mathtt{svr}}, \mathsf{View}^{\pi^{\mathsf{PSI}}}_{\mathtt{svr}}$.

Client's view: The view of a real protocol execution contains the input set C, the random coins, the view of π^{TST}, the view of π^{PSI} ($\mathsf{View}_C = (C, r^c, \mathsf{View}^{\pi^{\mathsf{TST}}}_{\mathtt{clt}}, \mathsf{View}^{\pi^{\mathsf{PSI}}}_{\mathtt{clt}})$).

- If $|C \cap S| \geq t$, the simulator Sim_C is given $C, C \cap S$ as input. Sim_C selects κ' from $\{0,1\}^\lambda$ and coins r^c uniformly at random, and computes $C^{\kappa'}$ and $C^{\kappa'} \cap S^{\kappa'}$ accordingly. Sim_C invokes the simulator for π^{PSI} and π^{TST} to obtain respectively the simulated views $\mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{clt}}(C^{\kappa'}, C^{\kappa'} \cap S^{\kappa'})$ and $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{clt}}(C, (\kappa', |C \cap S|))$. Sim_C then outputs $C, r^c, \mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{clt}}(C^{\kappa'}, C^{\kappa'} \cap S^{\kappa'})$ and $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{clt}}(C, (\kappa', |C \cap S|))$ as the simulated view. Because κ' is identically distributed as κ (which is selected uniformly at random by the server in the real protocol), $\mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{clt}}(C^{\kappa'}, C^{\kappa'} \cap S^{\kappa'})$ and $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{clt}}(C, (\kappa', |C \cap S|))$ are identically distributed as $\mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{clt}}(C^{\kappa}, C^{\kappa} \cap S^{\kappa})$ and $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{clt}}(C, (\kappa, |C \cap S|))$. Then by the security of π^{PSI} and π^{TST}, the simulated view is computationally indistinguishable from the real view.

– If $|C \cap S| < t$, Sim_C is the same as above except replacing $\mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{clt}}(C^{\kappa'}, C^{\kappa'} \cap S^{\kappa'})$ by $\mathsf{Sim}^{\pi^{\mathsf{PSI}}}_{\mathtt{clt}}(C^{\kappa'}, \perp)$ and $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{clt}}(C, (\kappa', |C \cap S|))$ by $\mathsf{Sim}^{\pi^{\mathsf{TST}}}_{\mathtt{clt}}(C, |C \cap S|)$.

In both cases, the client's view can be simulated. Combining the result for the server's view, we conclude that the protocol in Fig. 5 is secure in the semi-honest model.

6.3 Extending to PSI with Access Structure

In Sec. 5.4 we show how to construct STAS by replacing threshold secret sharing in TST with linear secret sharing scheme. Following the same vein, we obtain PSI with *expressive* access structure easily by replacing threshold secret sharing in t-PSI-CA with linear secret sharing scheme. The detailed description is largely the same, thus is omitted for brevity.

7 Evaluation

Comparing our two $\mathrm{OT}^{|I_s \cap I_r|}_{I_r}$ constructions, it is easy to see that both computational cost and communication cost of GBF* are linear in $n = |I_s|$, while OPE-based one is quadratic. For this reason, we only implemented the GBF*-based $\mathrm{OT}^{|I_s \cap I_r|}_{I_r}$ in C, by modifying the GBF source code[3] provided by Dong, and evaluated its performance. The major modification we did is adding OPPRF before GBF related operations. We use existing Paillier encryption implementation[4] for OPPRF. We consider $|I_r| = n$ here.

The experiment is conducted on a virtual machine running Ubuntu 12.04 LTS, allotted 2GB memory and 2 CPUs. Both the client and the server program are run on this virtual machine. The host machine is running Windows 8.1, with 2 Intel(R) Core(TM) i5-4590 3.30GHz CPUs, and 8GB RAM. We only implemented single thread version but we remark that both OPPRF and GBF* are easily parallelizable. We expect the resulting $\mathrm{OT}^{|I_s \cap I_r|}_{I_r}$ retains such property.

The major bottleneck of our GBF*-based $\mathrm{OT}^{|I_s \cap I_r|}_{I_r}$ is Paillier encryption, decryption and computing $\mathsf{Enc}_{pk}(c \cdot m)$ in OPPRF. For a naïve implementation of OPPRF with n elements, the client needs to encrypt n elements in its set X, together with n random number r_c. The server also needs to encrypt n random elements r_s. However, these values do not depend on the other party's input, so both the client and the server can *precompute* these values before starting the whole protocol. Our implementation exploits this observation. We remark that when a more efficient OPPRF is available, the efficiency of our protocol will be improved correspondingly.

We first fix the key length for Paillier encryption to be 1024-bit, and the security of GBF* as 80-bit, because NIST [4] suggests that factorization-based cryptography with 1024-bit key length has 80-bit security. We vary the set size n to be $64, 128, 256, 512, 1024, 2048$, and measure the execution time of GBF*-based $\mathrm{OT}^{|I_s \cap I_r|}_{I_r}$ construction. For higher level of security, we fix Paillier key length to be 3072-bit and the security of GBF* as 128-bit. The result is shown in Table 2 and Figure 6. From the table and the figure we can see that the computation time increases linearly with the set size. When the access structure is simple, *i.e.* the set size is < 20, $\mathrm{OT}^{|I_s \cap I_r|}_{I_r}$ terminates around half a second at 80-bit security. At 128-bit security, the protocol finishes in a few seconds.

8 Further Applications

We discuss a few more applications of our TST.

[3] `https://personal.cis.strath.ac.uk/changyu.dong/PSI/PSI.html`
[4] `http://acsc.cs.utexas.edu/libpaillier/`

Set size	80-bit security (ms)	128-bit security (ms)
64	1769.30	24811.71
128	3385.99	49184.89
256	6582.20	96890.47
512	13080.66	199575.18
1024	25737.88	400866.31
2048	51649.81	778964.15

Table 2: Execution time under various set size and different security level

8.1 Private Match-making

As discussed in the introduction, dating apps which work by connecting two users if their set of attributes overlap, can be supported in a privacy-preserving manner via TST.

A nice observation here is that, the asymmetry in the roles of the client and the server in our protocol may actually be useful in the context of private match-making. For example, a business model of paying user (client) and free user (server) may be employed since the client will know if there is a match and hence has the choice to contact the other party (server) upon receipt of the secret κ (which can be the profile picture).

Moreover, even if the desired threshold is not reached, the users may know to what extent they are similar, which can be a *useful feature* allowing the users to adjust their expected level of similarity for future matching.

It is also possible to store the shares of multiple secrets corresponding to different policies through a *single* invocation of our protocol. For example, a requester who satisfies only the gender criteria can get the ($(1, 1)$ share of) an user's pseudonym with a $(1, 2)$ share of a real first name, when another criteria such as the level of education is satisfied, the corresponding slot will contain only the other $(1, 2)$ share of the first name. This will give great flexibility for dating apps.

8.2 Publish/Subscribe System

As our protocol is for matching in general, it can also find applications in other scenarios where the transfer of material is based on matching of interests. One example is publish/subscribe system. Previous work (such as [46]) focus on using attribute-based encryption for the encryption and possibly also attribute-based mechanism or perhaps some other primitives for the interests matching part. Our solution can act as a handy tool.

8.3 Oblivious Transfer with Access Control

OT with access control (OTAC) is introduced by Camenisch *et al.* [9]. Their construction supports conjunctive policy and is based on a specific construction which covers the credential and the encryption mechanisms. TST enables for the first time OT with threshold access control, in a modular manner.

The initial setup and execution of our approach are similar to those considered in the setting of Camenisch *et al.* [9]. The server encrypts each data item p_i by a homomorphic encryption into e_i under the server's public key. The server publishes all these ciphertexts. The client can use private information retrieval (PIR) technique to get e_i of interest, re-randomizes it by a factor r, and sends it to the server.

Now, the decryption result of this ciphertext is treated as the secret κ of the TST protocol, to be transferred to the client. If the client's attribute satisfies the server's threshold policy, $r \cdot p_i$ will be transferred.

Note that we need to add the corresponding zero-knowledge proofs, in particular, to prove that the re-randomized ciphertext is originated from the server. This can be done by proving that it is a ciphertext signed by the server which is then randomized by a factor of r, without revealing the signature or r. Also, we need to add the proof for showing the credential of the client is certifying the attributes which are used as the selection strings in TST. Remarkably, one can plugin any credential scheme and any encryption scheme supporting the corresponding zero-knowledge proofs efficiently (which are abundant). We think that it is a conceptually simpler and possibly more efficient approach, yet we enjoy more expressive policy.

At the application level, this class of primitive can find applications in pay-per-download music repository, pay-per-retrieval DNA database, etc. For example, a specific solution based on private information retrieval instead of OT with an integrated payment system has been proposed with these kinds of e-commerce applications in mind [24].

9 Conclusion and Future Work

In this paper, we study the problem of secret transfer with access structure, with the aim of revealing as little information as possible. We provide two constructions of STAS, one is based on oblivious polynomial evaluation, and the other one is based on a new variant of garbled Bloom filter. The former one is conceptually simple while the latter one is computationally more efficient. We then show how to use STAS to construct private set-intersection with access structure, which is the first of the kind to the best of our knowledge. Further applications of STAS are also discussed.

All our proposed constructions are proven to be secure in the semi-honest model. It is of theoretical interest to consider stronger security model, like the malicious model and the universal composability model. Also, our t-PSI-CA construction always leaks the size of intersection, leaving an interesting open question to fill up gap.

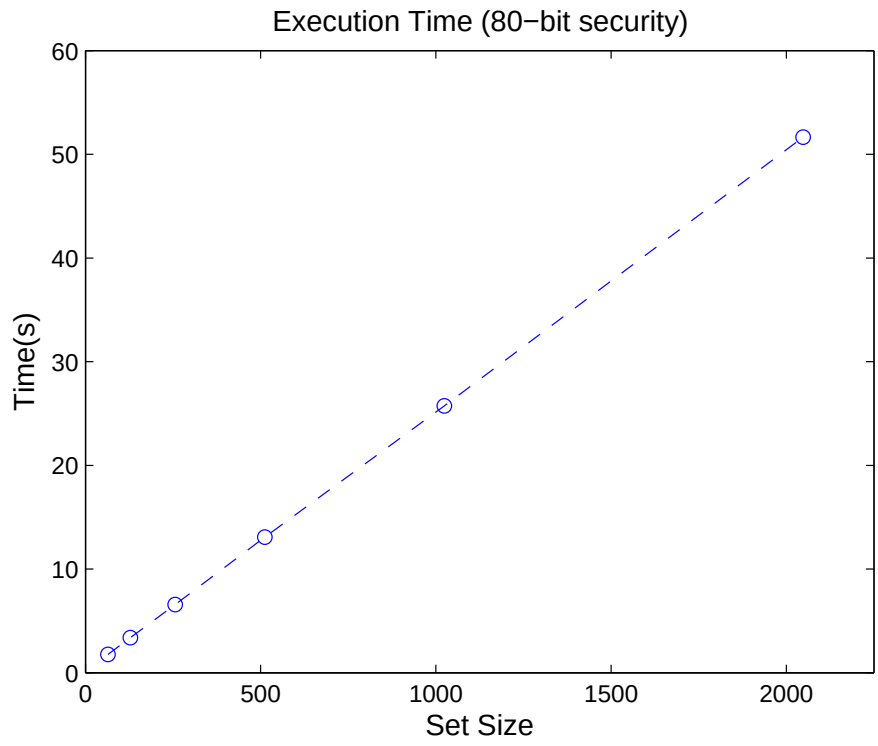
References

1. A. Abadi, S. Terzis, and C. Dong, "O-PSI: delegated private set intersection on outsourced datasets," in *ICT Systems Security and Privacy Protection - 30th IFIP TC 11 International Conference, SEC 2015, Hamburg, Germany, May 26-28, 2015, Proceedings*, 2015, pp. 3–17. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-18467-8_1
2. R. Agrawal, A. V. Evfimievski, and R. Srikant, "Information sharing across private databases," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, 2003, pp. 86–97. [Online]. Available: http://doi.acm.org/10.1145/872757.872771
3. G. Ateniese, E. D. Cristofaro, and G. Tsudik, "(if) size matters: Size-hiding private set intersection," in *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, 2011, pp. 156–173. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19379-8_10
4. E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management," *NIST Special Publication*, vol. 800, no. 57, pp. 1–142, 2007.
5. A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Technion - Israel Institute of Technology, Israel, 1996.
6. B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692
7. A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, 2003, pp. 31–46. [Online]. Available: http://dx.doi.org/10.1007/3-540-36288-6_3
8. P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. H. M. Smid, and Y. Tang, "On the false-positive rate of bloom filters," *Inf. Process. Lett.*, vol. 108, no. 4, pp. 210–213, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.ipl.2008.05.018

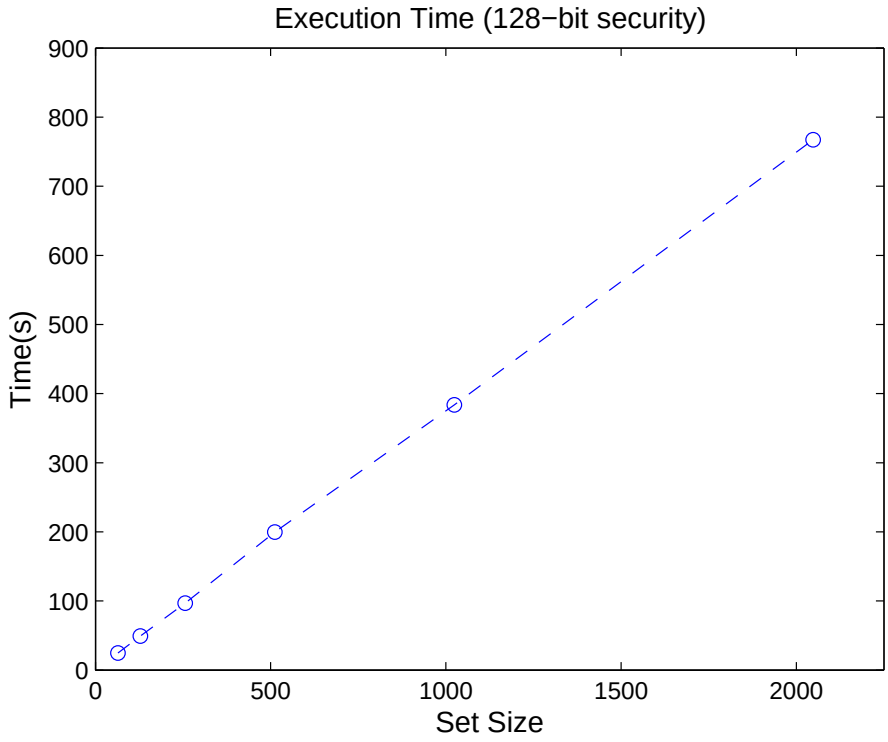
9. J. Camenisch, M. Dubovitskaya, and G. Neven, "Oblivious transfer with access control," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, 2009, pp. 131–140. [Online]. Available: http://doi.acm.org/10.1145/1653662.1653679
10. J. Camenisch and V. Shoup, "Practical verifiable encryption and decryption of discrete logarithms," in *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, 2003, pp. 126–144. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45146-4_8
11. J. Camenisch and G. M. Zaverucha, "Private intersection of certified sets," in *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, 2009, pp. 108–127. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03549-4_7
12. E. D. Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," in *Cryptology and Network Security, 11th International Conference, CANS 2012, Darmstadt, Germany, December 12-14, 2012. Proceedings*, 2012, pp. 218–231. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35404-5_17
13. E. D. Cristofaro, J. Kim, and G. Tsudik, "Linear-complexity private set intersection protocols secure in malicious model," in *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, 2010, pp. 213–231. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17373-8_13
14. E. D. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*, 2010, pp. 143–159. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14577-3_13
15. D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," in *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, 2009, pp. 125–142. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01957-9_8
16. P. D'Arco, M. I. G. Vasco, A. L. P. del Pozo, and C. Soriente, "Size-hiding in private set intersection: Existential results and constructions," in *Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings*, 2012, pp. 378–394. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31410-0_23
17. Y. Dodis and A. Yampolskiy, "A verifiable random function with short proofs and keys," in *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings*, 2005, pp. 416–431. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30580-4_28
18. C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, 2013, pp. 789–800. [Online]. Available: http://doi.acm.org/10.1145/2508859.2516701
19. M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, 2005, pp. 303–324. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30576-7_17
20. M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, 2004, pp. 1–19. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24676-3_1
21. O. Goldreich, *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
22. C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, 2008, pp. 155–175. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78524-8_10
23. C. Hazay and K. Nissim, "Efficient set operations in the presence of malicious adversaries," in *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, 2010, pp. 312–331. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13013-7_19

24. R. Henry, F. G. Olumofin, and I. Goldberg, "Practical PIR for electronic commerce," in *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, 2011, pp. 677–690. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046784
25. S. Hohenberger and S. A. Weis, "Honest-verifier private disjointness testing without random oracles," in *Privacy Enhancing Technologies - PET*, 2006, pp. 277–294. [Online]. Available: http://dx.doi.org/10.1007/11957454_16
26. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, 2003, pp. 145–161. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45146-4_9
27. S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection," in *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, 2009, pp. 577–594. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-00457-5_34
28. A. Juels and M. Sudan, "A fuzzy vault scheme," *Des. Codes Cryptography*, vol. 38, no. 2, pp. 237–257, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10623-005-6343-z
29. F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, 2012, pp. 85–86. [Online]. Available: http://doi.acm.org/10.1145/2414456.2414506
30. L. Kissner and D. X. Song, "Privacy-preserving set operations," in *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, 2005, pp. 241–257. [Online]. Available: http://dx.doi.org/10.1007/11535218_15
31. J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *7th ACM Symposium on Information, Computer and Communications Security, ASIACCS '12, Seoul, Korea, May 2-4, 2012*, 2012, pp. 18–19. [Online]. Available: http://doi.acm.org/10.1145/2414456.2414465
32. A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, 2011, pp. 568–588. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20465-4_31
33. J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, Hong Kong, China, March 22-24, 2011*, 2011, pp. 386–390. [Online]. Available: http://doi.acm.org/10.1145/1966913.1966964
34. J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*, 2009, pp. 347–362. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04474-8_28
35. P. D. MacKenzie, T. Shrimpton, and M. Jakobsson, "Threshold password-authenticated key exchange," *J. Cryptology*, vol. 19, no. 1, pp. 27–66, 2006. [Online]. Available: http://dx.doi.org/10.1007/s00145-005-0232-5
36. T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, 2008, pp. 111–129. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68914-0_7
37. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, 1999, pp. 223–238. [Online]. Available: http://dx.doi.org/10.1007/3-540-48910-X_16
38. B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in *Proceedings of the 24th USENIX Security Symposium*. Washington, D.C.: USENIX Association, Aug. 2015. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas
39. B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, 2014, pp. 797–812. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/pinkas
40. M. D. Raimondo and R. Gennaro, "Provably secure threshold password-authenticated key exchange," *J. Comput. Syst. Sci.*, vol. 72, no. 6, pp. 978–1001, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.jcss.2006.02.002

41. A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, 2005, pp. 457–473. [Online]. Available: http://dx.doi.org/10.1007/11426639_27
42. B. Schneier, *Applied cryptography - protocols, algorithms, and source code in C (2. ed.)*. Wiley, 1996.
43. A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176
44. V. Shoup, "Practical threshold signatures," in *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, 2000, pp. 207–220. [Online]. Available: http://dx.doi.org/10.1007/3-540-45539-6_15
45. 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. [Online]. Available: http://iospress.metapress.com/openurl.asp?genre=article&issn=0926-227X&volume=13&issue=4&spage=593
46. T. H. Yuen, W. Susilo, and Y. Mu, "Towards a cryptographic treatment of publish/subscribe systems," *Journal of Computer Security*, vol. 22, no. 1, pp. 33–67, 2014. [Online]. Available: http://dx.doi.org/10.3233/JCS-130486



(a)



(b)

Fig. 6: Execution time vs. set size: (a) 80-bit security / (b) 128-bit security