Improved Circuit-based PSI via Equality Preserving Compression

Kyoohyung Han, Dukjae Moon, and Yongha Son

Samsung SDS, Korea
{kh89.han, dukjae.moon, yongha.son}@samsung.com

Abstract. Circuit-based private set intersection (circuit-PSI) enables two parties with input set X and Y to compute a function f over the intersection set $X \cap Y$, without revealing any other information. State-of-the-art protocols for circuit-PSI commonly involves a procedure that securely checks whether two input strings are equal and outputs an additive share of the equality result. This procedure is typically performed by generic two party computation protocols, and its cost occupies quite large portion of the total cost of circuit-PSI. In this work, we propose *equality preserving compression* (EPC) protocol that compresses the length of equality check targets while preserving equality using homomorphic encryption (HE) scheme, which is secure against the semi-honest adversary. This can be seamlessly applied to state-of-the-art circuit-PSI protocol frameworks. We demonstrate by implementation that our EPC provides speed-up for circuit-PSI protocols over moderate to high bandwidth (over 100Mbps), which is up to 1.7x around 500Mbps.

Keywords: Private Set Intersection, Circuit-based Private Set Intersection, Homomorphic Encryption

1 Introduction

A two-party functionality of private set intersection (PSI) enables two parties P_0 and P_1 having respective input set X and Y to compute the intersection $X \cap Y$, without revealing any other information beyond the original set cardinality $|X|$ and $|Y|$ to each other.

There are many real-world applications related to PSI, and some of them only requiring the intersection set may find an efficient solution from PSI alone. However, there is another variant of PSI that outputs only $f(X \cap Y)$ for some target function f rather than the intersection set $X \cap Y$, and this would be more desirable for other applications. One typical but a popular example is PSI-Cardinality that computes cardinality of the intersection, where $f(X \cap Y) = |X \cap Y|$. Indeed these kinds of PSI are receiving growing attention from industry, for example, Google [22, 27] and Facebook [6] explored some variants PSI including PSI-Cardinality-with-Sum that computes the cardinality and the sum of associated values over the intersection set.

This PSI-with-computation notion is generalized to the *circuit-PSI* functionality, which outputs the intersection information in secret-shared form, instead of the intersection set itself. More precisely, for each element $x \in X$, circuit-PSI outputs each party random bits s_0 and s_1 respectively, such that $s_0 \oplus s_1 = 1$ if and only if $x \in X \cap Y$ (of course 0 otherwise). This is used as a general-purpose preprocessing, in the sense that two parties use the shares to perform target computation on the intersection. Notable examples would be PSI-Threshold that only reveals whether the cardinality of $X \cap Y$ is larger than some threshold, and private set union (PSU) that literally computes $X \cup Y$.

The work of Pinkas *et al.* [32] proposed a novel construction of circuit-PSI protocol which has linear communication complexity in the input set size. After that, several following works [7, 35] have proposed improved instantiation of the framework and those works indeed shows the state-of-the-art performance for circuit-PSI.

To generate final bits s_0 and s_1 in circuit-PSI, the framework involves $O(N)$ times of private *equality share generation* (ESG) that takes an input string from each party and outputs Boolean shares of the equality result between two strings for $N = |X| = |Y|$. This is one of the main differences of circuit-PSI from plain PSI, where the latter one typically uses private *equality test* that simply outputs the equality result itself. For private equality test, there are many efficient method such as *oblivious pseudo-random functions* (OPRFs) [16, 24, 35]. However it is not directly applicable for ESG, and the most of circuit-PSI protocols perform ESG by other costly methods such as generic two party computation (2PC).

It results in a large performance gap between plain PSI and circuit-PSI. More importantly, the cost for ESG occupies the largest part of circuit-PSI, about 96% and 91% of the total communication in circuit-PSI protocols of [32] and [7] respectively. Recently reported work [35] applied Silent-OT [5] to reduce the communication burden of ESG, but this communication reduction comes at the cost of running time. Then it still takes over 20 times of running time than plain PSI protocols, which means ESG is also the most heavy part of circuit-PSI [35]. Recently, there has been reported remarkable improvements on OT extensions [12, 38] that have small communication cost comparable to Silent-OT [5] and fast computational cost comparable to [23]. These works would improve the performance of ESG procedure, but ESG still takes quite large portion of circuit-PSI protocol.

1.1 Our Contribution

Our work starts with an observation that all known methods for ESG have complexity linear in ℓ, the bit-length of strings. Some works [32, 35] simply exploited two party GMW protocol [19] by evaluating equality check circuit composed of $\ell - 1$ AND gates, and it naturally results in complexity linear in ℓ. After then [7, 15] proposed more efficient protocol that has improved communication burden, but it still suffered from linear complexity in ℓ.

- With a purpose of reducing workload of ESG, we propose a functionality what we call *equality preserving compression* (EPC) that converts two large integers into smaller integers, while preserving the equality condition. Then we construct a homomorphic encryption (HE) based efficient protocol realizing EPC functionality with semi-honest security. Asymptotically it compresses ℓ-bit input integers into $O(\log \ell)$-bits, with $\tilde{O}(\ell)$ computational and communication complexity.

- We then combine our EPC into the circuit-PSI framework of [32], which achieves semi-honest security. Our EPC protocol *perfectly* preserve equality, in other words with zero failure probability, and hence the correctness analysis for previous circuit-PSI protocols remain exactly same. Moreover it provides concrete improvement since it changes the heavy ESG part to be executed logarithmic sized input.

1.2 Related Works

Plain PSI. The early proposal of PSI is based on Diffie-Hellman (DH) [26], and this still serve as a basis of modern PSIs with considerably low communication cost but high computational cost. Recently many OPRF-based (plain) PSI protocols [8, 24, 29, 30, 35] have been reported with rather low computational cost, at the cost of communication burden.

PSI-with-functionality. Toward PSI with additional functionality, Google [22, 27] provides PSI-with-computation protocol stem from DH-based PSI, which is tailored for specific target functionality that reveals computing cardinality of the intersection and summing all associated values of the intersection sets. After then Facebook [6] further developed this to a protocol that letting two parties have additive shares of intersected elements, with a purpose of supporting general computation over the intersection set.

Circuit-PSI. As a more generalized concept, circuit-PSI is firstly proposed by [20] and then continuous improvements have been reported [11, 31, 33]. In particular [31] has a similarity with our paper, as their main idea called permutation-based hashing is to cut-off the length of item while preserving equality, with a purpose of reducing the cost for equality check. However, the technique is only applicable to the initial hashing routine (will be explained by cuckoo/simple hashing later), and not compatible with the currently best framework of circuit-PSI due to Pinkas *et al.* [32] based on *oblivious programmable PRF* (OPPRF). As OPPRF-based circuit-PSI framework shows the best performance, whose details are presented later in Section 3. We note that, despite the similarity of their names, construction of OPPRF is quite different to OPRF, and hence OPRF-based PSI protocol does not implies OPPRF-based circuit-PSI protocol. Indeed, we are aware of only one work [35] that constructs plain PSI and circuit-PSI from the same underlying idea. There is another concept of PSI-with-computation [15] different to circuit-PSI, which improves the efficiency of PSI-with-computation while additionally reveals the cardinality of intersection set as well as the desired function evaluation $f(X \cap Y)$.

HE in PSI field. There are also HE-based PSI approaches [9,10], which mainly focused on extremely unbalance-sized set cases. The first work [10] considered plain PSI, and the main usage of HE is to solve private set membership (PSM) problem by evaluating inclusion polynomial; $x \in Y$ is equivalent to $F(x) = \prod_{y \in Y}(x-y) = 0$, which is quite different to our use of HE. The following work [9] extended this protocol to PSI having associated value and strengthened the security to malicious setting, but HE is applied in similar sense to the previous work. The authors of [9] leaved a short mention on circuit-PSI as a combination of their HE-based PSM protocol with the final equality share generation. As the circuit-PSI protocol was not the main interest of the paper, the authors merely mentioned that the final task can be done by 2PC without detailed analysis.

1.3 Roadmap

In Section 2, we recall the preliminaries including oblivious transfer and homomorphic encryption, and in Section 3, we present the state-of-the-art circuit-PSI framework due to [32]. In Section 4, we propose an equality preserving compression functionality concept and efficient protocol for that. Then in Section 5, we combine our proposed EPC protocol with the OPPRF-based circuit-PSI protocol to improve efficiency, and provide experimental results in Section 6.

2 Preliminary

2.1 Notations

We write vectors as bold lowercase letters, and matrices as bold uppercase letters. For any real number x, we denote $\lfloor x \rceil$ by the round-off to integer. The i-th component of a vector $\mathbf{v}$ is denoted by v_i, and i,j-th entry of a matrix M is denoted by $m_{i,j}$. For an integer k, a set $\{1, \cdots, k\}$ is denoted by $[k]$. The logarithm function log is assumed to have base 2 unless specially denoted by $\log_w$ with base w. For any statement T that can be determined by true or false (Boolean), we denote $\mathbf{1}(T)$ be the truth value for the equality, i.e., it is 1 if T is true and 0 else.

2.2 Oblivious Transfers

A 1-out-of-n oblivious transfer (OT) of ℓ-bit input messages $(n,1)$-OT_ℓ takes as input n messages $m_1, \cdots, m_n \in \{0,1\}^\ell$ from the sender and a choice index $c \in [n]$ from the receiver, and outputs m_c to the receiver and nothing to the sender. We also use a notion of 1-out-of-2 *correlated-OT* (COT) of ℓ-bit input messages $(2,1)$-COT_ℓ, where the sender inputs a correlation $d \in \{0,1\}^\ell$ and the receiver inputs a choice bit $b \in \{0,1\}$. Then the functionality outputs to the sender r and $d+r$ for a randomly chosen $r \in \{0,1\}^\ell$, and to the receiver $b \cdot d + r$. We write m times of $(n,1)$-$\mathsf{(C)OT}_\ell$ calls by $(n,1)$-$\mathsf{(C)OT}_\ell^m$.

There are protocols called OT-extension (OTe) that efficiently extend small numbers of *base* OTs to large numbers of OTs. Assuming that such small numbers of base OTs are done, the most typical IKNP OTe protocols execute $(2,1)$-OT_ℓ and $(2,1)$-COT_ℓ with communication $\lambda + 2\ell$ [23] and $\lambda + \ell$ [3] bits per one call. Recently another breakthrough line of OT extensions [5, 12, 38] are proposed, which greatly reduces communication overhead of IKNP-style OT-extension, while preserving similar communication cost. For sufficiently many OT and COT calls, for example more than 2^{20} calls, Silent OTe allows one to execute $(2,1)$-OT_ℓ and $(2,1)$-COT_ℓ with nearly $2\ell+1$ and $\ell+1$ bit communication per one call, respectively.

Boolean shares and Gate evaluations. For a bit $x \in \{0,1\}$, we say $x_0 \in \{0,1\}$ and $x_1 \in \{0,1\}$ satisfying $x = x_0 \oplus x_1$ be 2-party additive Boolean shares, or simply Boolean shares of x. Consider two bits x and y are shared as x_i and y_i by two party P_0 and P_1. Then two parties can privately compute Boolean shares of gate evaluations on input x and y using OT. Note that Boolean shares for XOR $x \oplus y$ can be easily computed by $x_i \oplus y_i$ by each party's own. Boolean shares for AND gate can be evaluated by $(2,1)$-COT_1^2 [13,19]. For the underlying idea, observe that $(2,1)$-COT_1 with the sender's input correlation bit d and the receiver's input choice bit b essentially computes Boolean shares of $b \wedge d$. To evaluate AND gate, two parties execute a correlated-OT with input x_i and y_{1-i} to have Boolean shares of $a = x_i \wedge y_{1-i}$, and then with input y_i and x_{1-i} to have Boolean shares of $b = x_{1-i} \wedge y_{1-i}$. Then the party P_i outputs $x_i \wedge y_i \oplus a_i \oplus b_i$ and the other party P_{1-i} outputs $x_{1-i} \wedge y_{1-i} \oplus a_{1-i} \oplus b_{1-i}$, which are Boolean shares of $x \wedge y = (x_0 \oplus x_1) \wedge (y_0 \oplus y_1)$.

2.3 RLWE-based Homomorphic Encryption

A homomorphic encryption (HE) scheme is an encryption scheme that supports a ring-structured plaintext $\mathcal{M}$, and homomorphic arithmetic operations between ciphertexts that acts on inner plaintext. We especially exploit a ring learning with errors (RLWE) based HE scheme, BFV scheme [14].

For simplicity, we restrict our description for RLWE-based HE using power-of-2 cyclotomic rings of integers, which is widely used in several HE libraries. Let $\mathcal{R} := \mathbb{Z}[X]/(X^n+1)$ be a polynomial quotient ring where n is a power-of-2 integer. This scheme supports a plaintext space $\mathcal{R}_p := \mathcal{R}/p\mathcal{R} = \mathbb{Z}_p[X]/(X^n+1)$ for some plaintext modulus prime integer p, and the corresponding ciphertext space is $\mathcal{R}_q^2$ for some $q \gg p$.

BFV Scheme. We will briefly review the BFV homomorphic encryption scheme. The IND-CPA security of BFV is based on the hardness assumption of the RLWE problem. For more details, we refer to [4,14].

Key Generation. Given a security parameter $\lambda > 0$, fix integers n, P (P be a positive integer that will be used in the evaluation key generation), and distributions $\mathcal{D}_{key}$, $\mathcal{D}_{err}$ and $\mathcal{D}_{enc}$ over $\mathcal{R}$ in a way that the resulting scheme is secure against any adversary with computational resource of $O(2^\lambda)$. Typically $\mathcal{D}_{key}$ is chosen by ternary coefficient polynomials in $\mathcal{R}$, and $\mathcal{D}_{err}$ and $\mathcal{D}_{enc}$ are chosen by discrete Gaussian distribution of appropriate standard deviation σ.

1. Sample $a \leftarrow \mathcal{R}_q$, $s \leftarrow \mathcal{D}_{key}$, and $e \leftarrow \mathcal{D}_{err}$. Then the secret key is defined as $\mathsf{sk} = (1, s) \in \mathcal{R}^2$, and the corresponding public key is defined as $\mathsf{pk} = (b, a) \in \mathcal{R}_q^2$, where $b = [-a \cdot s + e]_q$.
2. Sample $a' \leftarrow \mathcal{R}_q$ and $e' \leftarrow \mathcal{D}_{err}$. Then the evaluation key is defined as $\mathsf{evk} = (b', a') \in \mathcal{R}_q^2$, where $b' = [-a' \cdot s + e' + Ps']_q$ for $s' = [s^2]_q$.

Encryption. Given a public key pk and a plaintext $m \in \mathcal{R}$, Sample $r \leftarrow \mathcal{D}_{\mathsf{Enc}}$ and $e_0, e_1 \leftarrow \mathcal{D}_{err}$. Then compute $\mathsf{Enc}(\mathsf{pk}, 0) = [r \cdot \mathsf{pk} + (e_0, e_1)]_q$ and $\mathsf{Enc}^{\mathsf{BFV}}(\mathsf{pk}, m) = [\mathsf{Enc}(\mathsf{pk}, 0) + (\Delta_{\mathsf{BFV}} \cdot [m]_p, 0)]_q$, where $\Delta_{\mathsf{BFV}} = \lfloor q/p \rceil$.

Decryption. Given a secret key $\mathsf{sk} \in \mathcal{R}^2$ and a ciphertext $\mathsf{ct} \in \mathcal{R}_q^2$, $\mathsf{Dec}^{\mathsf{BFV}}(\mathsf{sk}, \mathsf{ct}) = \left\lfloor \frac{p}{q} [\langle \mathsf{sk}, \mathsf{ct} \rangle]_q \right\rceil$.

The ciphertext of BFV scheme is $(b(x), a(x))$ satisfying $b(x) = -a(x) \cdot s(x) + e(x)$. The $e(x)$ part is called as *noise term* of ciphertext. We note that infinite norm of noise term of ct in decryption function should be bounded by $\frac{q}{2p}$ for correctness of decryption.

Addition. Given ciphertexts ct_1 and ct_2 in $\mathcal{R}_q^2$, their sum is defined as $\mathsf{ct}_{\mathsf{Add}} = [\mathsf{ct}_1 + \mathsf{ct}_2]_q$.

Multiplication. Given ciphertexts $\mathsf{ct}_1 = (b_1, a_1)$ and $\mathsf{ct}_2 = (b_2, a_2)$ in $\mathcal{R}_q^2$ and an evaluation key evk, their product is defined as $\mathsf{ct}_{\mathsf{Mult}} = \left[(d_0, d_1) + \left\lfloor P^{-1} \cdot d_2 \cdot \mathsf{evk} \right\rceil\right]_q$, where (d_0, d_1, d_2) is defined by $\left[\left\lfloor \frac{p}{q}(b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \right\rceil\right]_q$.

Batching. BFV scheme basically supports encryptions of plaintext ring $\mathcal{R}_p$ element, and homomorphic addition and multiplication over $\mathcal{R}_p$. As a useful notion for batching multiple data in one ciphertext, one can use a ring isomorphism $\mathcal{R}_p \cong \mathbb{F}_{p^d}^{n/d}$ where d is the smallest integer such that $p^d = 1 \mod 2n$ and $\mathbb{F}_{p^d}$ is a finite field of order p^d. Using this isomorphism, one can perform slot-wise encryption and operation of n/d elements in $\mathbb{F}_{p^d}$ by single instruction on the ciphertext. It is worth to note when the plaintext modulus p and the polynomial quotient n satisfies

$$p = 1 \mod 2n, \tag{1}$$

which provides n slots of $\mathbb{Z}_p$ element. This can be achieved only with somewhat restrictive parameters, but the underlying plaintext slot $\mathbb{Z}_p$ is much simpler than extension fields $\mathbb{F}_{p^d}$ so that one can fully enjoy the power of batching. In this regard, we refer this case by *full batch* and indeed our paper mainly focus on full batch HE parameters.

Security Notions. For security, we consider the standard IND-CPA security that requires two ciphertexts of different messages are (computationally) indistinguishable given an encryption oracle. The IND-CPA security of RLWE-based HE literally comes from the hardness of ring learning with errors (RLWE) problem. For concrete parameter setting of IND-CPA security, the bit-size of ciphertext modulus $\log q$ and polynomial ring dimension n, and error distribution $\mathcal{D}_{err}$ should be selected to secure against various lattice reduction attacks.

3 Circuit-based PSI

The definition circuit-based PSI (circuit-PSI) functionality to generate Boolean additive shares is given as Figure 1. After circuit-PSI, the results can be used for one's desired function evaluation. In the rest of this section, we describe the abstract framework of [32] which continues to the following improvements [7,35]. Then we especially review the equality share generation methods of each works which occupies the largest part of the total cost, from which we can observe the input bit-length ℓ equality share generation plays the most crucial role for complexity.

Parameters: A receiver with an input set X of size N and a sender with an input set Y of size N.

Functionality: The functionality sends to the receiver an injective indexing function $\iota : X \to [M]$ for some $M \geq N$ and a vector $\mathbf{s}_0 \in \{0,1\}^M$, and to the sender a vector $\mathbf{s}_1 \in \{0,1\}^M$ such that $s_{0,i} \oplus s_{1,i} = \mathbf{1}(\iota^{-1}(i) \in X \cap Y)$ for $i \in \iota(X)$, and $s_{0,i} \oplus s_{1,i} = 0$ for $i \notin \iota(X)$.

Fig. 1: $\mathcal{F}_{\mathsf{CPSI}}$. (Ideal) Functionality of circuit-PSI

3.1 The OPPRF-based Circuit-PSI Framework

Let the receiver $\mathcal{R}$ holds a set X and the sender $\mathcal{S}$ holds a set Y of the same size N. The framework consists of the following three main stages.

Step 1. Hashing. For $\varepsilon > 0$, each party creates a hash table with $M = (1+\varepsilon)\cdot N$ bins, but with different hashing method. The receiver applies cuckoo hashing with d hash functions $h_1, \cdots, h_d : \{0,1\}^* \to [M]$ on input X. More precisely, for a suitable choice of ε, there is a cuckoo hashing algorithm that stores every element $x \in X$ in $h_j(x)$-th bin for some $j \in [d]$ with overwhelming probability, while ensuring that at most one element is stored in each bin. This yields a simple representation of the cuckoo hash table: $T_X[h_j(x)] = x$. Note that the mapping from $x \in X$ to $h_j(x)$ determines the indexing function ι in the circuit-PSI definition of Figure 1.

On the other hand, the sender creates a simple hash table with the same hash functions on input Y, which stores each $y \in Y$ in every bin $h_j(x)$ for every $j \in [d]$. Naturally each bin can hold more than one element, and hence the i-th bin of the simple hash table $T_Y[i]$ is indeed a set. It is known that that for $M = O(N)$ hash table size, the number of elements in each bin is $O(\log(N))$.

Since $h_j(x) \neq h_j(y)$ for some j implies $x \neq y$, two parties only need to compare each elements of the same bin of each hash tables. Since the cuckoo hash table T_X ensures at most one element of $x \in X$ per each bin, circuit-PSI reduces to the problem that securely outputs an additive share of $\mathbf{1}(T_X[i] \in T_Y[i])$ for each bin i, which is essentially a private set membership (PSM) problem. Here the receiver has to fill the empty bin in T_X with dummy value to prevent additional information leakage.

Step 2. Bin Tagging. This step further reduces the aforementioned PSM problem into an equality share generation (ESG) problem between two parties, where each party inputs a vector $\mathbf{v}$ and $\mathbf{v}^*$ of length M respectively, and is given as output a Boolean vector of additive share of $\mathbf{1}(v_i = v_i^*)$.

This is realized by a functionality called *oblivious programmable pseudo-random function* (OPPRF) [25] where the sender obliviously computes a PRF F on receiver's input while the sender can *program* F with values (y_i, z_i) so that $F(y_i) = z_i$. The formal definition of OPPRF is given as Figure 2. [32] is the first work that applies OPPRF functionality for this purpose, and then [7] and [35] developed more efficient OPPRF protocols to improve the performance of circuit-PSI.

Parameters: A sender with input $L = \{(y_i, z_i)\}$ where $y_i \in \{0,1\}^*$ and $z_i \in \{0,1\}^\ell$, and a receiver with input $X = \{x_i\}$ with $x_i \in \{0,1\}^*$.

Functionality: The functionality samples a random function $F : \{0,1\}^* \to \{0,1\}^\ell$ such that $F(y) = z$ for each $(y, z) \in L$, and sends $F(X) := \{F(x) : x \in X\}$ to the receiver.
After then, upon an input y of the sender, the functionality outputs $F(y)$ to the sender.

Fig. 2: $\mathcal{F}_{\mathsf{OPPRF}}$. (Ideal) Functionality of oblivious programmable PRF

To convert PSM problem to ESG problem, two parties execute a protocol for OPPRF functionality with the following input. The sender who has a simple table samples a random tag value $v_i \in \{0,1\}^\ell$ for each i-th bin, and generate the input set L obtained by concatenating each $y \in Y$ with the tag of the bins where y is stored, namely

$$L = \left\{ \left(y || h_j(y), v_{h_j(y)} \right) \right\}_{y \in Y, j \in [d]} = \left\{ (y' || i, v_i) \right\}_{i \in [M], y' \in T_Y[i]} .$$

The receiver feeds its input set by $\tilde{T}_X = \{T_X[i]||i\}_{i\in[M]}$. After the execution of OPPRF protocol, the receiver assigns

$$v_i^* = F\left(T_X[i]||i\right) \in \{0,1\}^\ell$$

in each hash address i to construct a vector $\mathbf{v}^*$ of length M. From the definition of OPPRF functionality, it holds that $v_i = v_i^*$ if the element $T_X[i]$ is in the set $T_Y[i]$, otherwise v_i^* is a random element. Therefore the original PSM-related problem is translated into equality share generation problem between $\mathbf{v}$ from the sender and $\mathbf{v}^*$ from the receiver.

Remark 1 (Failure Probability). Note that there is a failure probability of $2^{-\ell}$ where the random element v_i^* is same to v_i despite $T_X[i]$ is not in $T_Y[i]$. The length of tag ℓ should be chosen so that the overall failure probability is smaller than $2^{-\sigma}$ where σ is statistical security parameter. Since there are M bins, it should hold that $2^{-\sigma} > 1-(1-2^{\ell})^M$, which is sufficient with

$$\ell > \sigma + \lceil \log M \rceil. \tag{2}$$

One exception is OPPRF of [7] that requires $\ell > \sigma + \lceil \log 4M \rceil$, and this comes from different structure of their OPPRF. For the detailed explanation, see Appendix A.

Step 3. Equality Share Generation. In this step two parties finally generate Boolean shares of $\mathbf{1}(v_i = v_i^*)$, whose definition is formally given as Figure 3.

Parameters: A sender with an input string a and a receiver with an input string b.

Functionality: The functionality outputs bits s_0 and s_1 such that $s_0 \oplus s_1 = \mathbf{1}(a = b)$ to each party respectively.

Fig. 3: $\mathcal{F}_{\mathsf{ESG}}$. (Ideal) Functionality of equality share generation

There are several known methods [7,15] to perform this step in semi-honest model, but thanks to recent improvements on OT extension [12,38], GMW protocol is likely to be the most competitive one. Using GMW protocol, one can evaluate the equality check circuit on ℓ-bit string composed of $\ell-1$ AND gate evaluations using $(2,1)$-$\mathsf{COT}_1^{2\ell-2}$.

3.2 Applications of Circuit-PSI

Below we present some typical but popular applications of circuit-PSI. We would like to remark that the overheads for these applications are significantly small compared to circuit-PSI cost, as also remarked in [32].

Private Intersection Cardinality and Threshold. These applications would be most direct consequences of circuit-PSI. The cardinality of intersection set (*PSI-Ca*) can be obtained by evaluating a Hamming distance circuit that requires less than M AND gates on circuit-PSI outputs. Moreover, by augmenting one comparison circuit to the Hamming distance circuit (less than $M + \log M$ AND gates), we can let the parties know whether the cardinality is larger than some threshold t (*PSI-Th*).

Private Sum over Intersection. Assume the sender having set X additionally holds an associated values $\{v_x \in \mathbb{G} : x \in X\}$ for some additive group $\mathbb{G}$, and we want to let the receiver having set Y knows the sum of associated values over the intersection set, namely $V = \sum_{x \in X \cap Y} v_x$. This is sometimes called *PSI-Sum*[1]. For that we adapt a method of [15]: The sender samples $\mathbf{r} \in \mathbb{G}^M$ that sums to $\sum r_i = 0$. Then two parties execute OT upon the choice bit $s_{1,i}$ from the receiver, and two messages r_i for $s_{0,i}$ choice and $r_i + v_{\iota^{-1}(i)}$ for $1 - s_{0,i}$ choice from the sender, where $v_{\iota^{-1}(i)} = 0$ for $i \notin \iota(X)$. The receiver adds all received value to have $\sum r_i + V = V$, without knowing any other information since each summand is masked by random value r_i. This can be easily tweaked to let the sender know $V = \sum_{x \in X \cap Y} v_x$, by letting the sender samples $\mathbf{r}$ such that $\sum r_i = R$ for a sender-side chosen $R \in \mathbb{G}$. Then from the same protocol the receiver ends with $R + V$, and finally sends back the value to the sender so that the sender recover $V = (R + V) - R$.

Remark 2. Circuit-PSI can handle the case where both parties hold associated value sets so that parties perform further computations over those sets. However it is somewhat complicated as it requires some modification of OPPRF application (of Step 2. Bin Tagging). Thus we simply refer Section 6 of [32] for details.

Private Set Union. Circuit-PSI also leads to private set union (PSU), where the receiver obtains $X \cup Y$. Note that by assuming the receiver holds Y (and the sender holds X), PSU is equivalent to let the receiver know $X \backslash Y$. We can also adapt a method of [15] for PSU as follows: Two parties first run circuit-PSI on each input X and Y so that the result equality share is related to X, i.e., two parties obtain $s_{0,x}$ and $s_{1,x}$ for each $x \in X$. Then the set $X \backslash Y$ can be obtained from OT between two parties, with a choice bit $s_{1,x}$ from the receiver, and two messages x for $s_{0,x}$ choice and $\perp$ for $1 - s_{0,x}$ choice from the sender. Note that the receiver obtains x if $x \in X \backslash Y$, and $\perp$ otherwise, and the sender knows no information about which element of X is sent to the receiver.

[1] Some protocols [15, 22] outputs both cardinality and summation. It should be remarked that circuit-PSI based protocol can selectively exposes cardinality or summation, or even both.

4 Equality Preserving Compression

The final equality share generation procedure occupies the largest part of the total cost in circuit-PSI protocol, and the input bit-length ℓ of equality share generation plays an important role. In this section, we present a procedure that converts the equality share generation target inputs into another values whose size is asymptotically logarithm to the original input bit-length, while the equality results remain unchanged. More formally, we define the 2-party functionality *equality preserving compression* (EPC) $\mathcal{F}_{\mathsf{EPC}}$ that takes an integer $v \in \mathbb{Z}_t$ from the sender and another integer $v^* \in \mathbb{Z}_t$ from the receiver. The functionality outputs each party a random integer r and r^* in another modulus ring $\mathbb{Z}_p$, where it holds that $v = v^*$ in $\mathbb{Z}_t$ *if and only if* $r = r^*$ in $\mathbb{Z}_p$ for $p < t$.

Parameters: A sender with an input $v \in \mathbb{Z}_t$ and a receiver with an input $v^* \in \mathbb{Z}_t$, and the target size p.

Functionality: The functionality sends a random $r \in \mathbb{Z}_p$ and $r^* \in \mathbb{Z}_p$ to the sender and receiver respectively, such that $v = v^*$ in $\mathbb{Z}_t$ if and only if $r = r^*$ in $\mathbb{Z}_p$.

Fig. 4: $\mathcal{F}_{\mathsf{EPC}}$. (Ideal) Functionality of equality preserving compression

4.1 A Basic Protocol

Our protocol starts from the following simple observation on word decomposition. For any base w, the w-base decomposition of v and v^* by $v = \sum_{i=0}^{u-1} v_i \cdot w^i$ and $v^* = \sum_{i=0}^{u-1} v_i^* \cdot w^i$ where $u := \lceil \log_w t \rceil$ and $v_i, v_i^* \in [0, w)$ satisfies

$$v = v^* \iff D := \sum_{i=0}^{u-1} (v_i - v_i^*)^2 = 0 \text{ in } \mathbb{Z}. \tag{3}$$

Note that $D \le u \cdot (w-1)^2 \approx \log_w t \cdot (w-1)^2$, which has much smaller size than the original size t.

Based on this idea, we consider a simple 2-round protocol that privately computes D and output a random element $r \in \mathbb{Z}_p$ and $r^* := r + D \in \mathbb{Z}_p$ by Figure 5. However, the correctness may fails without any condition on the word base w and p, since it may happens that $r = r^* \in \mathbb{Z}_p$ despite of $v \neq v^*$ if D is divisible by p. To avoid this, the word base w has to be chosen so that D is always less than p, namely

$$p > u \cdot (w-1)^2. \tag{4}$$

We note that $u \cdot (w-1)^2 = O(\log t)$, this protocol asymptotically realizes $\mathcal{F}_{\mathsf{EPC}}$ for $p = O(\log t)$.

Parameters: A sender with input $v \in \mathbb{Z}_t$ and a receiver with input $v^* \in \mathbb{Z}_t$ and the target size p.

Protocol:

1. Sender generates a homomorphic encryption secret key sk, and decomposes in w-base $v \in \mathbb{Z}_t$ to $\{v_i\}_{0 \le i < u}$ for $u = \lceil \log_w t \rceil$. After that sender encrypts each v_i using sk, and sends them to receiver.
2. Receiver picks a random integer $r \in \mathbb{Z}_p$, and decomposes $v^* \in \mathbb{Z}_t$ to $\{v_i^*\}_{0 \le i < u}$. Then receiver homomorphically compute $r + \sum_{i=0}^{u-1} (v_i - v_i^*)^2$, and sends the resulting ciphertext back to sender.
3. Sender decrypts the received ciphertext using sk, to obtain $r^* = r + \sum_{i=0}^{u-1} (v_i - v_i^*)^2 \in \mathbb{Z}_p$.

Fig. 5: A basic protocol for $\mathcal{F}_{\mathsf{EPC}}$ functionalities

4.2 Optimizations and Full Protocol

Upon the basic protocol above, we specially focus on BFV scheme to utilize batching property. Furthermore, we achieve huge speed-up from a simple decomposition of $D = \sum (v_i - v_i*)^2$ by totally removing homomorphic ciphertext multiplication. On security aspect, we use noise flooding to ensure function privacy of homomorphic encryption. A full protocol description that puts everything together is presented by Figure 6, and below we provide some details for each technique.

Batching with RLWE-based HE. As reivewed in Section 3, two parties have to perform $O(N)$-many times of equality checks in circuit-PSI. In this regard, we can exploit batch property of BFV scheme to perform multiple calls of $\mathcal{F}_{\mathsf{EPC}}$, on some conditions on target size p and HE parameters. To recall, for the given RLWE dimension n, we can encrypt n/d number of $\mathbb{F}_{p^d}$ elements in one ciphertext for the smallest integer d such that $p^d = 1 \bmod 2n$. This means that using smaller p gives better compression ratio, but this makes the number of slots in a single ciphertext smaller. For example, $p > 2n$ is necessary to use full batch (i.e n slots).

Removing Ciphertext Multiplications. In most of HE schemes, homomorphic multiplication takes much larger time than scalar multiplication. To remove homomorphic multiplications, we let the sender additionally sends one more ciphertext which is an encryption of $\sum_{i=0}^{u-1} v_i^2$. In this case, the receiver can compute D by

$$D = \sum_{i=0}^{u-1} v_i^2 - 2 \cdot \sum_{i=0}^{u-1} v_i \cdot v_i^* + \sum_{i=0}^{u-1} v_i^{*2}.$$

As the receiver knows v_i^* values, it can compute $\sum_{i=0}^{u-1} v_i^{*2}$ part and then the receiver only needs to perform scalar multiplications and additions to obtain an encryption of D.

Remark 3 (Additive HE). This optimization opens possibility to apply *additive homomorphic encryption* (AHE) schemes such as Paillier scheme [28], but the performance of RLWE-based AHE is still better when we use small plaintext space and batching technique. See Appendix B for more detailed argument.

Realizing Function Privacy. For the security proof, we need to ensure the function privacy from the return ciphertext from receiver to sender. For that we apply randomization and noise flooding method, whose detail will be presented in the next subsection. Concretely this can be realized by letting receiver randomize the resulting ciphertext by homomorphically adding a fresh encryption of zero, and add large enough error to apply noise flooding method before send the computation result back to sender.

(In-)efficiency of Binary Case: $w = 2$. The extreme case $w = 2$ deserves to be considered independently, as it obviously results in the smallest output (exactly $\log t$), although requires the largest number of ciphertexts communication. In fact, we found in literature [18] a similar idea using (3) especially for bit decomposition, which further exploits a computational convenience of bit decomposition: Note that $(x - y)^2 = x \oplus y$ for binary x and y, and $x \oplus y$ can be computed by outputting x if $y = 0$ and $1 - x$ otherwise, which does not require even scalar multiplications.

Therefore, one may think $w = 2$ as an appealing choice due to these advantages, while sacrificing some communications. However, we would like to remark that the batching efficiency has to be importantly considered also, and this extreme case indeed has quite poor batching efficiency. It is because the desired plaintext modulus $p \approx \log t$ becomes smaller than a typical choice of the ring dimension $n \geq 4096$ of RLWE-based HE. For example, our interest t is less than 64 bits, and it can be easily checked that small primes of size ≈ 64 have order at least 32 in $\mathbb{Z}_{2n}$ for $n = 4096$. This means that we are only able to batch $128 = 4096/32$ elements in one ciphertexts. On the contrary, we can take full 4096 slots by taking larger word size w that provides $p > 2n$, which are indeed used for our experiments in Section 6.

4.3 Security and Cost Analysis

In this section, we will discuss about security of our protocol with correctness proof. We also analyze the computational and communication costs. Before that, we need to recall some details of RLWE-based HE scheme. We will focus on BFV scheme [14], but it does not mean that our method is restricted to this scheme.

Parameters: A sender with input $\mathbf{v} \in \mathbb{Z}_t^M$ and a receiver with input $\mathbf{v}^* \in \mathbb{Z}_t^M$ and the target size p.

Protocol:

1. [**Setup**] Two parties agree on a proper HE parameter (n, q) that supports plaintext space $\mathbb{Z}_p^n$, and satisfies IND-CPA security. Then the sender samples a key pair $(\mathsf{sk}, \mathsf{pk})$, and sends the public key pk to the receiver. The sender pads $\mathbf{v}$ by 0 and the receiver pads $\mathbf{v}^*$ by 1, until they have length divisible by n, say $\gamma \cdot n$. Two parties also agree on word base w satisfying $p > \lceil \log_w t \rceil \cdot (w-1)^2$, and define $u = \lceil \log_w t \rceil$.
2. [**Encryption**] Sender performs the following for $0 \le k < \gamma$:
   (a) Decompose each v_{nk+j} into $\sum_{i=0}^{u-1} v_{j,i} \cdot w^i$ for $1 \le j \le n$.
   (b) Batch them into $\mathbf{m}_{k,i} = (v_{j,i})_{1 \le j \le n} \in \mathbb{Z}_p^n$ for $0 \le i < u$.
   (c) Define $\mathbf{m}_{k,u} = (\sum_{i=0}^{u-1} v_{j,i}^2)_{1 \le j \le n} \in \mathbb{Z}_p^n$
   (d) Encrypt $\{\mathbf{m}_{k,i}\}$ into $\{\mathsf{ctxt}_{k,i}\}$ using sk and send those ciphertexts to the receiver.
3. [**Compute D and Masking**] Receiver performs the following for $0 \le k < \gamma$:
   (a) Decompose each v_{nk+j}^* into $\sum_{i=0}^{u-1} v_{j,i}^* \cdot w^i$ for $1 \le j \le n$.
   (b) Batch them into $\mathbf{m}_{k,i}^* = (v_{j,i}^*)_{1 \le j \le n} \in \mathbb{Z}_p^n$ for $0 \le i < u$.
   (c) Define $\mathbf{m}_{k,u}^* = (\sum_{i=0}^{u-1} v_{j,i}^{*2})_{1 \le j \le n} \in \mathbb{Z}_p^n$
   (d) Compute a ciphertext $\mathsf{ctxt}_{k,d} = \mathsf{ctxt}_{k,u} \oplus \sum_{i=0}^{u-1} \left(\mathsf{ctxt}_{k,i} \odot 2\mathbf{m}_{k,i}^*\right) \oplus \mathbf{m}_{k,u}^*$
   (e) Sample a random vector $\mathbf{r}_k^* \in \mathbb{Z}_p^n$.
   (f) Generate an encryption $\mathsf{ctxt}_{fp,k}$ (using pk) of zero of error size B_{fp} which is large enough for function privacy.
   (g) Send back $\mathsf{ctxt}_k := \mathsf{ctxt}_{k,d} \oplus \mathsf{ctxt}_{fp,k} \oplus \mathbf{r}_k^*$ to the sender.
4. [**Decryption**] Sender decrypts ctxt_k to have $\mathbf{r}_k \in \mathbb{Z}_p^n$ for $0 \le k < \gamma$.
5. [**Finalize**] Sender outputs $\mathbf{r} \in \mathbb{Z}_p^M$ by concatenating every $\mathbf{r}_k$ and cutting the last $\gamma \cdot n - M$ dummy elements. Receiver outputs $\mathbf{r}^* \in \mathbb{Z}_p^M$ by performing the same with $\mathbf{r}_k^*$.

Fig. 6: A full protocol Π_{BEPC} for M batch calls of $\mathcal{F}_{\mathsf{EPC}}$ functionalities

Randomizing BFV Ciphertexts. Recall that a BFV encryption of a message $m(x)$ is of the form

$$\left(-a(x) \cdot s(x) + \frac{q}{p} \cdot m(x) + e(x), a(x)\right) \in \mathcal{R}_q^2.$$

As secret key owner can recover not only $m(x)$ but also $e(x)$. For this reason, we need to add additional noise $e^*(x)$ such that $|e_i^*| > 2^\sigma \cdot B$ for the function privacy of homomorphic encryption scheme. Here B is upper bound of $e(x)$'s coefficients and σ is the statistical security parameter. This method is called *noise flooding* and this idea is firstly proposed by [17].

Noise Analysis. For the concrete choice of homomorphic encryption parameter, we need to analyze the noise term in our HE-based EPC protocol. Here we will consider the infinity norm $||f(x)||$ which is defined as $\max_i |f_i|$ and the expansion factor of ring $\mathcal{R}$ is defined as $\delta_R = \max\{||f(x) \cdot g(x)||/(||f(x)|| \cdot ||g(x)||) : f(x), g(x) \in \mathcal{R}\}$. In addition, we assume that the noise term of $\mathsf{ctxt}_{k,i}$ in Figure 6 is bounded by B_{fresh}.

Lemma 1 (Noise growth during homomorphic scalar multiplication). *For the given* BFV *ciphertext* $(b(x), a(x))$ *with noise term* $e(x)$ *such that* $||e(x)|| < B$*, the result ciphertext of homomorphic scalar multiplication has noise term* $e^*(x)$ *such that* $||e^*(x)|| < \delta_R \cdot p \cdot B + \delta_R \cdot p^2$.

Proof. In case of homomorphic scalar multiplication, it can be done by multiplying a polynomial $c(x)$ to each $a(x)$ and $b(x)$. Each coefficient of $c(x)$ is bounded by the plaintext modulus p. For the $a^*(x) = c(x) \cdot a(x)$ and $b^*(x) = c(x) \cdot b(x)$,

$$\begin{aligned} b^*(x) + a^*(x) \cdot s(x) &= \left\lfloor \frac{q}{t} \right\rceil \cdot (m(x) \cdot c(x)) + e(x) \cdot c(x) \\ &= \left\lfloor \frac{q}{p} \right\rceil \cdot ([m(x) \cdot c(x)]_p + p \cdot I(x)) + e(x) \cdot c(x) \\ &= \left\lfloor \frac{q}{p} \right\rceil \cdot [m(x) \cdot c(x)]_p + \left(\frac{q}{p} + \epsilon\right) \cdot p \cdot I(x) + e(x) \cdot c(x) \\ &= \left\lfloor \frac{q}{p} \right\rceil \cdot [m(x) \cdot c(x)]_p + \epsilon \cdot p \cdot I(x) + e(x) \cdot c(x) \mod q \end{aligned}$$

Therefore, $||e^*(x)|| = ||\epsilon \cdot p \cdot I(x) + e(x) \cdot c(x)|| \leq \delta_R \cdot p^2 + \delta_R \cdot p \cdot B$. □

Furthermore, homomorphic addition between two ciphertext with noise bound B_1 and B_2 returns ciphertext with noise bound $B_1 + B_2 + 2p$. Finally, homomorphic addition between ciphertext with noise bound B and plaintext returns ciphertext with noise bound $B + 2p$.

From now on, we can analyze the noise term in our HE-based EPC protocol. This analysis gives us concrete HE parameter choices. If we see Figure 6, the receiver have to compute following (at 3-(d)):

$$\mathsf{ctxt}_{k,d} = \mathsf{ctxt}_{k,u} \oplus \sum_{i=0}^{u-1} \left(\mathsf{ctxt}_{k,i} \odot 2\mathbf{m}^*_{k,i}\right) \oplus \mathbf{m}^*_{k,u}.$$

By Lemma 1, the noise term of output ciphertext $\mathsf{ctxt}_{k,d}$ will be bounded by $B^* = 2u \cdot (\delta_R \cdot p \cdot B_{\mathsf{fresh}} + \delta_R \cdot p^2) + B_{\mathsf{fresh}} + 4p$. After that we need to add encryption of zero of error size $B_{fp} = 2^\sigma B^*$ for statistic security parameter σ for the function privacy. At last, receiver needs to add random vector $\mathbf{r}^*_k$ to the ciphertext. So, for the correct BFV decryption at the decryption phase, the ciphertext modulus q should satisfies the following inequality:

$$\frac{q}{p} > (2^\sigma + 1) \cdot \left(2u \cdot (\delta_R \cdot p \cdot B_{\mathsf{fresh}} + \delta_R \cdot p^2) + B_{\mathsf{fresh}} + 4p\right) + 2p.$$

Recall that we have $u = O(\log t)$ for the target size $p = O(\log t)$, and therefore we asymptotically have $q = O(\log^4 t)$ where t is input size.

Theorem 1. *The protocol Π_{BEPC} of Figure 6 realizes M times of $\mathcal{F}_{\mathsf{EPC}}$ functionality calls in a semi-honest model if*

$$q > p \cdot (B_{fp} + B^* + 2p)$$

where $B^* = 2u \cdot (\delta_R \cdot p \cdot B_{\mathsf{fresh}} + \delta_R \cdot p^2) + B_{\mathsf{fresh}} + 4p$ *and* $B_{fp} = 2^\sigma B^*$ *for a statistical security parameter σ.*

Proof. It is already explained that the condition for q provides the correctness and the function privacy required for our protocol.

For the sake of simplicity, we forget batching for a while and assume each parties has integer v and v^* in $\mathbb{Z}_t$. During the protocol execution, the receiver has input v^* and a random output r^*, and its view consists of the public key pk and the ciphertexts of v_i (decomposed value) and $\sum v_i^2$. This can be simulated by replacing all ciphertexts to encryptions of zero, which is indistinguishable from the real execution thanks to the IND-CPA security of HE.

The sender has input v and its view is a ciphertext of $D + r$ and it outputs the plaintext $D + r \in \mathbb{Z}_p$ by decrypting the ciphertext. This can be simulated by encrypting the output $r' \in \mathbb{Z}_p$ of ideal functionality, since from the function privacy the sender cannot know any other information than the decryption result, and the distribution of $D + r$ is identical to the distribution of r' (uniform over $\mathbb{Z}_p$). □

Asymptotic cost analysis. As the ciphertext modulus q is determined as above, we can estimate the total costs. Let $\gamma = \lceil M/n \rceil$ and $u = \log_w t$ by following notations of Π_{BEPC}. For computational cost, our protocol requires $\gamma(u+2)$ encryptions, γu homomorphic scalar multiplications, $2\gamma(u+3)$ homomorphic additions, and γ decryptions for M numbers of EPC calls. Such HE operations including homomorphic scalar multiplication can be done by $O(1)$ numbers of $\mathcal{R}_q$ operations that is roughly translates into $O(n \log n \log q)$ bit operations [4]. By approximating $\gamma n \approx M$, we conclude that amortized computational cost per EPC call is $O(\log t \cdot \log n \cdot \log q)$ bit operations as $u = O(\log t)$. In case of secure RLWE parameters, $n \propto \log q$ roughly holds for the fixed computational security parameter λ. Since $q = O(\log^4 t)$, we conclude that the computational cost per one EPC is $\tilde{O}(\log t)$. Toward communication cost, the sender sends $\gamma(u + 1)$ fresh ciphertexts to the receiver and the receiver returns γ ciphertexts after HE oepration to the sender. The size of fresh ciphertexts is $\gamma(u + 1)(n \log q + \lambda)$ and the size of returned ciphertexts is $2\gamma n \log q$. Then the total communication cost is $\gamma n(u + 3) \log q + \gamma(u + 1)\lambda$ bits. We again approximate $\gamma n \approx M$ and divide the total cost by M to see amortized cost for one EPC call. Then it results in approximately $(u+3) \log q \approx (\log_w t+3) \log q$ bits communication and asymptotically $\tilde{O}(\log t)$ for one EPC call.

5 Application to Circuit-PSI Framework

Our equality preserving compression (EPC) of the previous section can be seamlessly augmented to the OPPRF-based circuit-PSI framework described in Section 3 as Figure 7.

Parameters: A receiver with an input set X of size N and a sender with an input set Y of size N, and compression target bit-length ℓ_{c}.

Protocol:

1. [**Hashing**] Both parties agree on hash functions $h_1, \cdots, h_d$, and table size parameter ε. The receiver construct a cuckoo hash table T_X from X, and the sender constructs a simple hash table T_Y from Y using hash functions $h_1, \cdots, h_d$ into $M = (1+\varepsilon) \cdot N$ bins. The receiver define the address mapping X to T_X by ι.
2. [**Bin Tagging**] The sender samples uniformly random tags $\mathbf{v} \in \mathbb{Z}_{2^\ell}^M$ and sends $L = \{(y' || i, v_i)\}_{i \in [M], y' \in T_Y[i]}$ to $\mathcal{F}_{\mathsf{OPPRF}}$. The receiver sends $\tilde{T}_X = \{T_X[i] || i\}_{i \in [M]}$ to $\mathcal{F}_{\mathsf{OPPRF}}$, and receives $\mathbf{v}^* \in \mathbb{Z}_{2^\ell}^M$ from $\mathcal{F}_{\mathsf{OPPRF}}$.
3. [**Equality Preserving Compression**] The sender sends $\mathbf{v}$ and the receiver sends $\mathbf{v}^*$ to $\mathcal{F}_{\mathsf{EPC}}$, and receives $\mathbf{r} \in \mathbb{Z}_{2^{\ell_{\mathsf{c}}}}^M$ and $\mathbf{r}^* \in \mathbb{Z}_{2^{\ell_{\mathsf{c}}}}^M$ from $\mathcal{F}_{\mathsf{EPC}}$ respectively.
4. [**Equality Share Generation**] For $1 \leq i \leq M$, the sender sends r_i and the receiver sends r_i^* to $\mathcal{F}_{\mathsf{ESG}}$, and receives $\mathbf{s}_{0,i} \in \{0,1\}$ and $\mathbf{s}_{1,i} \in \{0,1\}$ from $\mathcal{F}_{\mathsf{ESG}}$ respectively.

Fig. 7: Π_{CPSI}. Protocol of our circuit-PSI: OPPRF-based framework + EPC

Since EPC perfectly preserve equality (without failure probability), all previous works' analysis for correctness (or failure probability) are still valid. Moreover, as Theorem 1 shows that EPC is secure against semi-honest adversary, the semi-honest security Π_{CPSI} is also guaranteed.

Theorem 2. *The protocol Π_{CPSI} of Figure 7 realizes the $\mathcal{F}_{\mathsf{CPSI}}$ functionality in a semi-honest model in the hybrid model of $\mathcal{F}_{\mathsf{OPPRF}}, \mathcal{F}_{\mathsf{EPC}}$ and $\mathcal{F}_{\mathsf{ESG}}$.*

Effect of EPC. In asymptotic complexity view, the overall cost remains same since EPC itself takes $\tilde{O}(\ell)$ complexities. Thus, we have to figure out concrete costs to see the effect of EPC. We already observe that known methods for ESG has linear cost in ℓ_{c}, and in particular GMW protocol requires about $2M\ell_{\mathsf{c}}$ chosen-message OTs. For EPC from ℓ-bit to ℓ_{c}-bit, the word-size w should be maximally taken so that $2^{\ell_{\mathsf{c}}} \approx \frac{(w-1)^2}{\log w} \cdot \ell$, and it determines the corresponding chunk size $u = \ell / \log w$. Then EPC takes u times homomorphic operations including encryption, scalar multiplication, decryption, and addition with communication of u number of ciphertext. We point that n times of EPC calls can be done at once, thanks to batching property. Thus, as ESG input bit-length reduces from ℓ to ℓ_{c} thanks to EPC, we can save $2n(\ell - \ell_{\mathsf{c}})$ the number of OTs from $\approx u$ times of HE operations. More precisely, we have the trade-off below:

$$\begin{gathered} 2n(\ell-\ell_{\mathsf{c}})\times \text{ chosen-message OTs} \\ \updownarrow \\ u\times \text{ HE scalar-mults } +(u+2)\times \text{ HE adds } + \; u \text{ Ciphertext Comm.} \end{gathered}$$

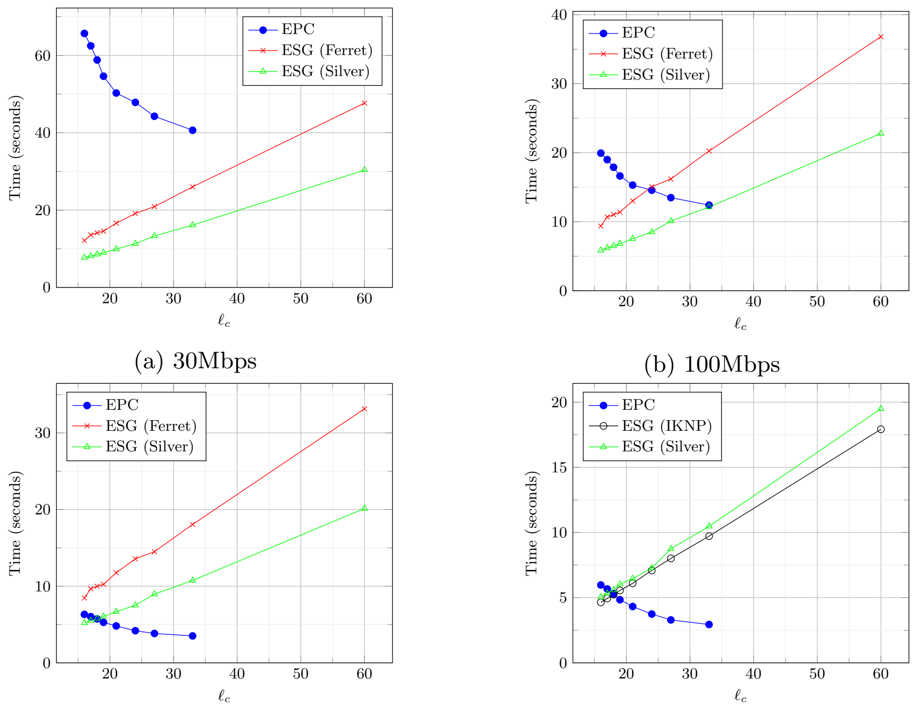
One may think that HE operations are incomparably slow than OT, and hence this trade-off provides no improvement. However, we would like emphasize that our HE operations only consist of scalar multiplication and additions: HE scalar multiplication is just two polynomial multiplication with degree n which is quite fast compare to multiplication between encrypted data. For a concrete example, we may take $n = 4096$, $\ell = 61$, $\ell_{\mathsf{c}} = 19$, and $u = 8$, which is one of exploited parameters in later experiment section. This reduces $344,064$ number of chosen-message OTs at the cost of 8 homomorphic scalar multiplication, 10 homomorphic additions, and 8 HE ciphertext communications.

Round Complexity. On round complexity view, one may think EPC requires additional one communication round than vanilla OPPRF-based framework. However, we remark that ESG stage takes $O(\log \ell)$ rounds for input length ℓ when performed by GMW protocol. Since EPC reduces ESG input length into $\ell_{\mathsf{c}} = O(\log \ell)$, EPC indeed brings asymptotic improvement on the round complexity when GMW protocol is used.

Offline Tag Encryption. The tag vector $\mathbf{v}$ sampled by the sender in the bin tagging step is independent to the input set of the protocol, so it can be sampled before the input set is known, in other words in offline phase. This observation brings negligible improvement in the original framework without equality preserving compression, as it only shifts the random $\mathbf{v}$ sampling time to offline. Meanwhile, it has a notable effect when combined with our equality preserving compression, as the server can perform the encryption phase of Π_{BEPC} in offline phase. Then the online phase of the protocol performs only HE operations, which leads to faster online execution.

6 Performance Evaluation

In this section, we consider concrete instantiation of our circuit-PSI protocol of Section 5 and evaluate the performance. More precisely, we first discuss concrete parameter selections of sub-protocols, especially with respect to the compression target ℓ_{c}. Then we evaluate the performances of several combinations of our EPC protocol and previous ESG protocols. Finally we provide full circuit-PSI protocol costs evaluation by attaching previous hashing and OPPRF steps, and some consequences of our protocols.



(a) 30Mbps (b) 100Mbps (c) 500Mbps (d) 5Gbps

Fig. 8: Timing result of EPC and ESG with various network bandwidth and schemes. All experiments are done with set size $N = 2^{20}$ and $\ell = 61$.

Throughout this section, we assume computational security parameter $\lambda = 128$ and statistical security parameter $\sigma = 40$. For experiments, we use a single machine equipped with 3.50GHz Intel Xeon processors with 128GBs of RAM, where the lower bandwidths is simulated by Wondershaper [21]. All experiments are executed with a single thread on each party in order to be consistent with previous works. For implementation, we use SEAL [36] library for homomorphic encryption, libOTe library [34] for Silver OT [12], and emp-ot library [37] for Ferret OT [38].

6.1 Parameter Selections

OPPRF output length ℓ. For circuit-PSI on input set size N, OPPRF output strings will be fed as input of EPC. We take the length $\ell = \sigma + \lceil \log M \rceil$ for failure probability less than $2^{-\sigma}$ (See Equation 2) where $M = (1 + \varepsilon) \cdot N$ is cuckoo/simple hash table size with d hash functions. We use $\varepsilon = 0.27$ and $d = 3$ by following previous works [32,35], and then OPPRF output length is given by $\ell = \sigma + 1 + \lceil \log N \rceil$.

HE parameters. First of all, we fix HE ring dimension $n = 2^{12}$ which is the minimal one supporting depth-1 scalar multiplication. For the choice of HE plaintext modulus p, one can easily expect that the full batch case is likely to be the most efficient case. Thus we take p to support full batch, whose concrete choice is a prime integer satisfying $p = 1 \mod 2n$. The minimal prime satisfying $p = 1 \mod 2n$ is $p = 40961$, and hence the minimal possible ℓ_{c} is $\lceil \log(40961) \rceil = 16$. For $\ell_{\mathsf{c}} > 16$, there would be several primes p such that $p = 1 \mod 2n$ having bit-length ℓ_{c}, and we choose maximal p among them for each ℓ_{c}. The word-base w is taken by the maximal one satisfying the correctness condition $p > u \cdot (w-1)^2$ where $u = \lceil \ell / \log w \rceil$ is the chunk size. Finally HE ciphertext modulus q is determined as following: an initial modulus is taken q' by the minimal one where our protocol is correct, and then the final modulus q is augmented by σ-bit margin on q' for function privacy. It empirically holds that $\log q \approx \sigma + 2 \log p + \log n$. Detailed HE and EPC parameters are presented in Appendix C.

6.2 The Choice of ℓ_{c} with ESG

Since state-of-the-art OTs [12, 38] have extremely low communication cost, the total communication cost is likely to increase when EPC is applied. Therefore, the choice of ℓ_{c} should be different along with the network environment. In this regard, Figure 8 shows timing results of EPC and ESG protocols for each ℓ_{c} over various bandwidths from 30Mbps to 5Gbps.

For each bandwidth, we choose ℓ_{c} of minimal total running time, whose result is summarized by Table 1. As expected, effect of EPC protocol is negative in low bandwidth case due to the communication cost growth. However, for bandwidth $\geq$ 100Mbps, our protocol shows 1.3 to 2.24 times faster result than ESG only cases. One can see that Silver [12] is always better than Ferret [38], but the both cases deserve to consider since the performance gain of Silver comes from so far non-standard assumption.

Remark 4 (Another ESG from [7]). We found another ESG method from [7], which is not based on GMW protocol. The original paper reported in Table 4 [7] their ESG method takes 9.27 seconds for $N = 2^{20}$ and $\ell = 61$ case, over LAN environment of 3Gbps bandwidth. As EPC takes 3.65 seconds in LAN network with $\ell_{\mathsf{c}} = 24$, we may roughly estimate the combination of EPC and their ESG method to take $7.3 = 3.65 + 9.27 \cdot 24/61$ seconds, and conclude that EPC still brings improvement. This would be the most competitive one for high bandwidth, but as their ESG method requires quite large communications (about 1GB for $N = 2^{20}$), Silver/Ferret based GMW protocol would still be the best ESG method for lower bandwidths. This is not presented in our table, as their implementation is not publicized and we fail to reproduce the reported performance from our re-implementation.

6.3 Performance of Circuit-PSI

We can complete circuit-PSI protocol by attaching hash step and OPPRF at the beginning. In this evaluation, we implement one of state-of-art OPPRF from [7],

ESG Cost	ℓ_c	OTe	With EPC		No EPC	
			Time	Comm.	Time	Comm.
30Mbps	33	Silver	57.78	160.4	**30.36**	37.86
	21	Ferret	66.9	186.1	**47.68**	37.86
100Mbps	21	Silver	22.85	186.1	**22.8**	37.86
	19	Ferret	**28.03**	199.8	36.81	37.86
500Mbps	19	Silver	**11.31**	199.8	20.14	37.86
	16	Ferret	**14.78**	236.5	33.15	37.86
5Gbps	19	IKNP	**10.39**	931.2	17.92	2638.7

Table 1: Performances of ESG with/without EPC, provided with the best choice of ℓ_{c}. All experiments are done with set size $N = 2^{20}$ and $\ell = 61$. Communications in MB, and timings in seconds.

but one may use any other OPPRFs [16, 35], and the choice of OPPRF has no effect on the post ESG and EPC phase. Tables below show the time result and communication cost for $N = 2^{16}$ and 2^{20}. The best case takes 25.32 seconds which was 43.69 seconds without EPC protocol (500Mbps with Ferret OT). Same with the previous sub-section, our method shows improvement when bandwidth larger than 100Mbps.

References

1. Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic Encryption Security Standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.
2. Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty Computation with Low Communication, Computation and Interaction via threshold FHE. In *EUROCRYPT*, pages 483–501. Springer, 2012.
3. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In *ACM CCS*, pages 535–548, 2013.
4. Jean-Claude Bajard, Julien Eynard, M Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*, pages 423–442. Springer, 2016.
5. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient Two-Round OT Extension and Silent Non-Interactive Secure Computation. In *ACM CCS*, pages 291–308, 2019.
6. Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. Private Matching for Compute. *IACR Cryptol. ePrint Arch.*, 2020:599, 2020.

Circuit-PSI $N = 2^{16}$	OTe	With EPC		No EPC	
		Time	Comm.	Time	Comm.
30Mbps	Silver	7.18	20.39	**5.42**	12.53
	Ferret	8.45	22.27	**6.57**	12.53
100Mbps	Silver	3.07	22.27	**2.99**	12.53
	Ferret	4.11	24.54	**3.77**	12.53
500Mbps	Silver	**1.62**	23.03	2.15	12.53
	Ferret	**2.52**	24.54	2.99	12.53
5Gbps	IKNP	**1.47**	66.11	1.89	175.1

Circuit-PSI $N = 2^{20}$	OTe	With EPC		No EPC	
		Time	Comm.	Time	Comm.
30Mbps	Silver	113.65	322.98	**86.23**	200.44
	Ferret	122.77	348.68	**103.55**	200.44
100Mbps	Silver	44.98	348.68	**44.93**	200.44
	Ferret	**50.16**	362.41	58.94	200.44
500Mbps	Silver	**21.85**	362.41	30.68	200.44
	Ferret	**25.32**	399.10	43.69	200.44
5Gbps	IKNP	**18.33**	1093.81	25.86	2801.29

Table 2: Resulting circuit-PSI performances obtained by attaching OPPRF protocol of [7] before ESG. Communications in MB, and timings in seconds.

7. Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-PSI with Linear Complexity via Relaxed Batch OPPRF. Cryptology ePrint Archive, Report 2021/034, 2021. https://eprint.iacr.org/2021/034.
8. Melissa Chase and Peihan Miao. Private Set Intersection in the Internet Setting From Lightweight Oblivious PRF. In *CRYPTO*, pages 34–63. Springer, 2020.
9. Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In *ACM CCS*, pages 1223–1237, 2018.
10. Hao Chen, Kim Laine, and Peter Rindal. Fast Private Set Intersection from Homomorphic Encryption. In *ACM CCS*, pages 1243–1255, 2017.
11. Michele Ciampi and Claudio Orlandi. Combining Private Set-Intersection with Secure Two-party Computation. In *SCN*, pages 464–482. Springer, 2018.
12. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent vole and oblivious transfer from hardness of decoding structured ldpc codes. In *CRYPTO*, pages 502–534. Springer, 2021.
13. Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY-A framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.
14. Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.
15. Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private Set Operations from Oblivious Switching, 2021.

16. Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In *CRYPTO*, pages 395–425. Springer, 2021.
17. Craig Gentry et al. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009.
18. Craig Gentry, Shai Halevi, Charanjit Jutla, and Mariana Raykova. Private Database Access with HE-over-ORAM Architecture. In *International Conference on Applied Cryptography and Network Security*, pages 172–191. Springer, 2015.
19. O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *STOC*, page 218–229, New York, NY, USA, 1987. Association for Computing Machinery.
20. Yan Huang, David Evans, and Jonathan Katz. Private Set Intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.
21. Bert Hubert, Jacco Geul, and Simon Séhier. The Wonder Shaper. https://github.com/magnific0/wondershaper, 2020.
22. Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality. In *EuroS&P*, pages 370–389. IEEE, 2020.
23. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending Oblivious Transfers Efficiently. In *CRYPTO*, pages 145–161. Springer, 2003.
24. Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient Batched Oblivious PRF with Applications to Private Set Intersection. In *ACM CCS*, pages 818–829, 2016.
25. Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical Multi-party Private Set Intersection from Symmetric-key Techniques. In *ACM CCS*, pages 1257–1272, 2017.
26. Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *S&P*, pages 134–134. IEEE, 1986.
27. Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided Malicious Security for Private Intersection-Sum with Cardinality. In *CRYPTO*, pages 3–33. Springer, 2020.
28. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238. Springer, 1999.
29. Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight Private Set Intersection from Sparse OT Extension. In *CRYPTO*, pages 401–431. Springer, 2019.
30. Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: fast, malicious Private Set Intersection. In *EUROCRYPT*, pages 739–767. Springer, 2020.
31. Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private Set Intersection Using Permutation-based Hashing. In *USENIX Security*, pages 515–530, Washington, D.C., August 2015. USENIX Association.
32. Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient Circuit-based PSI with Linear Communication. In *EUROCRYPT*, pages 122–153. Springer, 2019.
33. Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient Circuit-based PSI via Cuckoo Hashing. In *EUROCRYPT*, pages 125–157. Springer, 2018.

34. Peter Rindal. libOTe: an efficient, portable, and easy to use Oblivious Transfer Library. https://github.com/osu-crypto/libOTe.
35. Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and Circuit-PSI from Vector-OLE. In *EUROCRYPT*. Springer, 2021.
36. Microsoft SEAL (release 3.5). https://github.com/microsoft/SEAL, April 2020. Microsoft Research, Redmond, WA.
37. Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. https://github.com/emp-toolkit, 2016.
38. Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated ot with small communication. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1607–1626, 2020.

A Relaxed OPPRF [7]

The OPPRF (Oblivious Programmable PRF) functionality picks a random function F where the sender can program the value $F(x)$ by desired z. In [7], the authors proposed an extended notion called *relaxed OPPRF* functionality, which considers several random functions $F_1, \cdots, F_d$ and the sender programs $x \in X$ so that $F_i(x) = z$ with at least one $i \in [d]$. This converts a bin-wise PSM (Private Set Membership) problem $T_X[i] \in T_Y[i]$ to another PSM problem that checking $z \in \{F_1(x), \cdots, F_d(x)\}$. After then, they apply the standard table OPPRF [25] to further converts this into ESG problem. Rigorously speaking, this consecutive execution of relaxed OPPRF and table OPPRF does not exactly fit to OPPRF functionality definition, since the sender cannot program it's desired values. However, in this work, it only matters that two parties can attach some tags for each bin to convert PSM problem into ESG problem, and we simply say the consecutive execution by CGS OPPRF.

For the failure probability, we note that the authors uses $d = 3$ random functions for relaxed OPPRF that succeeds with probability $(1 - 2^{-\ell})^{3M}$, and then post-OPPRF succeeds with probability $(1 - 2^{-\ell})^{M}$. This results in the condition $\ell > \sigma + \lceil \log 4M \rceil$.

B Comparison with Paillier Additive HE

As our protocol only perform scalar multiplications, one may consider to use another *additive HE* (AHE), for example Paillier [28] scheme. Paillier scheme supports plaintext space $\mathbb{Z}_P$ for some integer P, and the corresponding ciphertext space is $\mathbb{Z}_{P^2}$. Here P is typically taken quite large ($\geq 2^{1024}$) to ensure certain security level, and a naive application of Protocol Π_{EPC} outputs huge random numbers in $\mathbb{Z}_P$. This can be circumvented by applying well-known *smudging* technique [2] where we take a sufficiently large random masking r so that r statistically hides the information of d, and each party take the final modulus reduction by p on each output $d + r$ and r.

However, we argue that RLWE-based AHE is still better for circuit-PSI purpose, where the encryption target message size is much less than 32-bit. RLWE-based AHE can supports plaintext space $\mathbb{Z}_p^N$ for rather small p, and the corresponding ciphertext space is taken $\mathcal{R}_q^2$ where $\log q = O(\log p)$. Then the amortized encryption cost per one message is $2 \log q$. For our interest message size, RLWE ciphertext modulus $q \approx 2^{100}$ suffices so that one message is encrypted into less than 200 bits,. However Paillier AHE encrypts a message into a quite large ciphertext of $2 \log P \geq 2048$ bits, and the amortized cost is less inefficient than RLWE-base AHE.

C HE and EPC Parameters

Below shows detailed parameter information that is used in our experiment. For all cases, ring dimension in HE scheme is fixed with 4096. And, this parameter satisfied 128 security based on homomorphic encryption standard document [1] (except the last row, as maximal possible $\log q$ for 4096 dimension is 109).

N	ℓ_c	p	$\log q$	w	u
2^{16}	16	40961	84	65	10
	17	114689	86	113	9
	18	188417	88	154	8
	20	1032193	92	385	7
	22	4169729	96	834	6
	26	67094289	104	3663	5
	31	2147377153	114	23170	4
2^{20}	16	40961	84	62	11
	17	114689	86	108	10
	18	188417	88	145	9
	19	417793	90	229	8
	21	2056193	94	542	7
	24	16760833	100	1672	6
	27	134176769	106	5181	5
	33	8589852673	118	46341	4

Table 3: HE and EPC parameters in our evaluations.