Overview on Secure Comparison

Quentin Sinh, Jan Ramon

Overview on Secure Comparison

Quentin Sinh and Jan Ramon
MAGNET Team, Univ. Lille, Inria, CNRS, Centrale Lille
UMR 9189 - CRIStAL, F-59000 Lille, France
Email: Quentin.Sinh@inria.fr, Jan.Ramon@inria.fr

***Abstract*—Introduced by Yao's Millionaires' problem [1], Secure Comparison (SC) allows parties to compare two secrets in a privacy-preserving manner. This article gives an overview of the different SC techniques in various settings such as Secret Sharing (SS) or Homomorphic Encryption (HE).**

***Index Terms*—Secure Comparison, Secret Sharing, Homomorphic Encryption, Function Secret Sharing**

I. Introduction

Multi-party computation (MPC) allows a set of n parties $P_1, P_2, \ldots, P_n$ to jointly compute a function $f(x_1, x_2, \ldots, x_n)$ over their respective inputs $x_1, x_2, \ldots, x_n$ without revealing those inputs. While linear operations are easy to realize in a secure multi-party setting, some non-linear operations such as comparison are non trivial. With Privacy-Preserving Machine Learning (PPML) being increasingly studied, computing securely statistics has become a priority. Several statistics requires comparisons, for this reason, we are interested in studying and optimizing Secure Comparison. We describe now some statistics requiring comparisons.

A. Applications

1) The Kendall τ coefficient: In the field of statistics, the Kendall rank correlation coefficient [2] measures the rank correlation of two quantities. Let $\{(x_i, y_i)\}_{i\in[1,n]}$ be a set of observation. A pair of observation (x_i, y_i) and (x_j, y_j) for $1 \leq i, j \leq n$ is said to be *discordant* if the following holds

$$(x_i < x_j \wedge y_i > y_j) \vee (x_i > x_j \wedge y_i < y_j).$$

The Kendall τ uses multiple comparisons as it can be defined as

$$\tau = 1 - \frac{2(\text{number of discordant pairs})}{\binom{n}{2}}. \tag{1}$$

2) Survival analysis: Survival analysis is a branch of statistics that aims at modeling the time remaining before death of biological subjects. The Kaplan-Meier estimator [3] may be used to estimate the survival function. It gives the probability that life is longer than t for a given time t. The Kaplan-Meier estimator $S(t)$ is defined as

$$S(t) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i}) \tag{2}$$

where d_i is the number of death at time t_i, n_i is the number of individuals who survived and t_i is a time that happened before t. Computing the Kaplan-Meier estimator in a privacy-preserving manner is challenging since parties may want to hide the death times t_i, emphasizing the need for secure inequality comparison.

This work has been partially funded by the Horizon Europe TRUMPET project grant no. 101070038.

MPC is an important building block in Federated Learning (FL), where due to the large volumes of data and the increasing availability of dedicated hardware such as GPU the communication cost of SC approaches is the main bottleneck. Although various ways of accomplishing MPC exists (e.g. oblivious transfer, garbled circuits, ...), we mainly focus on the Secret Sharing and Homomorphic Encryption settings. The main contribution of the paper is to compare the different approaches to realize the Less-Than (LT) operation.

The remainder of the article is structured as follows. Section 2 will give some brief preliminaries on notations and background to understand this paper. Section 3 will review the former works done on secure comparison in the SS setting and discuss some new approach using Function Secret Sharing (FSS). Section 4 will summarize some works on secure comparison in the HE setting. Section 5 will provide a comparison between the different approach.

II. Preliminaries

A. Notation

We let λ denote the security parameter throughout this paper. The security parameter measures the probability of an adversary breaking the scheme by brute-force. We let $[x]_M$ be the sharing of a secret x in the ring $\mathbb{Z}_M$ where M can be a prime number or a power of 2. $[x]_B$ will denote the shared vector representing the bit decomposition of value x. We let $\ell = \lceil \log M \rceil$. We denote the indicator function by $\mathbb{I}[\cdot]$, i.e., $\mathbb{I}[true] = 1$ and $\mathbb{I}[false] = 0$. We denote $\oplus$ the bitwise XOR operation. Let $r \xleftarrow{\$} S$ denote r is sampled uniformly from set S.

B. Security Models

1) Semi-honest model: In the semi-honest (passive) model, the participants follow the execution of protocols as expected. However, the goal of the corrupted participants is to gather undisclosed information. We assume that the corrupted parties can collude. While this model seems weak in practice, it ensures that a protocol do not leak information.

2) Malicious model: In the malicious (active) model, the corrupted participants may deviate from the expected execution of the protocol. Thus, this model provides a high level of security.

If n is the number of parties and t is the number of corrupted parties, in the *honest majority*, we have $t < n/2$, whereas we have $t \geq n/2$ in the *dishonest majority*.

C. Linear Secret Sharing (LSS)

Secret Sharing allows a secret $x \in \mathbb{Z}_M$ to be distributed between n parties. We consider here linear secret sharing scheme which allows to perform linear operations on the shares. Let $[a]_M$ and $[b]_M$ be sharings of some secrets $a, b \in \mathbb{Z}_M$. Let $d \in \mathbb{Z}_M$ a public constant value. An LSS scheme should be able to compute $[a + b \mod M]$ and $[da]_M$ without communication. The LSS scheme should also be able to compute multiplication of $[a]_M$ and $[b]_M$ with some communication such that $[ab \mod M] \leftarrow \text{MULT}([a]_M, [b]_M)$. It should also have an operation REVEAL such that $a \leftarrow \text{REVEAL}([a]_M)$.

D. Homomorphic Encryption

Homomorphic Encryption allows computation on encrypted data without decryption beforehand. More formally, let P be the set of plaintexts with the operation $\otimes_M$, C be the set of ciphertexts with $\otimes_C$, $E_k : P \to C$ an encryption function parameterized by a key $k \in K$ with K the set of all the possible keys. An encryption scheme is said to be homomorphic if $\forall m_1, m_2 \in P, E_k(m_1 \otimes m_2) = E_k(m_1) \otimes E_k(m_2)$.

E. Complexity

For both settings, we make the distinction between complexities in the offline phase and the online phase. During the offline phase, some randomized and independent information from the private inputs is computed in order to speed up the calculations during the online phase. We measure the complexity of the offline phase as the number of bits exchanged during this period for the HE setting and the number of multiplications in the SS setting.

We will measure the complexity of most of the protocols in the SS setting using two metrics. The round complexity describes the number of sequential multiplications. The communication complexity gives the overall number of multiplications involved in the protocol. As for the HE setting, we will measure complexity as the size of the ciphertexts exchanged between the parties.

III. Secret Sharing setting

A. Classic approaches

1) Damgard et al. [4]: In this seminal work, two values $a, b \in \mathbb{Z}_M$, where M is a prime number, are secretly shared. The protocol involves a bit decomposition of a and b that are shared between the parties. Once the shares of the bit decomposition are available, the BIT-LT primitive allows parties to compute a share of $[a]_B < [b]_B$.

The authors describe the protocol in the semi-honest model but it can be extended to the malicious model if the multiplication protocol is secure against active adversaries. In terms of complexity, the proposed LT protocol does not have an offline phase. In the online phase, it can be run in a constant number of rounds and requires $\mathscr{O}(\ell \log \ell)$ multiplications.

2) Nishide and Ohta [5]: This article is the first work achieving secure comparison without bit decomposition. The authors noticed that comparison $a < b$ with $a, b \in \mathbb{Z}_M$ can be expressed as an equation of $(a < \frac{p}{2}), (b < \frac{p}{2})$ and $(a - b \mod p < \frac{p}{2})$. If we note respectively w, x, y, z for $(a < \frac{p}{2}), (b < \frac{p}{2})$, $(a - b \mod p < \frac{p}{2})$ and $(a < b)$, one can write the following

$$\begin{aligned} z &= w\bar{x} \vee \bar{w}\bar{x}\bar{y} \vee wx\bar{y} \\ &= w(x + y - 2xy) + 1 - y - x + xy \end{aligned}$$

which can easily be computed in a secret sharing context.

The protocol is described in the semi-honest model. It does not require offline computation and has online communication complexity of $\mathscr{O}(\ell)$ in constant rounds.

3) Reistad [6]: Reistad studied LT on bounded inputs $[a], [b] \in \mathbb{Z}_M$ where M is a prime number and $[a], [b] < \frac{M-1}{2}$. Let $c \equiv 2(a-b) \mod p$ and let c_0 denote the least significant bit of c. Since $M > 2$ is prime, there follows $c_0 = \mathbb{I}[a < b]$.

Then, let $[r]_B$ and $[s]_B$ be a random bitwise shared values, let $[c'] = [c] + [r]_B$ and reveal c'. One can see that $[\mathbb{I}[a < b]] = [c_0] = c'_0 \oplus [r_0] \oplus [\mathbb{I}[[r]_B > c']]$. We hence need to compute $\mathbb{I}[[r]_B > c']$. Let $[x] = \sum_{i=0}^{l-1} [r_i](1 - c_i) 2^{\sum_{j=i+1}^{l-1} c_j \oplus [r_j]}$, then one can verify that $[x_0] = \mathbb{I}[[r]_B > c']$. To compute $[x]$, it requires fan-in multiplications that can be done in $\mathscr{O}(\ell)$ multiplications. To compute $[x_0]$ from $[x]$, let $[d] = [s]_B + [x]$ and $[\hat{d}] = [\hat{s}]_B + [x]$ where $[s]_B = 2^{l-1}[s_{l-1}]_B + 2^{l-2}[s_{l-2}]_B + [\hat{s}]_B$ (ensuring no modulo p is needed to get $[\hat{d}]$). d is revealed. Then, $[x_0]$ can be obtained from $[x_0] = [\hat{s}_0]_B \oplus [\hat{d}_0]$ after computing $[\hat{d}_0]$.

This requires a linear number of online secure multiplications in a constant number of rounds, and in offline a linear number of multiplications is needed to generate $[r]_B$ and $[s]_B$. The protocol proposed by Reistad is described in the semi-honest model.

4) Catrina and De Hoogh [7]: The authors split the working space $\mathbb{Z}_M$ into two spaces categorized as positive and negative integers. If we take $M = 2^k$, then we can consider the set $\{0, \ldots, 2^{k-1} - 1\}$ to be the positive integers and $\{2^{k-1}, \ldots, 2^k - 1\}$ to be the negative values. Under this assumption, the authors construct a truncation for shared values to take the most significant bit of $c \equiv a - b \mod M$ which acts as a sign function for values in the ring $\mathbb{Z}_M$. The truncation is done by computing $([c] - [c'])(2^{-(k-1)} \mod M)$ where $[c'] \equiv [c] \mod 2^{k-1}$. To calculate $[c']$, one first generate masks $[r'']$, $[r']$, $[r']_B$ and reveal $d = 2^{k-1} + [c] + 2^{k-1}[r''] + [r']$. Let $d' \equiv d \mod 2^{k-1}$. One can verify that $[c'] = d' - [r'] + 2^{k-1}[u]$ where $[u] = \mathbb{I}[d' < [r']_B]$.

The protocol works in the semi-honest model. It does not require an offline phase. During the online phase, it requires $\mathscr{O}(\ell)$ multiplications in $\mathscr{O}(\log \ell)$ rounds.

5) Lipmaa and Toft [8]: The proposed LT protocol will be built on an equality test. The authors first introduce two equal-

ity tests based on the hamming distance and on a Disclose-If-Equal protocol. The intuition behind the LT protocol is that it can be performed by doing $\log \ell$ equality tests. In a recursive manner, we can compare the $\ell/2$ most significant bits first. If they differ, the protocol divides those bitstrings into two bitstrings each, etc. More formally, let $[z] = 2^\ell + [a] - [b]$ and $[m] = [z] + [R^\ell]$ with $[R^\ell]$ a shared mask. Parties can reveal m and calculate $m_\top \equiv \lfloor m/2^{\ell/2} \rfloor \mod 2^{\ell/2}$ and $m_\perp \equiv m \mod 2^{\ell/2}$. Similarly, one can define $[R_\top]$ and $[R_\perp]$. Let $[b] = \mathbb{I}[m_\top = [R_\top]]$. If $[b] = 1$, then recursively, one can LT on $m_\perp$ with $[R_\perp]$. At the end, parties get $[z \mod 2^\ell]$ and can compute $([z] - [z])2^{-\ell} \mod 2^\ell$, similar to [7].

The given protocol for LT works in the malicious setting. Complexity-wise, the scheme needs $\mathcal{O}(\ell)$ multiplications for the offline phase for generating masks. During the online phase, LT can be done in $\mathcal{O}(\log \ell)$ multiplications in $\mathcal{O}(\log \ell)$ rounds as searching for the different bits is done as a dichotomic search.

While works presented above were addressing the issue of security in the honest majority, we now describe protocols that are secure in the dishonest majority (malicious model).

6) Escudero, Ghosh, Keller, Rachuri, Scholl [9]: This more recent work (2020) allows a secret shared arithmetic value $[x]_M$ to be easily transformed into binary shared data $[x]_B$. edaBit stands for *extended doubled authenticated bits*. An edaBit is a shared arithmetic value $[x]_M$ along with the bit decomposition of x shared which will be denoted $([x]_M, [x]_B)$. Hence, the value x is not known to any of the parties and the bit decomposition as well. edaBits can be generated in the dishonest majority malicious setting in two phases: the first phase allows parties to create *private* edaBits which will be combined into a *global* edaBit in the second phase.

The article also proposes a scheme to do the LT operation on edaBits which is similar to [7] by using a truncation and requires $\mathcal{O}(\ell)$ multiplications in $\mathcal{O}(\log \ell)$ rounds.

7) Makri, Rotaru, Vercauteren, Wagh [10]: The authors use edaBits to build a more efficient LT protocol in the dishonest majority setting. It uses two edaBits r, r' to mask the two shared values $[a]_M, [b]_M \in \mathbb{Z}_M$ we want to compare. The scheme is built on the primitive Less-Than-Bits (LTB) that compares a public known value with a shared bit decomposition. First, parties reveal $[x]_M = [a+r]_M$ and $[y]_M = [r'-b]_M$ and compute $T \equiv a + b \mod M \equiv b - a + r + r' \mod M$. They get shares $[w_1] = [\mathbb{I}[x < [r]_B]]$ and $[w_2] = [\mathbb{I}[y, [r']_B]]$ using LTB. Let $w_3 = \mathbb{I}[T < b]$. Then, parties compute bitwise addition of $[s]_M = [r]_M + [r']_M$. This operation requires $\mathcal{O}(\ell \log \ell)$ multiplications. We extract the last carry bit $[s_\ell]$ and compute $[w_5] = [\mathbb{I}[T < [s]_B]]$. Finally, $[\mathbb{I}[a < b]] = [w_1] + [w_2] + w_3 - [w_4] - [w_5]$. The intuition behind the protocol is that we can express T as a sum in two different ways.

Overall, the scheme requires two edaBits that can be computed during the offline phase, thus requiring $\mathcal{O}(\ell \log \ell)$ multiplications. In the online phase, the protocol requires $\mathcal{O}(\ell \log \ell)$ multiplications in $\mathcal{O}(\log \ell)$ rounds.

B. Novel approach using Function Secret Sharing

1) Function Secret Sharing: Introduced by [11], it allows parties to split a function f into additive shares $f_1, f_2, \ldots, f_n$ such that $f = f_1 + f_2 + \cdots + f_n$. Here, we give the primitive for a two party setting. FSS consists of two algorithms $(\text{Gen}, \text{Eval})$:

- Gen(1^λ, f^*): outputs a pair of keys (k_0, k_1). f^* is the description of a function f.
- Eval(b, k_b, x): outputs $f_b(x)$.

Correctness: if $(k_0, k_1) \leftarrow \text{Gen}(1^\lambda, f^*)$ then Pr[Eval(0, k_0, x) + Eval(1, k_1, x) = $f(x)$] = 1.

Security: *There exists a PPT Simulator $\mathcal{S}$ such that, for a function f^* and a bit $b \in \{0,1\}$, $\{\mathcal{S}(1^\lambda, b)\}_{\lambda \in \mathbb{N}}$ and $\{k_b | (k_0, k_1) \leftarrow \mathit{Gen}(1^\lambda, f^*)\}_{\lambda \in \mathbb{N}}$ are indistinguishable.*

2) Distributed Interval Function (DIF): An interval function $f_{(a,b),\beta}$, for $a, b \in \mathbb{G}^{\text{in}}$ and $b \in \mathbb{G}^{\text{out}}$, is a function such that $f(x) = \beta$ if $a \leq x \leq b$ and $f(x) = 0$ otherwise.

3) Boyle et al. [12]: The general protocol describes a way for two parties P_0 and P_1 to compute any circuit composed of functions g with optimal communication complexity if g can be expressed efficiently by a FSS scheme such as DIFs.

Suppose that function g can be expressed by a FSS scheme. g can be wrapped into another function $\hat{g}_{r^{\text{in}}}(x) = g(x - r^{\text{in}})$ with $x, r^{\text{in}} \in \mathbb{G}^{\text{in}}$. During the offline phase, each party will receive a key $k_{\hat{g}}^\sigma$ with $\sigma \in \{0,1\}$ of the function $\hat{g}_{r^{\text{in}}}$ and additive shares of $r^{\text{out}} = \langle r_0^{\text{out}}, r_1^{\text{out}} \rangle$ and $r^{\text{in}} = \langle r_0^{\text{in}}, r_1^{\text{in}} \rangle$. When evaluating the function g on x, parties reveal the masked value $x + r^{\text{in}}$. Party P_σ evaluates its key k_g^σ on $x + r^{\text{in}}$ with the Eval algorithm to get a share $[\hat{g}_{r^{\text{in}}}(x + r^{\text{in}})] = [g(x)]$. Party P_σ adds to it its additive share r_σ^{out} to get $[g(x)] + r_\sigma^{\text{out}}$. They then reveal $g(x) + r^{\text{out}}$ so that they can continue to evaluate the circuit without revealing the values. If function g is the last function to be evaluated, r^{out} can be disregarded.

4) Case of LT operation: Suppose P_0, P_1 has respective inputs $x, y \in \mathbb{G}^{\text{in}}$. It is required that the inputs $x, y \in \mathbb{G}^{\text{in}}$ should respect the following inequality $|x - y| < |\mathbb{G}^{\text{in}}$. This allows us to split the inequality into two simple cases:

- if $y - x$ wrap-around, then $x \geq y$,
- if $y - x$ does not wrap-around, then $x < y$.

Now, we need to take into account the offset created by $r^{\text{in}} = \langle r_0^{\text{in}}, r_1^{\text{in}} \rangle$, hence, this can be implemented using 2 instances of DIFs. Let $p = \lfloor |\mathbb{G}^{\text{in}}|/2 \rfloor$.

- If $-p \leq r_1^{\text{in}} - r_0^{\text{in}} \leq 0$, then, $\hat{f}_{r^{\text{in}}} = f_{\text{DIF}[(0, -(r_1^{\text{in}} - r_0^{\text{in}})), 1]} + f_{\text{DIF}[(0, p + r_1^{\text{in}} - r_0^{\text{in}}), -1]}$.
- If $0 < r_1^{\text{in}} - r_0^{\text{in}} \leq p$, then, $\hat{f}_{r^{\text{in}}} = f_{\text{DIF}[(0, r_1^{\text{in}} - r_0^{\text{in}}), 1]} + f_{\text{DIF}[(p + r_1^{\text{in}} - r_0^{\text{in}}, |\mathbb{G}^{\text{in}}|), 1]}$.

5) Benefits: Boyle's approach allows parties to compute any function that can be efficiently represented by an FSS scheme on secretly shared values. Suppose working with input $x \in \mathbb{G}^{\text{in}}$ secretly shared between parties P_0, P_1 where $x = x_0 + x_1$. If parties want to a function g with masks r^{in} and r^{out}, the trusted dealer can sample random additive shares $\langle r_0^{\text{in}}, r_1^{\text{in}} \rangle$ of r^{in} such that each party P_σ can mask their share of x_σ with r_σ. After evaluation, each party has a share $[g(x)]$.

This protocol has optimal online communication complexity as parties are only required to exchange their inputs for an evaluation. Thus, online communication complexity requires no multiplication and can be done in constant rounds. We will measure the online complexity for this scheme in bits exchanged which equals $\mathcal{O}(\ell)$ bits.

6) Limits: This optimal online phase comes with costs in the offline phase. Indeed, FSS keys are large and requires to exchange $\approx 4\lambda\ell$ bits in the offline phase for LT. The scheme requires to generate a pair of distinct FSS keys for each function as changing the mask is needed.

IV. Homomorphic Encryption setting

1) Damgard, Geisler, Kroigard [13]: The scheme uses a combination of additive secret sharing and homomorphic encryption. The described setting is the following: we want to compare a private value m that is bitwised shared between two parties such that $m_i = m_0^i + m_1^i \mod u$ for $1 \leq i \leq \ell$ and a public known value x. We have $sk = (p, q, v)$ and $pk = (n, u, g, h)$ such that $n = pq$ with p, q prime numbers, $u|(p-1), v|(q-1)$, $g, h \in \mathbb{Z}_n^*$ such that the order of h is $v \mod p$, q and g has order uv. The keys are generated by P_0. The plaintext space is $\mathbb{Z}_u$ while the ciphertext space is $\mathbb{Z}_n^*$.

- $\text{Enc}(pk, m) = g^m h^r \mod n$ for $r \leftarrow \mathbb{Z}_n^*$.
- $\text{Dec}(sk, c)$: it is not a real decryption as it decides if c encrypts 0 or not. If $c^v \mod n = 1$ then c encrypts 0.

Party P_σ calculates $c_\sigma^i = x_i - m_\sigma^i + 1 + \sum_{j=i+1}^{\ell} w_\sigma^i$ such that $w_\sigma^i = m_\sigma^i \oplus x_i$. We have $c^i = c_0^i + c_1^i$. If $m > x$, then there is exactly one position i where $c^i = 0$, otherwise no such position exists. Then, parties compute $\text{Enc}(pk, c_\sigma^i)$ and send it to P_1. Party P_1 can use the homomorphic property to add the shares of c^i and sends to P_0 the encryption for each bit. P_0 can then check if any bits is equal to 0 using the Dec algorithm. If so, $m < x$, otherwise $m \leq x$.

Security can be achieved for malicious adversaries. We require in the offline phase to exchange the public key, which is roughly about $\mathcal{O}(\lambda)$ bits long. If $k = \lceil \log u \rceil$ and $l = \lceil \log n \rceil$, then during the online phase, k encrypted bits are exchanged. Thus, the online communication complexity is $\mathcal{O}(k \cdot l)$ bits in constant rounds.

2) Carlton, Essex, Kapulkin [14]: The proposed scheme is based on the Small RSA Subgroup Decision Problem: given (n, b, d, g, u) and x a quadratic residue modulo $n = pq$ where $p = 2b^d p_s p_t + 1$ and $q = 2b^d q_s q_t + 1$ with q_s, p_s primes of bit-length u, q_t, p_t primes whose bit-length is not u, the goal is to determine if x has order $p_s q_s$. P_0, P_1 have respectively private inputs m_0, m_1 and the result of comparison is public. We require P_0 to generate the public key $pk = (n, b, d, g, h, u)$ where g, h are generators of some subgroups of $\mathbb{Z}_n^*$ and $sk = x = p_s q_s x'$ where $x' = (p_s q_s)^{-1} \mod b^d$.

- $\text{Enc}(pk, m \in \mathbb{Z}_d)$: pick a $r \xleftarrow{\$} \{1, \ldots, 2^u - 1\}$ and compute $C = g^{b^m} h^r \mod n$ with $C \in \mathbb{Z}_n^*$.
- $\text{Dec}(sk, C)$: compute $C^x \mod n = (g^{b^m} h^r)^{p_s q_s x'} = g^{b^m p_s q_s x'} = g^{b^m}$. To recover m, $b^m = \log_g(g^{b^m} \mod n)$ then $m = \log_b(b^m)$. It is easily computable since the order of g is a power of d of a small prime base b where b is chosen to be small.

The LT scheme works as follows:

1) Party P_0 encrypts m_0 such that $C \leftarrow \text{Enc}(pk, m_0)$ and sends C to P_1.
2) P_1 computes $D \leftarrow C^{b^{d-m_1}} g^s h^{r_1}$ where $r_1 \leftarrow \{1, \ldots, 2^u - 1\}$ and $s \leftarrow \{1, \ldots, b^d - 1\}$ and sends D to P_0.
3) P_0 calculates $g^w \leftarrow D^x$ and $w \leftarrow \log_g(g^w)$.
4) Both party engage in a Plaintext Equality Test (PET) protocol to determine if $w = s$.

D can be written as $\text{Enc}(pk, m_0)^{b^{d-m_1}} g^s h^{r_1} = g^{b^{d+m_0-m_1}+s} h^b$ for some b. If $m_0 - m_1 \geq 0$, then, when decrypting, $b^{d+m_0-m_1} + s \mod b^d = b^{d+a} + s$ for some a. Thus, $b^{d+a} + s = 0 \cdot b^a + s = s$. If $m_0 + m_1 < 0$ then $b^{d+m_0-m_1} + s \mod b^d = b^{m_0-m_1} + s \mod b^d$.

Let $l = \lceil \log n \rceil$. Complexity-wise, 2 ciphertexts of size $\mathcal{O}(l)$ are being exchanged and 2 PETs are used (4 Elgamal ciphertexts exchanged $\approx \mathcal{O}(l)$) in $\mathcal{O}(1)$ rounds.

V. Comparison

We remind that the online and offline complexity is measured in the number of multiplications (m) for [4]–[10] and in the bits exchanged (b) for [12]–[14]. k is the length of a plaintext and l the length of a ciphertext while ℓ denotes the length of the sharing space. 'A' means active model and 'P' means passive model. 'HM' stands for Honest Majority while 'DM' is for Dishonest Majority. λ is the security parameter which is different depending on the schemes. [12] requires $\lambda = 128$ as it uses a symmetric primitive whereas [13], [14] requires $\lambda = 2048$ as the schemes are based on RSA modulus.

The HE setting generally addresses the problem in the 2-party setting instead of n-party setting. The SS setting has the benefit of giving an immediate scalable solution.

TABLE I
Complexity comparison of the different schemes

Protocol	Setting	Offline	Online	Rounds	Model	Setting
[4]	SS (m)	-	$\mathcal{O}(\ell \log \ell)$	$\mathcal{O}(1)$	A	HM
[5]	SS (m)	-	$\mathcal{O}(\ell)$	$\mathcal{O}(1)$	P	HM
[6]	SS (m)	$\mathcal{O}(\ell)$	$\mathcal{O}(\ell)$	$\mathcal{O}(1)$	P	HM
[7]	SS (m)	-	$\mathcal{O}(\ell)$	$\mathcal{O}(\log \ell)$	P	HM
[8]	SS (m)	$\mathcal{O}(\ell)$	$\mathcal{O}(\log \ell)$	$\mathcal{O}(\log \ell)$	A	HM
[9]	SS (m)	$\mathcal{O}(\ell \log \ell)$	$\mathcal{O}(\ell)$	$\mathcal{O}(\log \ell)$	A	HM
[10]	SS (m)	$\mathcal{O}(\ell \log \ell)$	$\mathcal{O}(\ell \log \ell)$	$\mathcal{O}(\log \ell)$	A	DM
[12]	SS (b)	$\mathcal{O}(\lambda \ell)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	P	DM
[13]	SS / HE (b)	$\mathcal{O}(\lambda)$	$\mathcal{O}(k \cdot l)$	$\mathcal{O}(1)$	A	HM
[14]	HE (b)	$\mathcal{O}(\lambda)$	$\mathcal{O}(l \cdot \lambda)$	$\mathcal{O}(1)$	P	HM

VI. Conclusion

We have compared several different ways to conduct SC, both in the HE and SS settings. One of the most promising way is to use FSS as this leads to optimal communication complexity for the online phase. However, computing the overhead might be costly. Therefore, one line of work could be to reduce this overhead by creating a reusable gate for the same function.

REFERENCES

[1] Yao, A. C. "Protocols for secure computations." 23rd annual symposium on foundations of computer science. IEEE, 1982.

[2] Kendall M., "Rank correlation methods." Griffin, 1948.

[3] Kaplan E. L., Meier P. "Nonparametric estimation from incomplete observations." JASA, 53(282), 457-481, 1958.

[4] Damgård I., Fitzi M., Kiltz E., Nielsen J. B., Toft T. "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation." In TCC (pp. 285-304), 2006.

[5] Nishide T., Ohta K. "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol." In PKC 2007, April 16-20, 2007. Proceedings 10 (pp. 343-360), 2007.

[6] Reistad T. I. "Multiparty comparison-an improved multiparty protocol for comparison of secret-shared values." In SECRYPT, 2009.

[7] Catrina O., De Hoogh S. "Improved primitives for secure multiparty integer computation." In SCN 2010, (pp. 182-199), 2010.

[8] Lipmaa H., Toft T. "Secure equality and greater-than tests with sublinear online complexity." In ICALP 2013, July 8-12, (pp. 645-656), 2013.

[9] Escudero D., Ghosh S., Keller M., Rachuri R., Scholl P. "Improved primitives for MPC over mixed arithmetic-binary circuits." In CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, (pp. 823-852), 2020.

[10] Makri E., Rotaru D., Vercauteren F., Wagh S. "Rabbit: Efficient comparison for secure multi-party computation." In FC'2021, 2021.

[11] Boyle E., Gilboa N., Ishai Y. "Function secret sharing." In Eurocrypt 2015 (pp. 337-367), 2015.

[12] Boyle E., Gilboa N., Ishai Y. "Secure computation with preprocessing via function secret sharing." In TCC 2019, (pp. 341-371), 2019.

[13] Damgard I., Geisler M., Kroigard M. "Homomorphic encryption and secure comparison." IJACT, 1(1), 22-31, 2008.

[14] Carlton R., Essex A., Kapulkin K. "Threshold properties of prime power subgroups with application to secure integer comparisons." In RSA Conference (pp. 137-156), 2018.