IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 8, 2019, accepted March 17, 2019, date of publication March 29, 2019, date of current version April 13, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2908230*

# Optimally Efficient Secure Scalar Product With Applications in Cloud Computing

**BABAK SIABI[1], MEHDI BERENJKOUB[1], AND WILLY SUSILO[2]**
[1]Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran
[2]School of Computing and Information Technology, Institute of Cybersecurity and Cryptology, University of Wollongong, Wollongong, NSW, Australia

Corresponding author: Willy Susilo (wsusilo@uow.edu.au)

**ABSTRACT** Secure computation of scalar product is of considerable importance due to its central role in many practical computation scenarios with privacy and security requirements. This paper includes new results about the secure two-party scalar product. Specifically, a perfectly secure and universally composable two-party split scalar product (SSP) protocol is proposed in the preprocessing model. In addition to full security, the proposed SSP protocol enjoys the advantage of optimal efficiency. To show the optimality of this SSP protocol, information theoretic lower bounds on the amount of communicated data in a secure two-party computation in the preprocessing model are derived. These bounds are not limited to SSP functionality but apply to a large class of two-party functionalities. A part of this paper is devoted to applications of the proposed SSP protocol in secure cloud computing. Specifically, based on this protocol, a cloud-assisted privacy-preserving profile-matching scheme and a secure remote health monitoring scheme are proposed. Both of the solutions are highly efficient and significantly improve the previous work.

**INDEX TERMS** Secure computations, preprocessing model, optimal efficiency, privacy-preserving scalar product, privacy-preserving profile-matching, secure remote health monitoring.

## I. INTRODUCTION

Secure computations (SC) is an important research line in modern cryptography that constitutes powerful solutions to handle sensitive distributed computations. In such computation scenarios a number of mutually untrustworthy parties need to compute a function on their private inputs to acquire their possibly private outputs. Traditionally, such computations have been carried out either with the help of a trusted third party or by sacrificing a part of sensitive security requirements.

The first study of secure computations was carried out by Andrew C. Yao and resulted in a secure protocol for computing general two-party functionalities [1]. Subsequently, more fundamental possibility results were proposed [2], [3], [4], [5], and [6]. These results show that under appropriate conditions, it is possible for $n$ parties $P_1, P_2, \ldots, P_n$, each holding a private input $x_i$, to distributively compute a function $(y_1, y_2, \ldots, y_n) = f(x_1, x_2, \ldots, x_n)$ so that at the end of the computation, $P_i$ learns $y_i$ and nothing else beyond it. Though brilliant in the theoretical sense, the fundamental possibility results are not adequately efficient to be applied in most practical scenarios. Therefore, researchers have put a constant effort into seeking efficient solutions for secure computations. This has resulted in a considerable body of literature.

From a practical point of view, it is important to take a closer look at primitives and sub-protocols of SC protocols. Most SC protocols are composed of simpler primitives or sub-protocols such as oblivious transfer (OT), commitment, multiplication, or comparison which will be called repeatedly during the execution of the SC protocol. Consequently, little efficiency improvements in these sub-protocols will lead to considerable efficiency improvements in the overall SC protocols.

In this paper, we deal with secure computation of two-party scalar product. This operation constitutes the central primitive for a wide range of practical SC scenarios including pattern matching [7], dataset operations [8], data mining [9], [10], scientific computations [11], statistical analysis [12], optimization [13], [14], supply-chain management [15], [16], trust computation [17], benchmarking [18], [19] and secure cloud computing [20], [21]. It is also the core building block of various fundamental operations

The associate editor coordinating the review of this manuscript and approving it for publication was Malik Najmus Saqib.

in linear algebra (*e.g.* multiplication of matrices and solving systems of equations [22]).

We consider secure computation in the preprocessing model. This model is a relaxed setting for secure computations which comprises two separate phases: a preprocessing phase followed by a computation phase. During the preprocessing phase, each party receives a set of specific data, *pre-distributed data*. In the computation phase, parties run among themselves a secure computation protocol which uses the pre-distributed data as auxiliary inputs. During recent years, this model has become highly popular and been widely utilized in secure computations because, on the one hand, protocols in the preprocessing model are mostly simpler and more efficient than their counterparts in the plain model, and, on the other hand, this model can be exploited to achieve high levels of security.

The main goal of using the preprocessing model is to achieve efficiency. Typically, the pre-distributed data can be arranged in such a way that make the computation phase extremely simple and fast. It is worth to mention that preprocessing model can be realized with or without a third party. Specifically, for realization with a third party, pre-distributed data may be produced and distributed by a semi-trusted entity, often referred to as the *initiator*, or be purchased, in the form of a *commodity*, on the Internet [23]. Nevertheless, the preprocessing phase is an instance of secure computations and can be computed distributively by parties themselves without participation of any third party. A popular and effective idea is to use variants of homomorphic encryption to produce and distribute correlated randomness. This idea has been proposed in [24] and adopted in various dishonest majority SC protocols including [25], [26], [27]. In either form, the preprocessing phase is off-line and can be held any time before the main computation phase. Therefore, complexity of this phase is not of vital importance.

In addition to the mentioned practical advantages, preprocessing model helps to achieve higher levels of security. In particular, in this model it is possible to have unconditionally secure and universally composeble protocols for various functionalities [28], [29]. These security levels are important in practice. Unconditional security has two distinct advantages over computational security: (1) It is not based on intractability assumptions, and (2) unconditionally secure protocols impose less computational complexity in most cases. Universal composability framework [30] is a theoretically firm and practically influential security model that guarantees a strong composability property: A protocol which is composed of multiple UC secure protocols is UC secure. It is important to note that there are theoretical impossibility results on achieving these levels of security in the plain model (the model without extra set-up assumptions which assumes only pairwise private and authenticated channels between parties) [2], [31], [32], [33].

In this paper, exploiting capacities of the preprocessing model, we achieve unconditional security and universal composability (UC) in computing two-party scalar product.

### A. CONTRIBUTIONS

In this paper, we investigate secure two-party computation in the preprocessing model with an special emphasis on secure computation of two-party scalar product. In summary, the contributions of this paper are as follows:

#### 1) LOWER BOUNDS ON COMMUNICATIONS COMPLEXITY

It is valuable to understand limitations of secure computations models to fully exploit capabilities of them. From this point of view, important questions about the preprocessing model arise. In particular, a relevant question is to what extent can we improve efficiency of protocols in this model? As an effort at addressing this open question, we derive lower bounds on communicated bits during a two-party unconditionally secure computation in the preprocessing model. The lower bounds derived in this paper are generalizations of the work of [34] where lower bounds have been extracted for two-party oblivious polynomial evaluation (OPE). We stress that the results of [34] only apply to a limited functionalities. Specifically, these bounds only apply to two-party *linear* functionalities in which only one of the parties receives output while the other party has no output. Obviously, these bounds do not apply to general functionalities, including SSP. The lower bounds that we derive in this paper, although not general, hold for a wider range of functionalities. In particular, the extracted bounds apply to all (linear or nonlinear) functionalities in which input-output pair of each party reveals no information about the other party's input or output. We term such functionalities *private-output* and formally define the term in Section III.

#### 2) A SECURE SSP PROTOCOL

We propose an efficient SSP protocol which is perfectly secure, universally composable, and optimally efficient. Our protocol is designed in the preprocessing model. We rely on this model to achieve desired properties. Particularly, relying on the facilities of this model, our protocol achieves (perfect) unconditional security in the UC framework. The proposed secure SSP protocol is also optimally efficient in terms of communicated bits in both of the preprocessing and the computation phases. In the preprocessing phase, each party receives $(k+1)\sigma$ bits of pre-distributed data from the initiator where $k$ denotes the length of input vectors and $\sigma$ is the number of bits required to represent the underlying field upon which our protocol is constructed. In the computation phase, each party sends $k\sigma$ bits of data to the other party. We will show that these values are optimal for two-party SSP protocol in the preprocessing model. In the proposed protocol, each party sends a single message in the computation phase independently from the other party's message. Consequently, the number of communication rounds is one which is optimal as well.

In [22] and [35], various higher level privacy preserving linear algebra protocols are proposed that the SSP protocol is their fundamental building block. These higher level protocols use the SSP protocol as a black-box and, thus, they

can be implemented using any universally composable secure SSP protocol. Our SSP protocol (proposed in Section IV) is UC perfectly secure and, therefore, it enables the privacy preserving linear algebra protocols of [22] and [35].

#### 3) SECURE CLOUD COMPUTING APPLICATIONS

We provide secure solutions for two cloud computation scenarios based on the proposed SSP protocol. The first solution is a cloud-assisted privacy-preserving profile-matching (PPM) scheme with application in social media networks. PPM schemes contain mechanisms to privately store and maintain profile data of users on cloud servers while enabling the servers to process the data and calculate measures of matching without learning any information about user profiles. Therefore, they enable users to search for new friends with similar interests, activities, locations and other factors. Compared to relevant schemes, the proposed PPM scheme improves efficiency and security level and also enables additional functionality features.

The other solution is a cloud-assisted secure remote health monitoring (SRHM) scheme. Using SRHM schemes, cloud servers store and process health data of users without learning any information about them. Medical service providers can also queries on health data to check users health condition. The proposed SRHM scheme is unconditionally secure and highly efficient.

### B. RELATED WORK

Various forms of secure scalar product functionality are defined and studied in the literature. These functionalities are different regarding the number of parties, the arrangement of parties providing inputs, or the arrangement of parties receiving outputs. Here, we deal with two-party setting where each party provides a private vector as its input and, at the end of computation, receives a private additive share of the resulting scalar product. This functionality, formally defined in Fig. 1, is occasionally referred to as *split scalar product* (SSP) [10] because of its shared output.

There are several attempts at secure realization of scalar product functionalities in the literature. In [36], [37], [38] and [39], the two-party SSP functionality is considered and solutions without relying on a third party or intractable problems are targeted. Unfortunately, all of these solutions trade off security for efficiency. Specifically, in these solutions an amount of private information leaks during protocol run while such an information leakage does not occur in the ideal SSP functionality (see Fig. 1). Therefore these solutions deviate from the ideal functionality and are insecure according to rigorous standard security definitions (*i.e.* Definitions 1 and 2 of Section II). In fact, it has been proved that it is not possible to have secure two-party computation without relying on intractable problems or relaxed computation models [4]. In light of this general fact, we can conclude that the attempts of [36], [37], [38] and [39] have been condemned to failure from the beginning. Therefore, we do not discuss such solutions further.

A number of SSP protocols, *e.g.* [10], [40] and [41], utilize homomorphic encryption to fulfill security requirements. A one-sided unconditionally secure SSP protocol has been proposed in [10]. This protocol achieves unconditional security only against one of the parties while it uses semantically secure homomorphic encryption to establish security against the other party. Therefore, its security depends on security of the underlying homomorphic cryptosystem. In [40], based on the idea of permutation protocol of [12], a simple SSP protocol has been proposed. The core of this protocol is the realization of permutation protocol in the two-party setting with the help of homomorphic encryption. The idea of [41] for implementing scalar product is to transform secure scalar product into secure set intersection using homomorphic encryption and to implement set intersection using oblivious transfer. This method apply to binary vectors of limited density (*i.e.* proportion of 1 elements in a binary vector). We stress that among these protocols, only the one from [10] has a security proof.

The protocols proposed in [12], [42], [43], [35], and [22] are the most relevant to the work of this paper since they focus on the case of unconditionally secure SSP protocols in the relaxed computation models. In [12], an SSP protocol in the OT-hybrid model has been proposed and claimed to be unconditionally secure. Nevertheless, it has been shown in [10] that the protocol of [12] is insecure if inputs are binary vectors. In [42], a secret sharing-based SSP protocol has been proposed which uses the idea of multiplication on secret shared values of [4]. This protocol introduces a third party to whom each party sends a share of its private value. Then, the distributed addition and multiplication are carried out on shares of the three parties. The multiplication needs two communication rounds. In this protocol, multiple parties can jointly perform the role of the third party to have a distributed trust. In [43], [35], and [22] the approach of relying on the preprocessing model is adopted to achieve unconditionally secure SSP protocols. However, these proposals do not completely solve the problem. The protocol of [43], although seemingly secure, lacks a formal proof of security. This protocol is directly extended to a secure matrix multiplication protocol in the preprocessing model in [44]. The SSP protocol of [35] is not secure as discussed in [45]. The same insecurity arguments also apply to [22] which is the extended version of [35].

It is relevant to note that every (composable) secure protocol for computing two-party multiplication (with the result shared between parties) can be extended to an SSP protocol by composing copies of it in parallel. For instance, we can construct an SSP protocol in the preprocessing model using Beaver's pre-distributed multiplication triplets [46].

In previous studies it remains unaddressed that how efficient could an SSP protocol be in the preprocessing model? We answer this question by proposing lower bounds on the communications complexity of secure SSP protocols in the preprocessing model and also by proposing an optimal SSP protocol in this model. Efficiency of other forms of scalar

product in the preprocessing model has been investigated in [47]. Specifically, in the case that just one party learns the product and the other party has no output, it is showed that unconditionally secure protocols exist in which the number of bits communicated by each party (in computation phase) is linear in its input length. The protocol proposed in [47] uses an exponential (in input length) number of pre-distributed random bits.

### C. ORGANIZATION

We provide preliminary concepts, including security definitions, in Section II. In Section III, we focus on efficiency of secure two-party computations and derive lower bounds on communications complexity of such computations. Then we propose our SSP protocol and prove its security and optimality in Section IV. The applications of the proposed SSP protocol in secure cloud computation is demonstrated in Section V. We draw our conclusions and discuss future directions in Section VI.

## II. PRELIMINARIES

In this section, we provide definitions of security model as well as functionality. Specifically, we define two-party perfect security and discuss its relation to the universal composability framework. We also define split scalar product functionality which will be dealt with in the subsequent sections.

We use security definitions of [48, Ch. 7], [49, Ch. 2], and [50] to define perfect security with abortion for two-party secure protocols. The definitions are based on ideal/real paradigm. In [48, Ch. 7] and [49, Ch. 2], based on this paradigm, definitions of computational security with abortion in the two-party setting are provided. In [50], perfect security for the multiparty case is precisely defined.

### A. NOTATIONS

We consider a two-party protocol to be a process of computing a possibly probabilistic functionality $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ where, $(x_1, x_2) = (f_1(x_1, x_2), f_2(x_1, x_2))$. In an execution of a two-party protocol $\pi$ on inputs $(x_1, x_2)$, we denote the output and the view of the $i$th party $P_i$ by $OUT_i^{\pi}(x_1, x_2)$ and $VIEW_i^{\pi}(x_1, x_2)$ respectively. We denote by $OUT^{\pi}(x_1, x_2) = (OUT_1^{\pi}, OUT_2^{\pi})$ the pair of outputs. The view is defined as $VIEW_i^{\pi}(x_1, x_2) = (x_i, r_i, m_1, \ldots, m_l)$ where $x_i$ is $P_i$'s private input, $r_i$ is its internal random, and $m_j$ is the $j$th message that it receives during the protocol execution.

Suppose that $\Phi$ and $\Psi$ are two ensembles of random variables. We denote by $\Phi \equiv \Psi$ the identicalness of distributions of $\Phi$ and $\Psi$.

All values and computations throughout this paper are in a field of characteristic $p$ (a prime). We denote the underlying field by $Z_p$, the set $Z_p - 0$ by $Z_p^*$, and the number of bits required for presenting the elements of $Z_p$ by $\sigma$. That is, $\sigma = \log_2 |p|$. We also use over-bar lowercase letters to represent vectors in ${Z_p}^k$ or $Z_p^{*k}$ (*e.g.* $\bar{a}$ represents a vector), and uppercase letters to represent matrices in ${Z_p}^{k \times k'}$ or $Z_p^{*k \times k'}$ (*e.g.* $A$ represents a matrix).

### B. TWO-PARTY PERFECT SECURITY AGAINST SEMI-HONEST ADVERSARY

Semi-honest adversaries follow the protocol instructions faithfully. Hence, privacy is the only security aspect we are worried about in this setting. In Definition 1 we define a simulation-based privacy for two-party computations. The definition is restricted to stand-alone executions and to static adversaries. In this definition the simulator algorithm receives the input and the output of the corrupted party and constructs a view that is distributed the same as its view of the protocol execution.

*Definition 1 (privacy of Two-Party Protocols):* Let $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ be a probabilistic two-party functionality and let $\pi$ be a protocol. We say that $\pi$ privately computes $f$ if there exists a probabilistic polynomial-time algorithm $S$ such that for every $I \in \{1, 2\}$, and every $(x_1, x_2)$ where $|x_1| = |x_2|$, it holds that:

$$\{S(I, x_I, f_I(x_1, x_2)), f(x_1, x_2)\} \equiv \{(VIEW_I^{\pi}(x_1, x_2), OUT^{\pi}(x_1, x_2))\}. \quad (1)$$

### C. TWO-PARTY PERFECT SECURITY (WITH ABORT) AGAINST MALICIOUS ADVERSARIES

A malicious adversary may arbitrarily deviate from protocol instructions. For instance, it may not use the actual input, choose its input depending on the other party's input, or cause the output to be incorrectly distributed. Consequently, it does not suffice to simulate the view of the corrupted party.

We consider malicious adversaries with abortion and we use ideal/real paradigm based definition of security to analyze the security of protocols in the presence of such adversaries. We refer the reader to [49, Ch. 2] for details of the ideal world execution of two-party functionalities. The only difference between our model and the one of [49, Ch. 2] is the assumption on the computational power of the adversary. Since we are dealing with perfect security, we assume no limit on the computational power of the adversary. We also note that the security definition is restricted to stand-alone executions and to static adversaries.

Let $f$ be a two-party functionality and $\pi$ be a two-party protocol computing $f$. Let the real-world adversary $A$ be an arbitrary machine with auxiliary input $z$, and denote by $S$ an ideal-world adversary/simulator of comparable complexity (that is, it runs in time polynomial in the running time of $A$). We denote by $REAL_{(\pi,A(z),I)}(x_1, x_2)$ the outputs of the adversary A and the honest party following an execution of $\pi$. We also denote by $IDEAL_{(f,S(z),I)}(x_1, x_2)$ the corresponding outputs of the ideal-world adversary $S$ and the honest party after an ideal execution with a trusted party computing $f$.

*Definition 2 (Security of Two-Party Protocols With Abortion):* Let $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$ be a two-party functionality and let $\pi$ be a two-party protocol.

We say that $\pi$ securely computes $f$ with abort in the presence of malicious adversaries if for every probabilistic adversary $A$ in the real model, there exists a probabilistic adversary $S$ (of comparable complexity) in the ideal model with abortion, such that for every $I \in \{1, 2\}$, and every $(x_1, x_2)$ where $|x_1| = |x_2|$, it holds that:

$$\{IDEAL_{(f,S(z),I)}(x_1, x_2)\} \equiv \{REAL_{(\pi,A(z),I)}(x_1, x_2)\}. \quad (2)$$

### D. UNIVERSAL COMPOSABILITY FRAMEWORK

A protocol which is secure in the UC framework can be used in conjunction with other UC protocols (including copies of it) in any form of composition such as sequential, parallel, or concurrent. In the UC framework, security is defined based on protocol emulation paradigm. Loosely speaking, a protocol is said to emulate another one, if no environment can distinguish the executions. Assume that protocol $\rho^*$ securely computes functionality $f$. If protocol $\pi$ emulates protocol $\rho^*$ such that no environment can tell them apart, then $\pi$ is also a secure protocol for computing functionality $f$. We refer the reader to [30] for comprehensive formal discussions on UC framework.

The direct methodology for proving a protocol UC secure is to show that the protocol emulates the desired functionality defined in the ideal world of UC framework. Here we adopt a simple indirect proof procedure. Instead of working with UC framework, we prove that stand-alone execution of the protocol is perfectly secure in the standard model. Then we take advantage of the following result of [51], which guarantees UC security for perfectly secure protocols.

*Theorem 1: Let $\pi$ be a protocol that computes function $f$ with perfect security with abort in the stand-alone model, with a black-box straight-line simulator. Then, protocol $\pi$ computes $f$ with perfect security under concurrent general composition. This holds for both static and adaptive adversaries.*

### E. SPLIT SCALAR PRODUCT FUNCTIONALITY

Scalar product of two vectors $\bar{x}$ and $\bar{y}$ (of length $k$) is defined as $\langle \bar{x} \cdot \bar{y} \rangle = \sum_{j=1}^{k} x_j y_j$ where $x_j$ and $y_j$ denote the $j$th element of $\bar{x}$ and $\bar{y}$ respectively. In split scalar functionality we assume that $P_1$ and $P_2$ hold private vectors $\bar{x}_1 \in {Z_p}^k$ and $\bar{x}_2 \in {Z_p}^k$ respectively and intend to calculate $w = \langle \bar{x}_1 \cdot \bar{x}_2 \rangle$ such that for $i = 1, 2$ party $P_i$ receives an additive random share $w_i \in Z_p$ satisfying $w_1 + w_2 = w$.

In the ideal world, this functionality is handled by TTP as illustrated in Fig. 1. Since TTP honestly follows the illustrated procedure, output shares are uniformly distributed on $Z_p$ and parties learn nothing about $w$ or other party's input.

*Remark 1:* It is of vital importance to note the difference between SSP and conventional scalar product. In the latter, parties always (even in the ideal world) can derive an equation for the other party's input using their own input and output. Derivation of such an equation is not possible in the case of SSP ideal functionality.

**Functionality $F_{SSP}$**

**Inputs:** $P_1$ and $P_2$ hold respectively $\bar{x}_1 \in {Z_p}^k$ and $\bar{x}_2 \in {Z_p}^k$. Adversary $S$ holds an auxiliary input $z$.
**Notation:** $I \in \{1, 2\}$ and $J = \{1, 2\} - I$ denote the indices of corrupted and honest parties respectively.

$P_J$ : Sends $\bar{x}_J$ to TTP.
$P_I$: Sends $\bar{x}'_I$ to TTP, which may be an special early abortion message $abort_I$, its input $(\bar{x}_I)$ , or some other value based on adversaries auxiliary input $z$.
TTP: Upon receiving $\bar{x}_J$ and $\bar{x}'_I$,

- if $\bar{x}'_I = abort_I$, sends $w_1 = w_2 = abort_I$ to the parties and terminates the execution,
- if $\bar{x}'_I \notin {Z_p}^k$, sends $w_1 = w_2 = \lambda$ to the parties and halts,
- otherwise, chooses $r \in_R Z_p$ , sets $w_1 = r$ and $w_2 = \langle \bar{x}_1 \cdot \bar{x}_2 \rangle - r$, and sends $w_I$ to the ideal adversary $S$.

$S$: Sends the special message *order* to TTP.
TTP: Upon receiving adversary's order,

- if $order = continue$, sends $w_J$ to $P_J$,
- if $order = abort_I$, sends $w_J = abort_I$ to $P_J$,
- if $order = \lambda$, sends $w_J = \lambda$ to $P_J$.

**FIGURE 1. The ideal functionality of split scalar product with abortion ($F_{SSP}$).**

## III. LOWER BOUNDS ON COMMUNICATIONS COMPLEXITY

In this section, we consider a class of two-party functionalities and derive lower bounds on the amount of data communicated during the run of protocols that securely realize these functionalities in the preprocessing model. Specifically, we prove a lower bound on the amount of random bits that each party receives in the preprocessing phase as well as a lower bound on the total exchanged bits in the computation phase.

The results of this section can be seen as a generalization to the lower bounds of [34]. In [34], Tonicelli *et al.* have derived lower bounds on the communications complexity of secure oblivious polynomial evaluation (OPE) protocols in the preprocessing model. They have also proved the tightness of the bounds by proposing an OPE protocol secure against malicious adversary which achieves the bounds. We emphasize that the lower bounds presented in [34] only apply to the class of linear *sender-receiver* functionalities which take inputs from both parties and deliver an output only to the receiver (*e.g.* OPE). We show that similar bounds hold for a larger class of two-party functionalities including SSP and OPE (see Definition 3).

We will consider the semi-honest adversarial model and derive lower bounds in this setting. Since, for virtually all functionalities, protocol complexity in the malicious setting is not less than that in the semi-honest setting, we can say that the derived bounds hold for malicious setting as well. Notice that the derived bounds are not necessarily tight. For SSP

functionality, we present a protocol, secure against malicious adversary that achieves the bounds (see Section IV). This proves tightness of the lower bounds for SSP functionality.

To reach our goal of deriving the intended lower bounds, we need to restate the definition of unconditional security of two-party computations in terms of information theoretic conditions. Such conditions in the malicious adversary setting have been given in [52]. Here we need to consider the semi-honest adversarial model as we intend to derive lower bounds for this setting.

### A. INFORMATION THEORETIC MODELING OF UNCONDITIONALLY SECURE TWO-PARTY COMPUTATION

In the following we present information theoretic conditions governing perfectly secure two-party protocols in the semi-honest adversarial model.

*Theorem 2: Let $f$ be a probabilistic two-party functionality and let $\pi$ be a protocol. We say $\pi$ securely computes $f$ perfectly in the presence of semi-honest adversary if and only if the following conditions hold:*

- *Correctness: For every* $(x_1, x_2)$

$$P_{O_1^\pi O_2^\pi | X_1 X_2} = P_{f_1 f_2 | X_1 X_2}. \tag{3}$$

- *Privacy: For every* $I \in \{1, 2\}$*, and every* $(x_1, x_2)$ *where* $|x_1| = |x_2|$,

$$H(X_I O_I^\pi \mid V_J^\pi(x_1, x_2) O_J^\pi) = H(X_I O_I^\pi \mid X_J O_J^\pi). \tag{4}$$

*Proof:* We show these conditions are necessary and sufficient. Necessity of privacy and correctness is axiomatic considering the security requirements of secure computation in the presence of semi-honest adversary. So, we simply need to explain why these requirements can be expressed as above when dealing with perfect security. In the most general case, which includes probabilistic functionalities, perfect correctness can be formulated in terms of identicalness of distributions. Specifically, protocol $\pi$ is correct if the distribution of its output is identical to the distribution of the functionality's output. Mathematically, we can write it as $O^\pi(x_1, x_2) \equiv f(x_1, x_2)$ and, regarding the definition of distribution identicalness, we have

$$P_{O_1^\pi O_2^\pi | X_1\, X_2} = P_{f_1\, f_2 | X_1\, X_2}.$$

Perfect privacy for $P_I$ requires that after protocol execution, $P_J$ gains no information about $x_I$ and $O_I^\pi$ beyond what can be inferred from $x_J$ and $O_J^\pi$. In other words, given $x_J$ and $O_J^\pi$, mutual information of $P_J$'s view and $P_I$'s input-output pair should be zero. We can write it in the form of $I(X_I O_I^\pi; V_J^\pi(x_1, x_2) \mid X_J O_J^\pi) = 0$ and thus,

$$H(X_I O_I^\pi \mid V_J^\pi(x_1, x_2) X_J O_J^\pi) - H(X_I O_I^\pi \mid X_J O_J^\pi) = 0.$$

Since $X_J$ is a part of $V_J^\pi(x_1, x_2)$, we have

$$H(X_I O_I^\pi \mid V_J^\pi(x_1, x_2) O_J^\pi) = H(X_I O_I^\pi \mid X_J O_J^\pi).$$

This completes the proof of necessity. To prove that the correctness and privacy conditions are sufficient, we show they together imply the simulation based security definition (Definition 1). Specifically, we show that if protocol $\pi$ is correct and private as defined above, there is a simulator $S$ that can construct $P_I$'s view using its input-output pair only.

We can write privacy condition for $P_J$ in the form of conditional mutual information

$$I(X_J O_J^\pi; V_I^\pi(x_1, x_2) \mid X_I O_I^\pi) = 0.$$

This equation means that given $X_I$ and $O_I^\pi$, knowledge on $X_J$ and $O_J^\pi$ has no information about $V_I^\pi(x_1, x_2)$. This implies the view of $P_I$ can be constructed using $P_I$'s input-output pair with no need for private values of $P_J$. Therefore, provided privacy condition for $P_J$, there exists a simulator $S$ such that for every $(x_1, x_2)$ it holds that:

$$S(I, x_I, O_I^\pi(x_1, x_2)) \equiv V_I^\pi(x_1, x_2).$$

This is not the end of the proof since we need to show identicalness of joint distributions of view and outputs. Moreover, the simulator should be fed with $f_I$ instead of $O_I^\pi$. Consider the following events:

- $A =$ "$S(I, x_I, f_I(x_1, x_2)) \equiv V_I^\pi(x_1, x_2)$",
- and $B =$ "$f(x_1, x_2) \equiv O^\pi(x_1, x_2)$".

Thus,

- $A \mid B =$ "$S(I, x_I, O_I^\pi(x_1, x_2)) \equiv V_I^\pi(x_1, x_2)$",
- and $AB = \{S(I, x_I, f_I(x_1, x_2)), f(x_1, x_2)\} \equiv \{(V_I^\pi(x_1, x_2), O^\pi(x_1, x_2))\}$.

It is clear that event $B$ is equivalent to correctness and event $A \mid B$ is equivalent to privacy and the goal is to reach to event $AB$. But, these events are related under chain rule (*e.g.* $Pr(AB) = Pr(A \mid B)Pr(B)$) which is interpreted as stating that provided correctness and privacy for $P_J$, there exists a simulator $S$ such that for every $(x_1, x_2)$ it holds that:

$$\{S(I, x_I, f_I(x_1, x_2)), f(x_1, x_2)\} \equiv \{(V_I^\pi(x_1, x_2), O^\pi(x_1, x_2))\}.$$

■

### B. REMARKS ON FUNCTIONALITIES AND ON VIEWS OF PARTIES

We classify two-party functionalities into private-output and non-private-output functionalities and consider private-output functionalities in the remainder of this section. In an ideal run of a private-output functionality, conditioned on having access to input-output pair of one party, no information is revealed about the other party's input. Formally, we have the following definition.

*Definition 3 (Private-Output Functionality):* We say that $f$ is private-output two-party functionality if for every $I \in \{1, 2\}$ and $J \neq I$, $H(x_I \mid x_J O_J^\pi) = H(x_I)$. Otherwise we say that $f$ is non-private-output two-party functionality.

From a technical perspective, assuming the functionalities to be restricted to private-output enables us to prove Proposition 3 of the next subsection. Without this assumption, steps that we will take in the next subsection will result in trivial bounds.

Although not holding in general, the derived bounds cover a large class of functionalities required in secure computations. In particular, all functions with shared output are in this class.

Views of parties in a protocol in the preprocessing model contain preprocessed data in addition to inputs, local randomness and communicated messages. Therefore, we define the view of party $P_I$ after execution of protocol $\pi$ in the preprocessing model as $V_I^{\pi}(x_1, x_2) = (U_I, x_I, R_I, C_I)$, where $U_I$ is the set of all values that $P_I$ receives in the preprocessing phase and $C_I$ stands for the set of all messages that $P_I$ receives from $P_J$.

### C. LOWER BOUNDS

In this section, we derive lower bounds on the communications complexity of secure two-party computation of general private-output functionalities in the preprocessing model. We start with showing the fact that protocol execution binds each party's input to its preprocessed data so that learning the preprocessed data, reveals all information on the input.

*Proposition 1: Let $f$ be a two-party functionality and let $\pi$ be a two-party protocol computing $f$ unconditionally privately in the preprocessing model. Assume that $P_I$ and $P_J$ execute an instance of protocol $\pi$. After protocol completion, $P_I$ learns $x_J$ if it has access to $U_J$. In mathematical terms, $H(x_J \mid C_I C_J U_I U_J) = 0$.*

*Proof:* After completion of the protocol (we will call it the main run), having access to $U_J$, $P_I$ locally runs instances of protocol $\pi$ for all possible inputs of $P_J$ (we will denote it by $\hat{x}_J$) using the same values of $x_I$, $U_I$, and $U_J$ which were used in the main run of the protocol. $P_I$ then compares the produced view of each local run to its view of the main run. If the views are the same in the two runs, the corresponding value used as $\hat{x}_J$ is a candidate for $P_J$'s input to the main protocol ($x_J$). Let $\mathcal{E}$ be the set of all candidates for $x_J$. To prove $H(x_J \mid C_I C_J U_I U_J) = 0$, we need to show that $\mathcal{E}$ contains a single member. We note that, on the one hand, $\mathcal{E}$ is not empty since $x_J$ is in it. On the other hand, because of information theoretic privacy for $P_I$, $\mathcal{E}$ cannot accept more than one member. In terms of simulation based security, if $|\mathcal{E}| > 1$, there is no simulator being able to construct the view of $P_J$ only by accessing to $x_J$ and $O_J^{\pi}$. This is because the view of $P_J$ has a distinctive property which depends on $x_I$ and $U_I$ in addition to $x_J$ and $O_J^{\pi}$. Specifically, for the given $x_I$, $U_I$, and $U_J$, there are more than one $x_J$ producing same views (note that the simulator cannot find the values of these $x_J$s because it does not know the $x_I$). ■

In the following, using the privacy condition, we show properties that the communications between parties should possess in secure two-party computation of a general private-output functionality in the preprocessing model. Upon these properties we will build lower bounds on the communications complexity.

*Proposition 2: In an unconditionally secure two-party protocol in the preprocessing model, $I(x_I; C_I C_J) \leq I(x_I; x_J O_J^{\pi})$.*

*Proof:*

$$\begin{aligned} I(x_I; C_I C_J) &\leq I(x_I; C_I C_J U_J x_J O_J^{\pi}) \\ &= I(x_I; x_J O_J^{\pi}) + I(x_I; C_I C_J U_J \mid x_J O_J^{\pi}) \\ &= I(x_I; x_J O_J^{\pi}). \end{aligned}$$

The $I(x_I; C_I C_J U_J \mid x_J O_J^{\pi}) = 0$ holds due to the privacy condition for $P_I$. ■

*Corollary 1: In an unconditionally secure two-party protocol computing a private-output functionality in the preprocessing model, $I(x_I; C_I C_J) = 0$.*

*Proof:* Due to Definition 3, for a private-output functionality $I(x_I; x_J O_J^{\pi}) = 0$. Therefore, $I(x_I; C_I C_J) = 0$. ■

*Proposition 3: In an unconditionally secure two-party protocol computing a private-output functionality in the preprocessing model, $I(x_I O_I^{\pi}; C_I C_J) = 0$.*

*Proof:* Due to Corollary 1 we have that

$$\begin{aligned} I(x_I f_I; C_I C_J) &= I(x_I; C_I C_J) + I(f_I; C_I C_J \mid x_I) \\ &= I(f_I; C_I C_J \mid x_I). \end{aligned}$$

Now we show that privacy for $P_I$ necessarily entails $I(f_I; C_I C_J \mid x_I) = 0$. Otherwise, after protocol completion, $P_J$ tries all possible values for $x_I$ in the following experiment:

For each value,

- it computes the functionality on the current $x_I$ and its input $x_J$ to get $f_I(x_1, x_2)$,
- it observes the communicated messages and extracts information on $f_I$ conditioned on the current $x_I$,
- it excludes the current $x_I$ and puts it in a set $\mathcal{E}$, if the extracted information is not consistent with functionality output.

Note that since we have assumed $I(f_I; C_I C_J \mid x_I) \neq 0$, set $\mathcal{E}$ is not empty. Furthermore, inconsistency indicates that excluded values could not be used by $P_I$ as input to the protocol execution. But, this contradicts privacy condition for $P_I$ because such information on $P_I$'s input cannot be deduced from $x_J$ and $O_J^{\pi}$ as $I(x_I; x_J O_J^{\pi}) = 0$. Therefore, $I(x_I f_I; C_I C_J) = 0$ and considering correctness, $I(x_I O_I^{\pi}; C_I C_J) = 0$. ■

*Proposition 4: In an unconditionally secure two-party protocol computing a private-output functionality in the preprocessing model, $H(x_I O_I^{\pi} | C_I C_J U_J) = H(x_I O_I^{\pi})$.*

*Proof:* Due to Proposition 3, $x_I O_I^{\pi}$ and $C_I C_J$ are independent. Therefore, we can apply $I(x_I O_I^{\pi}; C_I C_J | U_J) = 0$ in the following

$$\begin{aligned} I(x_I O_I^{\pi}; C_I C_J U_J) &= I(x_I O_I^{\pi}; U_J) + I(x_I O_I^{\pi}; C_I C_J | U_J) \\ &= I(x_I O_I^{\pi}; U_J). \end{aligned}$$

From the definition of preprocessing model, pre-distributed data is independent of inputs. Specifically, $I(x_I; U_J) = I(x_J; U_J) = I(x_I x_J; U_J) = 0$. Using privacy condition for $P_I$ we have

$$\begin{aligned} I(x_I x_J; U_J) &= I(x_I f_I x_J f_J; U_J) \\ &= I(x_I f_I; U_J | x_J f_J) + I(x_J f_J; U_J) \\ &= I(x_J f_J; U_J). \end{aligned}$$

Now, considering $I(x_J f_J; U_J) = 0$, we can rewrite $I(x_I x_J; U_J)$ as

$$\begin{aligned} I(x_I x_J; U_J) &= I(x_I f_I x_J f_J; U_J) \\ &= I(x_J f_J; U_J | x_I f_I) + I(x_I f_I; U_J) \\ &= I(x_I f_I; U_J). \end{aligned}$$

Putting things together and applying the correctness condition, we conclude $I(x_I O_I^\pi; C_I C_J U_J) = 0$ and hence $H(x_I O_I^\pi | C_I C_J U_J) = H(x_I O_I^\pi)$. ■

*Theorem 3: Let $f$ be a private-output functionality and let $\pi$ be an unconditionally secure two-party protocol computing $f$ in the preprocessing model. The following lower bound holds on the size of the data pre-distributed to $P_I$ in the protocol $\pi$:*

$$H(U_I) \geq H(x_I) + H(O_I^\pi | x_I). \tag{5}$$

*Proof:*

$$\begin{aligned} H(U_I) &\geq I(U_I; x_I O_I^\pi | C_I C_J U_J) \\ &= H(x_I O_I^\pi | C_I C_J U_J) - H(x_I O_I^\pi | C_I C_J U_I U_J) \\ &= H(x_I O_I^\pi | C_I C_J U_J) \\ &= H(x_I O_I^\pi) \\ &= H(x_I) + H(O_I^\pi | x_I). \end{aligned}$$

In the third line, we have applied $H(x_I O_I^\pi | C_I C_J U_I U_J) = 0$ from Proposition 1 and in the fourth line, we have used $H(x_I O_I^\pi | C_I C_J U_J) = H(x_I O_I^\pi)$ from Proposition 4. ■

*Theorem 4: Suppose that $\pi$ is an unconditionally secure two-party protocol in the preprocessing model and for $I \in \{1, 2\}$, $C_I$ is a random variable representing communications in the computation phase sent to $P_J$ by $P_I$. Then we have the following bound on communications in the computation phase:*

$$H(C_I) + H(C_J) \geq H(x_I) + H(x_J). \tag{6}$$

*Proof:*

$$\begin{aligned} H(x_I) + H(x_J) &= H(x_I x_J) \\ &= I(x_I x_J; U_I U_J C_I C_J) \\ &= I(x_I x_J; U_I U_J) + I(x_I x_J; C_I C_J | U_I U_J) \\ &= I(x_I x_J; C_I C_J | U_I U_J) \\ &\leq H(C_I C_J | U_I U_J) \\ &\leq H(C_I C_J) \leq H(C_I) + H(C_J). \end{aligned}$$

We have used known identities and inequalities along with independence of $x_I$ and $x_J$ to write the first line, Proposition 1 to reach the second line and the independence of pre-distributed data and inputs (that is, $I(x_I x_J; U_I U_J) = 0$) to reach the fourth line. ■

## IV. OPTIMALLY EFFICIENT AND UNIVERSALLY COMPOSABLE SSP PROTOCOL

In this section, we present an SSP protocol in the preprocessing model. We prove that the proposed protocol is perfectly secure and universally composable in the presence of malicious adversary. We also discuss efficiency of the proposed protocol and show its optimality based on the lower bounds derived in Section III. Therefore, the proposed SSP protocol implies the tightness of lower bounds of Section III in the case of SSP functionality.

### A. THE PROPOSED SSP PROTOCOL

Fig. 2 depicts the proposed SSP protocol (we will refer to this protocol as $\pi_{SSP}$). In this protocol, $P_1$ and $P_2$ provide input vectors $\bar{x}_1$ and $\bar{x}_2$ respectively. We assume that an ideal preprocessing functionality, $F_{Pre}$, distributes correlated randomness to the players (through secure channels) in the preprocessing phase. This includes $(\bar{r}_1 \in {Z_p}^k, r_3 \in Z_p)$ for $P_1$ and $(\bar{r}_2 \in {Z_p}^k, r_4 \in Z_p)$ for $P_2$ where $r_3 + r_4 = \langle \bar{r}_1 \cdot \bar{r}_2 \rangle$. During the computation phase, $P_i$ generates message $\bar{v}_i = \bar{x}_i + \bar{r}_i$ and sends it to the other party. After this single communication round, parties will be able to calculate their outputs.

**Protocol $\pi_{SSP}$**

**Inputs:** $\bar{x}_1 \in {Z_p}^k$ by $P_1$ and $\bar{x}_2 \in {Z_p}^k$ by $P_2$.
**Result:** $P_1$ learns $w_1 \in Z_p$ and $P_2$ learns $w_2 = \langle \bar{x}_1 \cdot \bar{x}_2 \rangle - w_1$.

**Preprocessing Phase**

$P_1$ and $P_2$ run $F_{Pre}$ and receive $(\bar{r}_1 \in {Z_p}^k, r_3 \in Z_p)$ and $(\bar{r}_2 \in {Z_p}^k, r_4 \in Z_p)$ respectively where $r_3 + r_4 = \langle \bar{r}_1 \cdot \bar{r}_2 \rangle$.

**Computation Phase**

$P_1$: Sends $\bar{v}_1 = \bar{x}_1 + \bar{r}_1$ to $P_2$
$P_2$: Sends $\bar{v}_2 = \bar{x}_2 + \bar{r}_2$ to $P_1$
$P_1$: Waits until the end of current communication round. If it has received no message from $P_2$, outputs $w_1 = abort_2$ and quits the protocol. Otherwise, checks whether $\bar{v}_2 \in {Z_p}^k$:

- if $\bar{v}_2 \notin {Z_p}^k$, aborts and outputs $w_1 = \lambda$,
- otherwise, computes $w_1 = \langle -\bar{r}_1 \cdot \bar{v}_2 \rangle + r_3$

$P_2$: Waits until the end of current communication round. If it has received no message from $P_1$, outputs $w_2 = abort_1$ and quits the protocol. Otherwise, checks whether $\bar{v}_1 \in {Z_p}^k$:

- if $\bar{v}_1 \notin {Z_p}^k$, aborts and outputs $w_2 = \lambda$,
- otherwise, computes $w_2 = \langle \bar{x}_2 \cdot \bar{v}_1 \rangle + r_4$

**FIGURE 2.** The proposed split scalar product protocol ($\pi_{SSP}$).

### B. REALIZATION OF PREPROCESSING PHASE

As mentioned before, preprocessing model can be realized with or without a third party (*initiator*). To realize the preprocessing phase of $\pi_{SSP}$ with the help of a third party, the initiator runs the following instructions:

1) Choose two vectors $\bar{r}_1$ and $\bar{r}_2$ randomly from ${Z_p}^k$ and a scalar $r_3$ randomly from $Z_p$.
2) Compute $r_4 = \langle \bar{r}_1 \cdot \bar{r}_2 \rangle - r_3$.
3) Send $(\bar{r}_1, r_3)$ to $P_1$ through a secure channel with $P_1$.
4) Send $(\bar{r}_2, r_4)$ to $P_2$ through a secure channel with $P_2$.

Under the assumption that the initiator does not collude with the parties, the above procedure provides the required correlated randomness with perfect security.

On the other hand, parties can run any secure two-party SSP protocol to compute the required pre-distributed data without the help of a third party. Let $\pi^*$ be a secure SSP protocol. Party $P_i$ runs the following instructions to compute $(\bar{r}_i, r_{i+2})$:

1) Choose a random vector $\bar{r}_i$ from ${Z_p}^k$.
2) Participate in a run of protocol $\pi^*$ with the other party. Use $\bar{r}_i$ as your input.
3) Receive your output share from the protocol $\pi^*$ and set $r_{i+2}$ equal to the received share.

We can employ homomorphic encryption-based SSP protocols (*e.g.* the SSP protocol of [10]) in the above procedure. The important point is that the preprocessing phase is off-line, and therefore complexity of the employed protocol is not a major problem to be considered. It is also relevant to note that multiple instances of the above procedure can be executed in parallel and therefore advantages of parallelism can be utilized. For example, in parallel executions of the SSP protocol of [10], it suffices to perform a single setup step and use the result in all executions.

### C. SECURITY ANALYSIS

In this section we show that Protocol $\pi_{SSP}$ is perfectly UC secure. With this end in view, in Theorem 5 we show that Protocol $\pi_{SSP}$ is perfectly secure in the stand-alone standard model. We prove Theorem 5 using straight line simulators. Therefore, Theorem 5 will immediately result in UC perfect security of Protocol $\pi_{SSP}$ due to Theorem 1.

Correctness of Protocol $\pi_{SSP}$ is easily verifiable:

$$\begin{aligned} w_1 + w_2 &= (\langle -\bar{r}_1 \cdot \bar{v}_2 \rangle + r_3) + (\langle \bar{x}_2 \cdot \bar{v}_1 \rangle + r_4) \\ &= \langle -\bar{r}_1 \cdot (\bar{x}_2 + \bar{r}_2) \rangle + r_3 + \langle \bar{x}_2 \cdot (\bar{x}_1 + \bar{r}_1) \rangle \\ &\quad + \langle \bar{r}_1 \cdot \bar{r}_2 \rangle - r_3 \\ &= \langle \bar{x}_1 \cdot \bar{x}_2 \rangle \end{aligned}$$

To formally show the security of Protocol $\pi_{SSP}$, we prove the following theorem.

*Theorem 5: Protocol $\pi_{SSP}$ computes $F_{SSP}$ in the preprocessing model with perfect security in the presence of static malicious adversary.*

*Proof:* We use the ideal/real paradigm based on Definition 2. The ideal scenario is the TTP aided computation of $F_{SSP}$ (as depicted in Fig. 1). We construct an ideal world simulator $S$ which invokes the real world adversary $\mathcal{A}$ and uses it in an ideal execution of $F_{SSP}$ such that the outputs of this ideal run is identically distributed to the outputs of a run of protocol $\pi_{SSP}$ in the presence of $\mathcal{A}$. We consider two cases regarding the corrupted party. (Note that if both parties are malicious or if both parties are honest, Protocol $\pi_{SSP}$ is trivially secure. The latter is true due to correctness.)

*Case 1 (Corrupted $P_1$):* The simulator for corrupted $P_1$ is illustrated in Fig. 3. We need to show that for every $\bar{x}_1, \bar{x}_2 \in {Z_p}^k$, the joint distributions of outputs in the two executions are identical. We discuss three cases based on the adversary's action in sending the protocol specified message $(\bar{v}_1)$:

1) $\mathcal{A}$ does not respond: The real protocol outputs are $(w_1, w_2) = (r, abort_1)$ where $r = \langle -\bar{r}_1 \cdot \bar{v}_2 \rangle + r_3$ is uniformly distributed on $Z_p$. In the simulated ideal scenario, the outputs are $(w'_1, w'_2) = (r', abort_1)$ where $r'$, chosen by TTP, is a random variable uniformly distributed on $Z_p$. Thus, $w'_1 \equiv w_1$ and $w'_2 = w_2$ which indicates that $\{(w'_1, w'_2)\} \equiv \{(w_1, w_2)\}$.
2) $\mathcal{A}$ sends an invalid $\bar{v'}_1 \notin {Z_p}^k$: The argument is similar to above except that in this case $w'_2 = w_2 = \lambda$.

**Simulator (Corrupted $P_1$)**

$S$ follows the instructions of the following procedure

- Invoke $\mathcal{A}$ with its auxiliary input $z$.
- Simulate the preprocessing phase: choose $\bar{r'}_1 \in_R {Z_p}^k$ and $r'_3 \in_R Z_p$ and send them to $\mathcal{A}$.
- Simulate the honest party's message: choose $\bar{v'}_2 \in_R {Z_p}^k$ and send it to $\mathcal{A}$.
- Wait for $\mathcal{A}$'s response $(\bar{v'}_1)$ until the end of the current round:
  - if $\mathcal{A}$ has not responded, set $\bar{x'}_1 \in_R {Z_p}^k$ and $order = abort_1$,
  - else if $\bar{v'}_1 \notin {Z_p}^k$, set $\bar{x'}_1 \in_R {Z_p}^k$ and $order = \lambda$,
  - else, extract the input $\bar{x'}_1 = \bar{v'}_1 - \bar{r'}_1$ and set $order = continue$.
- Instruct $P_1$ in the ideal world to send $\bar{x'}_1$ to TTP as its input.
- Receive the ideal world output $w'_1$ from TTP and
  - if $w'_1 \neq \lambda$ and $w'_1 \neq abort_1$, send $order$ to TTP.

**FIGURE 3.** The simulation in the case that $P_1$ is corrupted.

**Simulator (Corrupted $P_2$)**

$S$ follows the instructions of the following procedure

- Invoke $\mathcal{A}$ with its auxiliary input $z$.
- Simulate the preprocessing phase: choose $\bar{r'}_2 \in_R {Z_p}^k$ and $r'_4 \in_R Z_p$ and send them to $\mathcal{A}$.
- Simulate the honest party's message: choose $\bar{v'}_1 \in_R {Z_p}^k$ and send it to $\mathcal{A}$.
- Wait for $\mathcal{A}$'s response $(\bar{v'}_2)$ until the end of the current round:
  - if $\mathcal{A}$ has not responded, set $\bar{x'}_2 \in_R {Z_p}^k$ and $order = abort_2$,
  - else if $\bar{v'}_2 \notin {Z_p}^k$, set $\bar{x'}_2 \in_R {Z_p}^k$ and $order = \lambda$,
  - else, extract the input $\bar{x'}_2 = \bar{v'}_2 - \bar{r'}_2$ and set $order = continue$.
- Instruct $P_2$ in the ideal world to send $\bar{x'}_2$ to TTP as its input.
- Receive the ideal world output $w'_2$ from TTP and
  - if $w'_2 \neq \lambda$ and $w'_2 \neq abort_2$, send $order$ to TTP.

**FIGURE 4.** The simulation in the case that $P_2$ is corrupted.

3) $\mathcal{A}$ sends a valid $\bar{v'}_1 \notin {Z_p}^k$: The real protocol outputs are $(w_1, w_2) = (r, \langle \bar{x'}_1 \cdot \bar{x}_2 \rangle - r)$ where $r = \langle -\bar{r}_1 \cdot \bar{v}_2 \rangle + r_3$ is a random variable uniformly distributed on $Z_p$ and $\bar{x'}_1$ is the adversary's input to the protocol. In the simulated scenario, the simulator extracts the adversary's input, $\bar{x'}_1$, and sends it to TTP. Therefore, the outputs of the simulated ideal scenario are $(w'_1, w'_2) = (r', \langle \bar{x'}_1 \cdot \bar{x}_2 \rangle - r')$ where $r'$, chosen by TTP, is a random variable uniformly distributed on $Z_p$. This indicates that $\{(w'_1, w'_2)\} \equiv \{(w_1, w_2)\}$.

*Case 2 (Corrupted $P_2$):* The security proof is very similar to Case 1. The simulator for this case illustrated in Fig. 4. Similar to the previous argument, we discuss the equality of the joint distributions in three cases based on the adversary's action in sending the protocol specified message ($\bar{v}_2$):

1) $\mathcal{A}$ does not respond: The real protocol outputs are $(w_1, w_2) = (abort_2, r)$ where $r = \langle \bar{x}_2 \cdot \bar{v}_1 \rangle + r_4$ is uniformly distributed on $Z_p$. In the simulated ideal scenario, the outputs are $(w'_1, w'_2) = (abort_2, r')$ where $r'$, chosen by TTP, is a random variable uniformly distributed on $Z_p$. Thus, $w'_1 \equiv w_1$ and $w'_2 = w_2$ which indicates that $\{(w'_1, w'_2)\} \equiv \{(w_1, w_2)\}$.
2) $\mathcal{A}$ sends an invalid $\bar{v'}_2 \notin {Z_p}^k$: The argument is similar to above except that in this case $w'_1 = w_1 = \lambda$.
3) $\mathcal{A}$ sends a valid $\bar{v'}_2 \notin {Z_p}^k$: The real protocol outputs are $(w_1, w_2) = (\langle \bar{x}_1 \cdot \bar{x'}_2 \rangle - r, r)$ where $r = \langle \bar{x'}_2 \cdot \bar{v}_1 \rangle + r_4$ is a random variable uniformly distributed on $Z_p$ and $\bar{x'}_2$ is the adversary's input to the protocol. In the simulated scenario, the simulator extracts the adversary's input, $\bar{x'}_2$, and sends it to TTP. Therefore, the outputs of the simulated ideal scenario are $(w'_1, w'_2) = (\langle \bar{x}_1 \cdot \bar{x'}_2 \rangle - r', r')$ where $r'$, chosen by TTP, is a random variable uniformly distributed on $Z_p$. This indicates that $\{(w'_1, w'_2)\} \equiv \{(w_1, w_2)\}$.

■

*Corollary 2: Protocol $\pi_{SSP}$ is perfectly secure against dynamic malicious adversary in the UC framework.*

*Proof:* The proof is straightforward since the requirements of Theorem 1 are satisfied for Protocol $\pi_{SSP}$. Specifically, on the one hand, Protocol $\pi_{SSP}$ is perfectly secure in the stand-alone standard model based on Theorem 5, and on the other hand the simulator that is constructed in order to prove the security in the standard model (see Fig. 3 or Fig. 4) is straight line. ■

*Remark 2: Protocol $\pi_{SSP}$ is UC secure against semi-honest adversary as well. We omit the proof.*

### D. EFFICIENCY AND COMPARISON

In this section we analyze Protocol $\pi_{SSP}$ from the efficiency point of view. First, using the lower bounds extracted in Section III, we show that Protocol $\pi_{SSP}$ is of optimal communications complexity. Furthermore, we compare the complexity of Protocol $\pi_{SSP}$ to other relevant SSP protocols.

*Theorem 6: Protocol $\pi_{SSP}$ is optimal regarding*

*a) the number of bits pre-distributed to each party in the preprocessing phase, and*

*b) the number of bits communicated between parties in the computation phase.*

*Proof:* a) In SSP functionality, for $I \in \{1, 2\}$ we have $H(O_I^{\pi} | \bar{x}_I) = H(O_I^{\pi})$, since the output is a random value uniformly chosen from $Z_p$. Consequently, based on Theorem 3, for a secure SSP protocol in the preprocessing model $H(U_I) \geq H(\bar{x}_I) + H(O_I^{\pi})$ where $U_I$ is the data pre-distributed to party $P_I$. On the other hand, $H(\bar{x}_I) = k\sigma$ for a vector of length $k$ whose elements are from $Z_p$ and $H(O_I^{\pi}) = \sigma$ for a random share uniformly distributed on $Z_p$ (where $\sigma = \log_2 |p|$). Therefore, for a secure SSP protocol on input vectors of length $k$ in the preprocessing model we have that

$$H(U_I) \geq (k + 1)\sigma$$

In protocol $\pi_{SSP}$, party $P_I$ receives $\bar{r}_I$ and $r_{I+2}$ in the preprocessing phase where $\bar{r}_I$ is a random vector from ${Z_p}^k$ and $r_{I+2}$ is a random scalar from $Z_p$. Thus, $(k + 1)\sigma$ bits are enough to pre-distribute the required randomness to each party.

b) Based on Theorem 4, for a secure SSP protocol in the preprocessing model $H(C_I) + H(C_J) \geq H(\bar{x}_I) + H(\bar{x}_J)$ where $C_I$ is the total massages sent by party $P_I$ in the computation phase. For a random vector $\bar{x}$ of length $k$ that its elements are in $Z_p$, we have $H(\bar{x}) = k\sigma$. Thus

$$H(C_I) + H(C_J) \geq 2k\sigma.$$

In protocol $\pi_{SSP}$, party $P_I$ needs to send to the other party a single message ($\bar{v}_I$) which is a random vector from ${Z_p}^k$ represented with $k\sigma$ bits. Thus, in total, $2k\sigma$ bits are exchanged during computation phase. ■

In Table 1 we compare the concrete efficiency of proposed SSP protocol with SSP protocols of [43], [22], [42], and a direct extension of Beaver's multiplication triplets [46]. The latter SSP protocol comprises parallel runs of two-party multiplication of [46] to multiply the corresponding elements of input vectors. All of these protocols are in the preprocessing model except the one from [42].

In the preprocessing phase, the proposed protocol and the protocol of [43] reach the optimal amount of total pre-distributed bits, while the extension of [46] is inefficient. A notable observation is that in the SSP protocol of [22], the amount of pre-distributed bits is less than the optimal value implied by the corresponding lower bound (see Theorem 3). This indicates excessive correlation of pre-distributed data which in fact is the origin of insecurity of the protocol (see [45]).

In the computation phase, the proposed protocol and the extension of [46] reach the optimal amount of communicated bits. Moreover, the proposed protocol runs in a single communication round thus is optimal in this sense too.

The SSP protocol of [42] uses a third party but not in an efficient manner. This protocol requires $4k\sigma$ bits to be communicated in order to share the input vectors among the three parties (*e.i.* the participants and the third party) and $6k\sigma$ bits to compute multiplications. This protocol requires 2 communication rounds.

**TABLE 1. Efficiency comparison SSP protocols.**

| SSP protocol | Pre-processing Phase | Computation Phase | |
|---|---|---|---|
| | Total pre-distributed bits | Total communicated bits | Communication rounds |
| Proposed protocol ($\pi_{SSP}$) | $2(k+1)\sigma$ | $2k\sigma$ | 1 |
| Protocol of [43] | $2(k+1)\sigma$ | $(2k+1)\sigma$ | 2 |
| Protocol of [22] | $(2k+1)\sigma$ | $(2k+1)\sigma$ | 2 |
| Protocol of [46] | $6k\sigma$ | $2k\sigma$ | 1 |
| Protocol of [42] | – | $10k\sigma$ | 2 |

## V. APPLICATIONS

The proposed SSP protocol can be applied in various practical computation scenarios. In this section, we present two applications of this protocol in secure cloud computing. Before describing the applications, we briefly explain secure computation on signed integers which will be needed in the remainder of this section. To achieve notation clarity, in this section, we will use superscripts to denote shares. Specifically, $x^i$ will denote the $i$th share of $x$ or $\overline{v}^{U_j}$ will denote $U_j$'s share of vector $\overline{v}$.

### A. COMPUTATION ON SIGNED INTEGERS

The set of $l$-bit signed integers is defined as $Z_{\langle l \rangle} = \{x \in Z \mid -2^{l-1}+1 \leq x \leq 2^{l-1}-1\}$. The first step to securely compute on signed integers is to map them to members of a field $Z_p$ where $p > 2^l$. We do it as $\tilde{x} = x \ mod \ p$. To reconstruct a signed value from the corresponding field value, inverse map should be computed. After mapping to a field member, we can secret share the resulting field element. We use additive secret sharing. To share $x \in Z_p$, a random number, $r_1 \in_R Z_p$ is selected and shares are set to $x^1 = r_1$ and $x^2 = x - r_1 \ mod \ p$. Addition, subtraction and multiplication of two secret signed values can be performed securely on corresponding shared values as long as the result of computation lies in $Z_{\langle l \rangle}$ [53]. The same argument holds for addition, subtraction and multiplication of a secret value by a public value (both from $Z_{\langle l \rangle}$) [53].

### B. APPLICATION I: PRIVACY-PRESERVING PROFILE-MATCHING

Social networks provide users with various fascinating features. Searching for new friends is one of the interesting features of social networks. When searching for a new friend, users naturally look for similarities in interests, activities, personality or even location. Therefore, they need to perform *profile-matching* to find out how much their interests match to other users' interests. However, there are serious concerns about privacy and efficiency of performing profile-matching in social networks. On the one hand, performing profile-matching requires the network to collect, maintain and process private personal information about users. Therefore, privacy-preserving profile-matching (PPM) solutions are necessary to deal with this issue. On the other hand, since users are mostly equipped with devices of limited resources, PPM solutions should be adequately efficient so that they do not cause users to store or communicate large amounts of data or to perform intensive computations.

Outsourcing storage and processing to cloud servers, is a promising solution to alleviate efficiency issues of profile-matching. However, it may introduce new privacy concerns. In this section, we describe the application of the proposed SSP protocol, $\pi_{SSP}$, in achieving an efficient cloud-assisted PPM solution. Profile-matching functionality is closely related to scalar product. In profile-matching, user profiles are typically quantized and represented as vectors, known as profile vectors (PVs), and the profile-matching functionality is modeled as computing scalar product of PVs [21].

#### 1) SYSTEM AND FUNCTIONALITY MODEL

Our assumptions and system model are similar to [21]. Fig. 5 depicts the system model where $n$ users ($U_j, j = 1, 2, \ldots, n$) and two cloud servers (CS1 and CS2) are assumed. Users can communicate with servers through secure channels while we do not assume communication between users. Cloud servers are also able to securely communicate with each other. We assume that each user has a personal profile which reflects personal interests of that user and can be quantized and represented by a vector $\overline{v} = (v_1, v_2, \ldots, v_k)$. The dimension $k$ is the number of attributes and is the same for all users. It is supposed that values of attributes are limited to an arbitrary

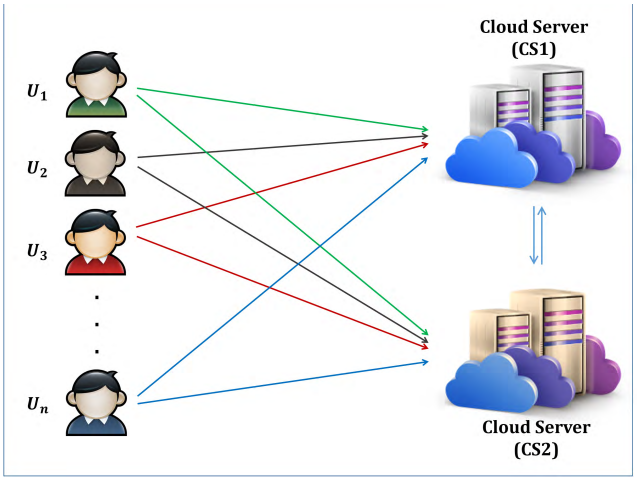


**FIGURE 5. Profile-matching with two cloud servers.**

range (*e.g.* [0, 10] in [21]) and are mapped to the set of $l$-bit signed integers, $Z_{\langle l \rangle}$ where the value of $l$ is selected properly to guarantee an arbitrary precision. Therefore, $v_i \in Z_{\langle l \rangle}$ for $i = 1, 2, \ldots, k$. In this model, the scalar product $\langle \overline{v} \cdot \overline{w} \rangle = \sum_{i=1}^{k} v_i w_i$ is regarded as the quantitative measure of matching for profile vectors $\overline{v}$ and $\overline{w}$. That is, profile vector $\overline{v}$ matches to profile vector $\overline{w}_1$ better than profile vector $\overline{w}_2$ iff $\langle \overline{v} \cdot \overline{w}_1 \rangle > \langle \overline{v} \cdot \overline{w}_2 \rangle$. We suppose that adversary controls only one of the cloud servers while there is no restriction on the number of users under control of the adversary. Therefore, the cloud servers will never collude with each other.

### 2) DESCRIPTION OF THE PROPOSED SCHEME

The proposed PPM scheme comprises of profile data uploading and query submission phases followed by profile-matching phase as represented in Fig. 6. In the profile data uploading phase, users secret share their PVs (after mapping them to a prime field $Z_p$ where $p > 2^l$). Users send the corresponding shares to CS1 and CS2 through secure channels. In the query submission phase, if $U_Q$ wishes to examine how much her profile is matched with $U_T$'s profile, sends a simple query message, $(query, U_Q, U_T)$, to CS1 and CS2. Upon receiving a query message $(query, U_Q, U_T)$, CS1 and CS2 obtain IDs $U_Q$ and $U_T$ from it, fetch corresponding shares of $\overline{v}_Q$ and $\overline{v}_T$, and run two instance of protocol $\pi_{SSP}$ to compute shares of $\langle \overline{v}_Q \cdot \overline{v}_T \rangle$. They send the resulted shares to $U_Q$.

The proposed scheme supports more complex queries. The case of batch profile-matching queries, where $U_Q$ requests servers to perform profile-matching with multiple users, will be easily handled if $U_Q$ includes the IDs of users in the query message. The case of partial profile-matching queries, where $U_Q$ requests the servers to compute profile-matching on a subset $\mathcal{S} \subseteq \{1, 2, \ldots, k\}$ of profile attributes, is also supported. In this case, servers run two instances of $\pi_{SSP}$ to compute shares of $z = \sum_{i \in \mathcal{S}} v_i w_i$ ($\overline{v}$ and $\overline{w}$ are the corresponding profile vectors) and send them to $U_Q$. Partial profile-matching allows profile-matching as fine as desired.

**Protocol $\pi_{PPM}$**

**Inputs:** $\overline{v} \in {Z_{\langle l \rangle}}^k$ by $U_j$ $(j = 1, 2, ..., n)$. $U_T$ by user $U_Q$.
**Result:** $U_Q$ learns $w_1$ and $w_2$ both in $Z_p$ where $w_1 + w_2 = \langle \overline{v}_Q \cdot \overline{v}_T \rangle$.

**Profile Data Uploading**

$U_j$:

- Selects $r_i \in Z_p$ for $i = 1, 2, ..., k$.
- Sets $\overline{v}^{CS1} = (r_1, r_2, , r_k)$ and $\overline{v}^{CS2} = (r'_1, r'_2, , r'_k)$, where $r'_i = v_i - r_i \ mod \ p$.
- Sends $(\overline{v}^{CS1}, U_j)$ and $(\overline{v}^{CS2}, U_j)$ to CS1 and CS2 respectively.

Upon receiving shares, cloud servers save them along with the corresponding user ID.

**Query Submission**

$U_Q$:

- Sends the query message $(\text{query}, U_Q, U_T)$ to CS1 and CS2.

**Profile-Matching**

CS1 and CS2:

- Fetch the information corresponding to $U_Q$ and $U_T$.
- Run $\pi_{SSP}$ to compute $z_1 = \langle \overline{v}_Q^{CS1} \cdot \overline{v}_T^{CS2} \rangle$ and receive $z_1^1$ and $z_1^2$ respectively.
- Run $\pi_{SSP}$ to compute $z_2 = \langle \overline{v}_Q^{CS2} \cdot \overline{v}_T^{CS1} \rangle$ and receive $z_2^1$ and $z_2^2$ respectively.

CS1:

- Computes its share as $w_1 = \langle \overline{v}_Q^{CS1} \cdot \overline{v}_T^{CS1} \rangle + z_1^1 + z_2^1$.
- Sends $w_1$ to $U_Q$.

CS2:

- Computes its share as $w_2 = \langle \overline{v}_Q^{CS2} \cdot \overline{v}_T^{CS2} \rangle + z_1^2 + z_2^2$.
- Sends $w_2$ to $U_Q$.

**FIGURE 6.** The proposed privacy-preserving profile-matching scheme ($\pi_{PPM}$).

### 3) CORRECTNESS

It is straightforward to verify that $z = w_1 + w_2 = \langle \overline{v}_Q \cdot \overline{v}_T \rangle$. Therefore, $U_Q$ receives the correct response to her query.

### 4) SECURITY

Under the trust assumption that cloud servers do not collude with each other, a server can learn no information about secret shared profile vectors since the employed secret sharing scheme is information theoretically secure. Note that the privacy of $U_j$'s profile information will be preserved even if the server colludes with an arbitrary number of users (obviously except $U_j$). A related observation is that, users also learn no information about uploaded data of other users even if they collude with one of the servers.

Privacy is also preserved during the query submission and profile-matching phases due to universally composable security of Protocol $\pi_{SSP}$. Particularly, privacy is preserved in the following cases:

- Against each of the cloud servers (which do not collude with each other)
- Against the requesting user, $U_Q$
- In the case that a cloud server colludes with the requesting user.

Security against cloud servers is trivial as they receive no messages containing private information except messages interacted in the two runs of $\pi_{SSP}$ which is UC secure. The requesting user, $U_Q$, receives two shares at the end of protocol that summation of them gives the expected scalar product $z = \langle \overline{v}_Q \cdot \overline{v}_T \rangle$. But this is not violating the privacy since $U_Q$ learns nothing more than what the functionality requires her to learn (*i.e.* $U_Q$ should receive the result).

The latter case is the worst case regarding security of query submission and profile-matching phases. In this case, the adversary controls $U_Q$ and one of the cloud servers and

**TABLE 2.** Security and functionality of privacy-preserving profile-matching protocols.

| PPM protocols | Security | | | Functionality | | |
|---|---|---|---|---|---|---|
| | Trust assumption | Set-up phase | Security level | Non-interactive for users | Batch profile-matching | Partial profile-matching |
| Proposed scheme ($\pi_{PPM}$) | Non-colluding servers | Not required | Information theoretic | ✓ | ✓ | ✓ |
| Scheme of [21] | Non-colluding servers | Required | Computational | ✓ | ✓ | X |

**TABLE 3.** Communications complexity of privacy-preserving profile-matching protocols.

| PPM protocols | Set-up phase communication | Data uploaded by a user (in bits) | Data uploaded per query (in bits) | Inter-cloud interactions (in bits) | Number of rounds (per matching) |
|---|---|---|---|---|---|
| Proposed scheme ($\pi_{PPM}$) | - | $2k\sigma$ | - | $4k\sigma$ | 1 |
| Scheme of [21] | $\sigma$ | $2(k+1)\sigma$ | $2(k+1)\sigma$ | $4(k+1)\sigma$ | 4 |

**TABLE 4.** Total storage and computation of privacy-preserving profile-matching protocols.

| PPM protocols | Required Memory (in bits) | | | Public-key operations (Per-query) | | |
|---|---|---|---|---|---|---|
| | User side | CS1 | CS2 | User side | CS1 | CS2 |
| Proposed scheme ($\pi_{PPM}$) | - | $kn\sigma$ | $kn\sigma$ | - | - | - |
| Scheme of [21] | $2\sigma$ | $(2k+3)n\sigma$ | $2\sigma$ | $3k+4$ MP, 0 BM | $2k+3$ MP, $2k+2$ BM | $4k+4$ MP, 1 BM |

Note: MP and BM stand for modular power and bilinear map operations respectively.

learns $z = \langle \bar{v}_Q \cdot \bar{v}_T \rangle$. Note that the goal of PPM functionality is to provide $U_Q$ with $z$. However, the adversary is not able to learn any information more than what $U_Q$ can solely learn observing $\bar{v}_Q$, and the result of scalar product, *i.e.* $z$. Therefore, the collusion is useless for adversary.

### 5) EFFICIENCY AND COMPARISON

In Table 2, 3 and 4 we present security aspects, functionality features and efficiency of the proposed PPM scheme. We also compare our scheme to the PPM scheme of [21] which is based on a system model identical to ours. In [21] the profile-matching functionality is modeled as computing scalar product and a cloud-assisted PPM scheme is proposed. The PPM scheme of [21] suggests outsourcing scalar product computations to two non-colluding cloud servers and utilizing the additive homomorphic proxy re-encryption of [54] to perform privacy preserving profile-matching.

As shown in Table 2, both schemes rely on the same trust assumption (*i.e.* access to two non-colluding cloud servers) while the proposed PPM scheme achieves information theoretic security. Our scheme also does not require any set-up phase or key management operation.

The proposed PPM scheme provides important features with regard to functionality and usability. Specifically, it avoids user-to-user communication, thus eliminating relevant complications. The proposed PPM scheme also supports more complex queries including batch profile-matching queries and partial profile-matching queries as explained previously. Batch queries allow a user to request multiple profiles to be processed for matching in a single query. On the other hand, partial matching queries allow a user to search for finer similarities in profile-matching process. The PPM scheme of [21] cannot support partial-matching unless users provide $O(2^k)$ extra information in data uploading phase.

In Table 3, we present the amount of communicated data in each phase of PPM protocols (for profile vectors with $k$ attributes and a filed with $\sigma$ bits representation). The proposed PPM scheme does not involve any set-up phase while in the PPM scheme of [21], each user interacts with one of the cloud servers (say CS2) to generate her re-encryption key and sends it to the other server (CS1). In data uploading phase, the proposed PPM scheme instructs users to send shares of their profile vectors to cloud servers where each share is of the same size as profile vectors (*i.e.* $k\sigma$ bits). In the PPM scheme of [21], a user encrypts her profile vector and an auxiliary value and sends them to CS1. Since the encryption scheme used in [21] doubles the size of cipher-text, user sends $2(k+1)\sigma$ bits in this phase. The proposed PPM scheme does not require any data transfer in the query submission phase as all data needed for profile-matching are the shares of profile vectors which have been uploaded to the cloud previously. In a single profile-matching process, the proposed PPM scheme involves two runs of Protocol $\pi_{SSP}$, hence transfer of $4k\sigma$ bits of data. This process involves only one communication round. The PPM scheme of [21] needs interaction of $4(k+1)\sigma$ bits in 4 communication rounds.

Table 4 reflects efficiency of the proposed PPM scheme in terms of storage and computation requirements. In the proposed PPM scheme, users do not need to store any data while each cloud server stores a share of profile vector per

user. Therefore, in a network of $n$ users, $kn\sigma$ bits of memory will be required on each server. In the PPM scheme of [21], users should securely save their private keys. One of the cloud servers (say CS1) needs to store encrypted profile vectors and re-encryption keys which require $(2k + 3)n\sigma$ bits together. The other cloud server only needs to store its private key.

The proposed PPM scheme completely avoids public key operations while the PPM scheme of [21] imposes intensive computations not only on cloud servers but also on users. Per submitted query, the requesting user, encrypts her profile vector and decrypts the result which require $O(k)$ modular power operations. The computation complexity of cloud servers is not balanced. One of the servers which acts as a proxy, CS1 in Table 4, computes $O(k)$ modular power operations plus $O(k)$ bilinear map operations per query. The other server computes $O(k)$ modular power operations and $O(1)$ bilinear map operations per query. Note that, computation complexity is critical for cloud servers, since the rate of query submissions is proportional to the size of network thus typically high.

### C. APPLICATION II: SECURE REMOTE HEALTH MONITORING

In remote health monitoring, the process of analyzing health data with respect to pre-defined criteria is outsourced to cloud servers. Users upload their own health data to cloud where dedicated servers analyze the data and send results to medical service provider (MSP) which will do proper actions correspondingly. Due to the high availability of cloud servers, users can upload their health data to cloud in regular time intervals to let their health be continuously monitored. From security perspective, cloud servers should not learn information about users health condition.

In this section, we present a secure remote health monitoring (SRHM) scheme based on the proposed SSP protocol, $\pi_{SSP}$. The proposed SRHM scheme is unconditionally secure and highly efficient.

#### 1) SYSTEM AND FUNCTIONALITY MODEL

Fig. 7 illustrates a simplified setting of the system. We assume a network of $n$ users, $(U_j, j = 1, 2, \ldots, n)$, a medical service provider (MSP) and two cloud servers (CS1 and CS2). Each user is typically equipped with multiple sensors which measure various health factors of her. Sensors send the measured data to a central device which is responsible for arranging health data in a predefined appropriate format and sending them to the cloud. This device is securely connected to the cloud servers. Therefore, the assumptions about communication in the system is as follows. Users are connected to cloud servers through secure channels. The connection between MSP and each cloud server is secure too. Furthermore, the cloud servers are able to securely communicate with each other. There is no need for user-to-user or user-to-MSP communications.

We assume that health data of user $U_j$ are arranged in the form of the so called health data vector (HDV), $\overline{h}_j = (h_1, h_2, \ldots, h_k)$. The dimension $k$ is the number of factors measured by sensors and is the same for all users. It is supposed that values of factors are limited to an arbitrary range and are mapped to the set of $l$-bit signed integers, $Z_{\langle l \rangle}$. The value of $l$ is selected properly to guarantee an arbitrary precision. Therefore, $h_i \in Z_{\langle l \rangle}$ for $i = 1, 2, \ldots, k$.

To enable secure remote health monitoring, given $j$, $d$, $\theta_1$ and $\theta_2$, the MSP should be able to find out whether the $i$th element of $\overline{h}_j$, *i.e.* $h_d$, lies in the valid range $[\theta_1, \theta_2]$ or not (while the cloud servers learn no information about $\overline{h}_j$, $\theta_1$ or $\theta_2$). This is equivalent to providing the MSP with the ability of securely checking both the inequalities $h_d \geq \theta_1$ and $h_d \leq \theta_2$. We model this functionality as computing $z = h_d - \theta_1$ and $z = \theta - h_d$ and observing the sign of $z$. Specifically, the MSP inputs the tuple $(j, d, \alpha, op)$ and receives $z = h_d - \alpha$ if $op =$ "*Lower*" or $z = \alpha - h_d$ if $op =$ "*Upper*" while the servers do not learn $d$, $\alpha$ or $op$. If $z < 0$, the MSP should act properly.

We suppose that adversary controls only one of the cloud servers while there is no restriction on the number of users under control of the adversary. Therefore, we assume that the cloud servers will never collude with each other.

#### 2) DESCRIPTION OF THE PROPOSED SCHEME

Assuming the system setting of Fig. 7, the protocol of Fig. 8 realizes remote health monitoring functionality. Suppose that $\overline{h}_j = (h_1, h_2, \ldots, h_k)$ is the health data vector of user $U_j$ where $h_i \in Z_{\langle l \rangle}$ are $l$-bit signed integers. To securely upload the health data vector to the cloud, $U_j$ constructs extended data vector $\overline{h}_j^e = (h_1, h_2, \ldots, h_k, 1)$. Then, $U_j$ maps $\overline{h}_j^e$ to $\overline{h}_j^* = (h_1 \ mod \ p, \ h_2 \ mod \ p, \ \ldots, \ h_k \ mod \ p, \ 1)$ and secret shares $\overline{h}_j^*$ element by element using additive secret sharing scheme of Section V-A. Specifically, $U_j$ selects $k+1$ random values, $r_i \in_R Z_p$, and constructs shares $\overline{h} - j^{CS1} = (r_1, r_2, \ldots r_{k+1})$ and $\overline{h}_j^{CS2} = (r'_1, r'_2, \ldots r'_{k+1})$ where $r'_i = h_i - r_i \ mod \ p$. $U_j$ sends each share (along with her ID) to the corresponding cloud server. Cloud servers save the received shares and IDs.

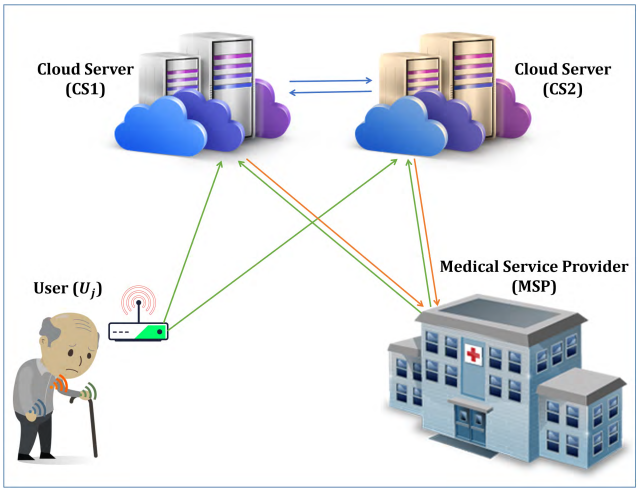


**FIGURE 7.** Remote health monitoring with two cloud servers.

**Protocol $\pi_{SRHM}$**

**Inputs:** $\overline{h}_j \in {Z_{\langle l \rangle}}^k$ by $U_j$ $(j = 1, 2, ..., n)$ and $\alpha \in Z_{\langle l \rangle}$, $d \in \{1, ..., k\}$ and $op \in \{`Upper', `Lower'\}$ by the MSP.
**Result:** CS1 and CS2 learn $w_1$ and $w_2$ both in $Z_p$ where $w_1 + w_2 = h_d - \alpha \ mod \ p$ if $op = `Lower'$ and $w_1 + w_2 = \alpha - h_d \ mod \ p$ if $op = `Upper'$.

**Health Data Uploading**

$U_j$:

- Constructs extended HDV as $\overline{h}_j^e = (h_1, h_2, , h_k, 1)$.
- Selects $r_i \in Z_p$ for $1 \leq i \leq k+1$ and sets $\overline{h}_j^{CS1} = (r_1, r_2, , r_{k+1})$ and $\overline{h}_j^{CS2} = (r'_1, r'_2, , r'_{k+1})$, where $r'_i = h_i^e - r_i \ mod \ p$.
- Sends $(\overline{h}_j^{CS1}, U_j)$ and $(\overline{h}_j^{CS2}, U_j)$ to CS1 and CS2 respectively.

Upon receiving shares, cloud servers save them along with the corresponding user ID.

**Query Submission**

MSP:

- Constructs the query vector $\overline{y}$ as follows:

$$\overline{y} = \begin{cases} (0, ..., 0, -1, 0, ..., 0, \alpha), & if \ op = `Upper' \\ (0, ..., 0, 1, 0, ..., 0, -\alpha), & if \ op = `Lower' \end{cases}$$

  where the $d$th element is set to $'+1'$ or $'-1'$ correspondingly.
- Selects $\rho_i \in Z_p$ for $1 \leq i \leq k+1$ and sets $\overline{y}^{CS1} = (\rho_1, \rho_2, , \rho_{k+1})$ and $\overline{y}^{CS2} = (\rho'_1, \rho'_2, , \rho'_{k+1})$, where $\rho'_i = h_i - \rho_i \ mod \ p$.
- Sends $(\overline{y}^{CS1}, U_j)$ and $(\overline{y}^{CS2}, U_j)$ to CS1 and CS2 respectively.

**Health Data Processing**

CS1 and CS2:

- Find the information corresponding to $U_j$.
- Run $\pi_{SSP}$ to compute $z_1 = \langle \overline{h}_j^{CS1} \cdot \overline{y}^{CS2} \rangle$ and receive $z_1^1$ and $z_1^2$ respectively.
- Run $\pi_{SSP}$ to compute $z_2 = \langle \overline{h}_j^{CS2} \cdot \overline{y}^{CS1} \rangle$ and receive $z_2^1$ and $z_2^2$ respectively.
- Compute their shares as $w_1 = \langle \overline{h}_j^{CS1} \cdot \overline{y}^{CS1} \rangle + z_1^1 + z_2^1$ and $w_2 = \langle \overline{h}_j^{CS2} \cdot \overline{y}^{CS2} \rangle + z_1^2 + z_2^2$.

**FIGURE 8.** The proposed secure remote health monitoring scheme ($\pi_{SRHM}$).

For each element of HDV, the MSP needs to check if it is in a specific healthy range. Therefore, the MSP needs comparison queries (both '$\geq$' and '$\leq$') which compare elements of HDV with secret thresholds. To compare the $d$th element of HDV to a threshold $\alpha$, the MSP constructs the query vector $\overline{y}$ as follows:

$$\overline{y} = \begin{cases} (0, \ldots, 0, -1, 0, \ldots, 0, \alpha), & \textit{if } \alpha \textit{ is upper bound} \\ (0, \ldots, 0, 1, 0, \ldots, 0, -\alpha), & \textit{if } \alpha \textit{ is lower bound} \end{cases}$$

where the $d$th element of $\overline{y}$ is filled with '1' or '-1' correspondingly. Then, the MSP maps $\overline{y}$ to $Z_p$ and achieves $\overline{y}^*$, secret shares it as $\overline{y}^{CS1}$ and $\overline{y}^{CS2}$ and sends each share (along with the ID of the queried user) to the corresponding cloud server.

The cloud servers run two instances of protocol $\pi_{SSP}$ on shares of $\overline{h}^*$ and $\overline{y}^*$ to compute $z = \langle \overline{h}^* \cdot \overline{y}^* \rangle$. Each server achieves a share of $z$. They can either send the shares to MSP or reconstruct $z$ and inform MSP if $z < 0$.

### 3) CORRECTNESS

Protocol $\pi_{SRHM}$ is correct because

$$\begin{aligned} z = w_1 + w_2 &= \langle (\overline{h}^{CS1} + \overline{h}^{CS2}) \cdot (\overline{y}^{CS1} + \overline{y}^{CS2}) \rangle \\ &= \langle \overline{h}^* \cdot \overline{y}^* \rangle \\ &= \begin{cases} \alpha - h_d \ mod \ p, & if \ op = `Upper' \\ h_d - \alpha \ mod \ p, & if \ op = `Lower' \end{cases} \end{aligned}$$

Therefore, $z < 0$ iff $h_d$ violates queried threshold, $\alpha$.

### 4) SECURITY

We discuss two cases regarding the action of the cloud servers at the last step. The first case is when the servers send the final shares to MSP. Obviously, in this case, Protocol $\pi_{SRHM}$ is secure under the assumption that cloud servers do not collude. The intuition is that Protocol $\pi_{SRHM}$ only uses Protocol $\pi_{SSP}$ which is UC secure. Furthermore, CS1 and CS2 only see additively hidden values during the protocol execution.

The other case is when the cloud servers reconstruct $z$ and inform MSP if $z < 0$. In this case, the cloud servers only learn the value of $z = h_d - \alpha$ (or $z = \alpha - h_d$) while the query type, $d$, $\alpha$ and $h_i$ for $1 \leq i \leq k$ are perfectly hidden. The amount of information leakage due to learning $z$ by servers is arguable. As it is needed to process repeated queries on health data of different users, a curious server can run various analyses on the set of results. However, notice that using appropriate preprocessing on data and query (*e.g.* proper offset and scaling), it is possible to transform all values to a specific range to make the distribution of $z$ uniform. This increases ambiguity since adversary (*i.e.* curious cloud server) cannot distinguish the results of different queries.

### 5) EFFICIENCY AND COMPARISON

In Table 5, security and functionality of the proposed SRHM scheme is compared to the scheme of [55]. In [55] Dong *et al.* have proposed a secure continuous remote health monitoring (SCRHM) system based on asymmetric scalar-product-preserving encryption (ASPE) scheme of [56]. In SCRHM system of [55], each user encrypts her health data vector (HDV) using ASPE scheme with the secret key that she have shared with the MSP in the setup phase. Then, the user uploads encrypted data vector to the cloud server. For each element of the encrypted data vector, cloud server can check arbitrary range queries provided that the value of query is properly encrypted using ASPE scheme (with the same key) and sent to the could server. The cloud server can learn neither

**TABLE 5.** Security and functionality of secure remote health monitoring protocols.

| SRHM protocols | Security | | | Functionality | | |
|---|---|---|---|---|---|---|
| | pre-shared secret key | Trust assumption | Security level | Non-interactive for users | Continuous monitoring | Batch query |
| Proposed scheme ($\pi_{SRHM}$) | Not required | Non-colluding servers | Information theoretic | ✓ | ✓ | ✓ |
| Scheme of [55] | Required | Secret key | Computational | ✓ | ✓ | X |

**TABLE 6.** Communications complexity of secure remote health monitoring protocols.

| SRHM protocols | Data uploaded by a user (in bits) | Data uploaded per query (in bits) | Inter-cloud interactions (in bits) |
|---|---|---|---|
| Proposed scheme ($\pi_{SRHM}$) | $2(k+1)\sigma$ | $2(k+1)\sigma$ | $4(k+1)\sigma+2\sigma$ |
| Scheme of [55] | $2(k+1)\sigma$ | $4(k+1)\sigma$ | - |

**TABLE 7.** Query and storage complexity of secure remote health monitoring protocols.

| SRHM protocols | Total queried data (in bits) | Required memory (in bits) | | |
|---|---|---|---|---|
| | | User side | MSP side | On a cloud server |
| Proposed scheme ($\pi_{SRHM}$) | $2k(k+1)\sigma$ | - | - | $(k+1)(k+n)\sigma$ |
| Scheme of [55] | $4k(k+1)n\sigma$ | $(k+1)(k+2)\sigma$ | $(k+1)(k+2)n\sigma$ | $4k(k+1)n\sigma$ |

the HDV nor the queried range while it can find out whether the element lies in the queried range or not.

The proposed scheme is based on two non-colluding servers while the scheme of [55] is based on secure pre-shared keys. Therefore, an advantage of the proposed scheme is that users and the MSP do not need to deal with the issues of sharing and management of secret keys. The other advantage of the proposed scheme is that it achieves unconditional security since the underlying SSP protocol ($\pi_{SSP}$) is unconditionally UC secure. On the other hand, the ASPE encryption used in [55] limits the security to computational.

Both schemes have the advantage of not needing user-to-user or user-to-MSP communication. Furthermore, both schemes support continuous monitoring where users continuously collect and send their health data to cloud where dedicated servers run predefined queries on the received data and inform the MSP of anomalies. Additionally, The proposed SRHM scheme provides the batch query feature. That is, a single query can be applied on health vectors of all users or a bunch of users. This feature has a direct impact on efficiency of the scheme.

Communications complexity of the proposed SRHM scheme is compared to the scheme of [55] in Table 6. In this table, the communicated bits at each step of the schemes is presented for a single user and a single query. The complexity of uploading health data is the same in both schemes while the complexity of a query in our scheme is less than the scheme of [55] by a factor of 2. The proposed scheme requires communication between cloud servers while the scheme of [55] is based on a single server.

Table 7 reflects efficiency improvements of the proposed SRHM scheme in a full practical scenario. For a remote health monitoring system with $n$ users, the amount of communicated data to query all elements of all HDVs is reduced by a factor of $n$. This improvement is due to the batch query feature of the proposed scheme. Efficiency improvement in terms of data storage is also considerable. In the proposed scheme, users and MSP do not need to save and manage secret keys. This is a considerable advantage especially in the MSP side. Notice that in the scheme of [55], the MSP needs to store and manage $n$ secure keys which require $O(nk^2\sigma)$ bits of memory. Furthermore, in the proposed SRHM scheme, cloud servers need $O(nk\sigma)$ bits of memory to save their corresponding shares of HDVs while in the scheme of [55] the cloud server needs $O(nk^2\sigma)$ bits of memory.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a secure two-party protocol for computing split scalar product of private vectors. The proposed SSP protocol works in the preprocessing model. In this model, parties are provided with proper sets of correlated randomness before deciding their inputs and thus before starting the computation. Owing to this model, our protocol is super efficient and unconditionally secure. This protocol is also secure in the universal composability framework. Therefore, its security is guaranteed when being used along with other UC protocols. The proposed SSP protocol is applicable in various secure computation scenarios as well as MPC general constructions (as a sub-protocol for distributed multiplication in the latter case).

We also showed that our SSP protocol is optimal in terms of the size of pre-distributed randomness as well as the size of communicated messages. In order to show the optimality of our protocol, we derived lower bounds on the size of pre-distributed randomness and the size of communicated messages in unconditionally secure two-party computations in the preprocessing model. The lower bounds derived in this paper are restricted to a class of two-party functionalities. Our results can be regarded as a step forward toward understanding the limits and capabilities of the preprocessing model. It would be valuable and also quite challenging to study the possibility of extending these bounds to the general case of two-party unconditionally secure computation in the preprocessing model.

Cloud computing scenarios typically require computation on private data and therefore can be directly mapped to secure computation scenarios. We presented two applications of the proposed SSP protocol in secure cloud computing. Particularly, we proposed a cloud-assisted privacy-preserving profile-matching scheme and a secure remote health monitoring scheme based on our SSP protocol. We also discussed security, functionality features and efficiency of the proposed schemes and demonstrated that both of the proposed schemes are secure and highly efficient.

## REFERENCES

[1] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci.*, Nov. 1982, pp. 160–164.
[2] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proc. 19th Annu. ACM Symp. Theory Comput.*, Jan. 1987, pp. 218–229.
[3] D. Chaum, I. Damgård, and J. van de Graaf, "Multiparty computations ensuring privacy of each party's input and correctness of the result," in *Advances in Cryptology—CRYPTO*. New York, NY, USA: Springer, 2006, pp. 87–119.
[4] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proc. 10th Annu. ACM Symp. Theory Comput.*, Jan. 1988, pp. 1–10.
[5] D. Chaum and C. Crépeau, and I. Damgard, "Multiparty unconditionally secure protocols," in *Proc. 9th Annu. ACM Symp. Theory Comput.*, Jan. 1988, pp. 11–19.
[6] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proc. 21st Annu. ACM Symp. Theory Comput.*, Feb. 1989, pp. 73–85.
[7] J. Baron, K. E. Defrawy, K. Minkovich, R. Ostrovsky, and E. Tressler, "5pm: Secure pattern matching," in *Security and Cryptography for Networks*. New york, NY, USA: Springer, 2012, pp. 222–240.
[8] L. Kissner and D. Song, "Privacy-preserving set operations," in *Advances in Cryptology—CRYPTO*. New York, NY, USA: Springer, 2005, pp. 241–257.
[9] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations Newslett.*, vol. 4, no. 2, pp. 28–34, Dec. 2002.
[10] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining," in *Information Security and Cryptology—ICISC*. New York, NY, USA: Springer, 2004, pp. 104–120.
[11] W. Du and M. J. Atallah, "Privacy-preserving cooperative scientific computations," in *Proc. CSFW*, vol. 1, Jun. 2001, pp. 273–239.
[12] W. Du and M. J. Atallah, "Privacy-preserving cooperative statistical analysis," in *Proc. 17th Annu. Comput. Secur. Appl. Conf.*, Dec. 2001, pp. 102–110.
[13] O. Catrina and S. De Hoogh, "Secure multiparty linear programming using fixed-point arithmetic," in *Computer Security—ESORICS*. New York, NY, USA: Springer, 2010, pp. 134–150.
[14] J. Dreier and F. Kerschbaum, "Practical privacy-preserving multiparty linear programming based on problem transformation," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust*, Oct. 2011, pp. 916–924.
[15] F. Kerschbaum *et al.*, "Secure collaborative supply-chain management," *IEEE Comput.*, vol. 44, no. 9, pp. 38–43, Sep. 2011. doi: 10.1109/MC.2011.224.
[16] R. Pibernik, Y. Zhang, F. Kerschbaum, and A. Schröpfer, "Secure collaborative supply chain planning and inverse optimization The JELS model," *Eur. J. Oper. Res.*, vol. 208, no. 1, pp. 75–85, Jan. 2011.
[17] J.-G. Dumas, P. Lafourcade, J.-B. Orfila, and M. Puys, "Dual protocols for private multi-party matrix multiplication and trust computations," *Comput. Secur.*, vol. 71, pp. 51–70, Nov. 2017.
[18] F. Kerschbaum, "Secure and sustainable benchmarking in clouds," *Bus. Inf. Syst. Eng.*, vol. 3, no. 3, pp. 135–143, Jun. 2011.
[19] A. Schroepfer, A. Schaad, F. Kerschbaum, H. Boehm, and J. Jooss, "Secure benchmarking in the cloud," in *Proc. 18th ACM Symp. Access Control Models Technol.*, Jun. 2013, pp. 197–200.
[20] P. Li *et al.*, "Multi-key privacy-preserving deep learning in cloud computing," *Future Generat. Comput. Syst.* vol. 74, pp. 76–85, Sep. 2017.
[21] C.-Z. Gao, Q. Cheng, X. Li, and S.-B. Xia, "Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network," *Cluster Comput.*, vol. 5, pp. 1–9, Feb. 2018.
[22] B. David, R. Dowsley, J. van de Graaf, D. Marques, A. C. Nascimento, and A. C. Pinto, "Unconditionally secure, universally composable privacy preserving linear algebra," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 59–73, Jan. 2016.
[23] D. Beaver, "Commodity-based cryptography," in *Proc. 19th Annu. ACM Symp. Theory Comput.*, May 1997, pp. 446–455.
[24] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias, "Semi-homomorphic encryption and multiparty computation," in *Advances in Cryptology—EUROCRYPT*. New York, NY, USA: Springer, 2011, pp. 169–188.
[25] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology—CRYPTO*. New York, NY, USA: Springer, 2012, pp. 643–662.
[26] I. Damgård and S. Zakarias, "Constant-overhead secure computation of Boolean circuits using preprocessing," in *Theory Cryptography*. New York, NY, USA: Springer, 2013, pp. 621–641.
[27] E. Larraia, E. Orsini, and N. P. Smart, "Dishonest majority multi-party computation for binary circuits," in *Advances in Cryptology—CRYPTO*. New York, NY, USA: Springer, 2014, pp. 495–512.
[28] R. Rivest, "Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer," Tech. Rep., 1999.
[29] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass, "Universally composable protocols with relaxed set-up assumptions," in *Proc. 45th Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2004, pp. 186–195.
[30] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. 42nd IEEE Symp. Found. Comput. Sci.*, 2001, pp. 136–145.
[31] R. Canetti, "Security and composition of cryptographic protocols: A tutorial (part i)," *ACM SIGACT News*, vol. 37, no. 3, pp. 67–92, 2006.
[32] V. Goyal and J. Katz, "Universally composable multi-party computation with an unreliable common reference string," in *Theory Cryptography*. New York, NY, USA: Springer, 2008, pp. 142–154.
[33] R. Canetti, E. Kushilevitz, and Y. Lindell, "On the limitations of universally composable two-party computation without set-up assumptions," *J. Cryptol.*, vol. 19, no. 2, pp. 135–167, Jun. 2006.
[34] R. Tonicelli *et al.*, "Information-theoretically secure oblivious polynomial evaluation in the commodity-based model," *Int. J. Inf. Secur.*, vol. 14, no. 1, pp. 73–84, 2015.
[35] R. Dowsley, J. Van De Graaf, D. Marques, and A. C. Nascimento, "A two-party protocol with trusted initializer for computing the inner product," in *Information Security Applications*. New York, NY, USA: Springer, 2010, pp. 337–350.
[36] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Feb. 2002, pp. 639–644.
[37] Y. Zhu and T. Takagi, "Efficient scalar product protocol and its privacy–preserving application," *Int. J. Electron. Secur. Digit. Forensics*, vol. 7, no. 1, pp. 1–19, 2015.
[38] Y. Zhu, T. Takagi, and L. Huang, "Efficient secure primitive for privacy preserving distributed computations," in *Advances in Information and Computer Security*. New York, NY, USA: Springer, 2012, pp. 233–243.

[39] Y. Zhu, Z. Wang, B. Hassan, Y. Zhang, J. Wang, and C. Qian, "Fast secure scalar product protocol with (almost) optimal efficiency," in *Collaborative Computing: Networking, Applications, and Worksharing*. New York, NY, USA: Springer, 2015, pp. 234–242.
[40] A. Amirbekyan and V. Estivill-Castro, "A new efficient privacy-preserving scalar product protocol," in *Proc. 6th Australasian Conf. Data Mining Analytics*, Dec. 2007, pp. 209–214.
[41] C. Dong and L. Chen, "A fast secure dot product protocol with application to privacy preserving association rule mining," in *Advances in Knowledge Discovery and Data Mining*. New York, NY, USA: Springer, 2014, pp. 606–617.
[42] M. Shaneck and Y. Kim, "Efficient cryptographic primitives for private data mining," in *Proc. 43rd Hawaii Int. Conf. Syst. Sci.*, Jan. 2010, pp. 1–9.
[43] W. Du and Z. Zhan, "A practical approach to solve secure multi-party computation problems," in *Proc. Workshop New Secur. Paradigms*, Sep. 2002, pp. 127–135.
[44] M. D. Cock, R. Dowsley, A. C. Nascimento, and S. C. Newman, "Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data," in *Proc. 8th ACM Workshop Artif. Intell. Secur.*, Oct. 2015, pp. 3–14.
[45] B. Siabi and M. Berenjkoub. (2018). "On the security of an unconditionally secure, universally composable inner product protocol." [Online]. Available: https://arxiv.org/abs/1809.08441
[46] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proc. Annu. Int. Cryptol. Conf.*, 1991, pp. 420–432.
[47] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky, "On the power of correlated randomness in secure computation," in *Theory Cryptography*. New York, NY, USA: Springer, 2013, pp. 600–620.
[48] O. Goldreich, *Foundations of Cryptography: Basic Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[49] C. Hazay and Y. Lindell, *Efficient Secure Two-Party Protocols: Technology Constructions*. New York, NY, USA: Springer, 2010.
[50] G. Asharov and Y. Lindell, "A full proof of the BGW protocol for perfectly secure multiparty computation," *J. Cryptol.*, vol. 30, no. 1, pp. 58–151, Aug. 2017.
[51] E. Kushilevitz, Y. Lindell, and T. Rabin, "Information-theoretically secure protocols and security under composition," *SIAM J. Comput.*, vol. 39, no. 5, pp. 2090–2112, Sep. 2010.
[52] C. Crëpeau, G. Savvides, C. Schaffner, and J. Wullschleger, "Information-theoretic conditions for two-party secure function evaluation," in *Advances in Cryptology—EUROCRYPT*. New York, NY, USA: Springer, 2006, pp. 538–554.
[53] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *Proc. Int. Conf. Financial Cryptography Data Security*. May 2010, pp. 35–50.
[54] B. K. Samanthula, Y. Elmehdwi, G. Howser, and S. Madria, "A secure data sharing and query processing framework via federation of cloud computing," *Inf. Syst.*, vol. 48, pp. 196–212, Mar. 2015.
[55] Q. Dong, Z. Guan, K. Gao, and Z. Chen, "Scrhm: A secure continuous remote health monitoring system," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 12, 2015, Art. no. 392439.
[56] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure KNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, May 2009, pp. 139–152.

**BABAK SIABI** received the B.Sc. degree in electrical engineering from the Iran University of Science and Technology and the M.Sc. degree in electrical engineering from the Isfahan University of Technology, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His main research interest includes secure computation, with a focus on efficiency enhancement of critical subprotocols and primitives.

**MEHDI BERENJKOUB** received the Ph.D. degree from the Department of Electrical and Computer Engineering, Isfahan University of Technology (IUT), in 2000. The title of his dissertation is two-party key distribution protocols in cryptography. He started his career as an Assistant Professor with the Department of Electrical and Computer Engineering, Isfahan University of Technology, where he taught fundamentals of cryptography, cryptographic protocols, network security, and intrusion detection. He has supervised dozens of M.Sc. students and Ph.D. candidates in related areas. He established a research group on Security in Networks and Systems in IUT along with his colleagues. He is also responsible for an established academic CSIRT, IUT, where he is currently an Associate Professor. His current research interests include advanced security protocols, wireless network security, authentication protocols, and intrusion detection systems. He was also one of the founder members of the Iranian Society of Cryptology, in 2001. He has continued his cooperation with the society as an active member.

**WILLY SUSILO** received the Ph.D. degree in computer science from the University of Wollongong, Australia. He is currently a Professor and the Head of the School of Computing and Information Technology and the Director of the Institute of Cybersecurity and Cryptology (iC2), University of Wollongong. He has published more than 500 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. His work has been cited more than 13 000 times in Google Scholar. He was previously awarded the prestigious ARC Future Fellow by the Australian Research Council (ARC) and the Researcher of the Year Award by the University of Wollongong, in 2016. He has served as a Program Committee Member in dozens of international conferences. He is currently serving as an Associate Editor in several international journals, including the IEEE Transactions in Dependable and Secure Computing, *Computer Standards and Interface* (Elsevier), and the *International Journal of Information Security* (Springer). He is the Editor-in-Chief of *Information* journal.