Impossibility of unconditionally secure scalar products

Thomas B. Pedersen *, Erkay Savaş

Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey

A B S T R A C T

The ability to perform scalar products of two vectors, each known to a different party, is a central problem in privacy preserving data mining and other multi-party computation problems. Ongoing search for both efficient and secure scalar product protocols has revealed that this task is not easy. In this paper we show that, indeed, scalar products can never be made secure in the information theoretical sense. We show that any attempt to make unconditionally secure scalar products will inevitably allow one of the parties to learn the other parties input vector with high probability. On the other hand, we show that under various assumptions, such as the existence of a trusted third party or the difficulty of discrete logarithms, both efficient and secure scalar products do exist. We proposed two new protocols for secure scalar products and compare their performance with existing secure scalar products.

Keywords:
Security and privacy
Data mining
Scalar products

1. Introduction

For almost three decades the cryptographic community has known that any distributed algorithm we may think of can, at least in theory, be solved securely with standard multi-party computation techniques [33,16,4]. However, the communication and computation cost of these standard techniques are too high for most commercial applications. Recently, researchers in areas such as privacy preserving data mining have proposed special purpose protocols to overcome these inefficiency issues. The solutions presented fall in two categories: solutions which perturbs the data in ways which preserves the statistical properties of interest while hiding sensitive information (first introduced in [1], see [31] for a more recent example), and solution which rely on cryptographic primitives. In this paper we focus on cryptographic approaches approach.

A quick survey of the research in data mining will soon reveal that some problems occur again and again; as for example secure two-party computation of scalar products. Scalar products are useful in a wide range of modern applications of secure computation such as privacy preserving data mining, scientific computing, and web personalisation. Indeed, Kargupta and co-workers [9], motivated by the importance of this problem in data mining, propose an efficient algorithm for estimating the top-l scalar products between distributed vectors. Also motivated by the usefulness of scalar products, but this time from a security perspective, Goethals et al. [14] give a secure implementation of scalar products between distributed vectors.

There are many concrete examples of the use of scalar products in data mining and machine learning. Support vector machines and page ranking, for instance, directly uses scalar products. Yang and Wright [32,30] use a secure scalar product protocol in privacy preserving computation of Bayesian networks. In [2] the authors use an (insecure) scalar product to compute correlation coefficients between random variables held by different parties, and to compute linear regression models of distributed data. Clifton and Vaidya show how to securely perform the apriori algorithm used in association rule mining by using secure scalar products [28]. In [29] the same authors show how to combine state of the art query processing algorithms with cryptographic techniques to securely compute kth element score. Their algorithm relies on secure set intersection,

* Corresponding author.
E-mail addresses: pedersen@sabanciuniv.edu (T.B. Pedersen), erkays@sabanciuniv.edu (E. Savaş).

which can efficiently be computed with scalar products [13]. In [27] Vaidya show how to perform privacy preserving linear programming. A central element of his approach is the secure scalar product protocol from [14].

The standard ways to compute scalar products are to use homomorphic encryption, secret sharing, or solutions based on oblivious transfer [14]. However, some researchers have expressed concern that these techniques give too much communication and computation overhead. Instead of using the standard techniques these authors propose ad. hoc. protocols for secure scalar products [2,28,17,24,20].

Unfortunately, many of the proposed scalar products are insecure. Goethals et al. [14] demonstrate attacks on two of the proposed protocols [28,2]. In this paper we show that *no unconditionally secure scalar product protocol exists* – even when the adversary is semi-honest. Our result strengthen the claim of Goethals et al. [14] that: "In almost all solutions [which do not rely on extra assumptions], one can construct a system of linear equations based on the specification of the protocol, and solve it for the secret values." As a consequence, extra assumptions, such as the difficulty of factoring, are needed to construct secure scalar products.

It is no big surprise that unconditionally secure scalar products are impossible. The special case of computing the scalar product of binary vectors can be used compute Boolean *and* gates. However, unconditionally secure computation of *and* gates is impossible [5]. To investigate the possibility of scalar products in fields larger than the Boolean field, we turn to the well-known fact that *commitment schemes* cannot be unconditionally secure. We prove the impossibility of unconditionally secure scalar products by showing a reduction from commitment schemes to scalar products. More than that, we show that, without extra assumptions, any two-party scalar product protocol between Alice and Bob has an n such that Alice learns approximately n scalar products, while Bob learns Alice's vector with probability $1/n$. However, the reduction also suggests that secure scalar products exist in alternative models where commitment schemes have been demonstrated.

Goethals et al. [14], and Wright and Yang [30,32] propose computationally secure scalar product protocols. In many cases these scalar product protocols have less communication overhead than the protocols proven insecure in [14]. This is in contrast to the initial motivation of the authors of the previous scalar product protocols; namely that standard cryptographic techniques were suspected to give protocols which are too inefficient for practical purposes. In this paper, we further reduce the communication overhead of the scalar product protocol from [14] in the case of computing scalar products over small fields.

It is common in the privacy preserving data mining literature to assume that the parties involved in the protocol are semi-honest and non-colluding (e.g. [28,17,2]). We propose a new scalar product protocol which is secure in this model, and has a communication overhead of only $O(n^{1.6})$ when computing scalar products of vectors of n-bit integers. The hidden constant is very small, and the resulting protocol is efficient enough to be of interest in practical protocols.

Our contributions are: (1) the impossibility of scalar products, secure against semi-honest adversaries without extra assumptions, (2) a lower bound on the information leaked in any scalar product protocol when the adversary is computationally unbounded, and (3) two efficient scalar product protocols which are secure in two different models.

This paper is organised as follows: In Section 2, we give a brief discussion of the model we work in. In Section 2.3, we show an attack on commitment schemes, when no extra assumptions are made. Our first main result is the reduction of commitment schemes to scalar products in Section 3 which proves the impossibility of unconditionally secure scalar products, and in Section 4, we apply our bound to the scalar product from [20]. In Section 5, we give two efficient and secure scalar product protocols, the first is an improvement of the protocol from [14], the second is a new scalar product protocol. We show test results supporting our claim that our proposed scalar products perform considerably better than previously proposed scalar product protocols. We give some concluding remarks in Section 6.

2. Preliminaries

When implementing cryptologic protocols it is important to be precise about the model which guarantees security of the protocol. The contribution of this paper is to show that scalar products cannot be securely computed by two parties when no extra assumptions are made, but that efficient *and* secure implementations exists in other models.

2.1. Security models

In our setting, two parties, Alice and Bob, try to compute a function without help from any external party. They both have unlimited time and space available for their computations. We prove the impossibility of scalar products even for semi-honest parties. That is: they do exactly as the protocol is describing, but collect information during the execution of the protocol, which they use to gain information about the other parties inputs. Limiting our attention to semi-honest parties is not a restriction, since this gives the strongest statement: *No scalar product can be secure against semi-honest adversaries if no extra assumptions are made*. The impossibility results automatically apply to adversaries who actively try to cheat, since any protocol which is secure against active adversaries is also secure against semi-honest adversaries.

In the following we give an informal overview of the cryptographic definition of secure protocols. For more details see [15]. Suppose that Alice and Bob have private (secret) inputs drawn from random variables A and B, respectively. The random variables A and B can have any distribution which fits the way inputs are chosen when the protocol is being used in a real world setting. In particular, it may be wellknown that the input of Alice is, i.e. an odd number, or English text. The aim of

Alice and Bob is to compute a well-known function[1] $f(A,B)$, such that as little information about their private inputs as possible is leaked. Note that the distributions A and B may carry information in themselves (i.e. we know that A is a vector of integers). The output $f(A,B)$ may also carry information about the inputs. We can never prevent information which is computable from the input distributions, and the output from being computed. However, we do not wish that anything else can be computed.

The natural way to model a distributed algorithm for computing a function, f, is with two randomised Turing machines, Π_f^A and Π_f^B, which share a *communication tape* [15]. By the statement "Alice sends message m to Bob," we mean that the Turing machine Π_f^A writes m to the communication tape, and that the Turing machine Π_f^B reads (and removes) it. In this formalism semi-honest simply means that Alice and Bob really do run the Turing machines Π_f^A and Π_f^B and not Turing machines devised to cheat. Clearly any non-trivial function f requires that Alice and Bob communicate. The question is: does the communication and other things observed by Alice allow her to learn something about the input of Bob, which she should not have learned (and vice versa for Bob)? Formally, the things observed by Alice are described by random variable $view_A$ which contains the input (A), output ($f(A,B)$), random choices (the Turing machine Π_f^A is probabilistic), messages seen by Alice during the execution of the protocol, and prior knowledge (and $view_B$ is defined in a similar fashion). Ideally we want to guarantee that Alice cannot compute anything from her view that she cannot compute from her own input, A, and the output $f(A,B)$. Notice that this is the same as saying that $view_A$ could be generated from A and $f(A,B)$ without ever executing the protocol (and thus without $view_A$). Clearly, if we can device an algorithm which can *simulate* $view_A$ from A and $f(A,B)$, then $view_A$ does not leak any unwanted information at all. Formally we require the existence of a probabilistic polynomial time *simulator* S such that for all *distinguisher* functions D

$$|Pr[D(S(A,f(A,B))) = 1] - Pr[D(view_A) = 1]| \leqslant \epsilon. \tag{1}$$

In the unconditional case, this means that *no* algorithm, D, should be able to tell the difference. In other words: the probability distributions of the simulated and the real view are close to within ϵ. In this case we write

$$S(A,f(A,B)) \overset{s}{\equiv} view_A. \tag{2}$$

If $\epsilon = 0$ the simulation has exactly the same probability distribution as the real view. In this case we write

$$S(A,f(A,B)) \equiv view_A. \tag{3}$$

Finally, if no *probabilistic, bounded error, polynomial time* distinguisher, D, can tell the difference (except, possibly, with negligible probability, and under the condition of a computational assumption, like the RSA assumption) we say that the protocol is computationally secure. In this case we write

$$S(A,f(A,B)) \overset{c}{\equiv} view_A. \tag{4}$$

A protocol for computing f is secure, if a simulator S exists for both Alice and Bob.

The claim of this paper is thus that there is *no* protocol for scalar products for which there exists a pair of simulators capable of simulating the views of Alice and Bob perfectly, or even approximately, except if extra assumptions are put on the distinguisher (that it is computationally bounded), or on the model itself (i.e. the presence of a third party).

2.2. Commitment schemes and oblivious transfer

A class of protocols of particular interest in this paper are commitment schemes. In a commitment scheme Alice "commits" to a value a in a way that Bob does not learn a, but such that, at a later time, Alice can prove that it was a she committed to. The standard metaphor for commitment schemes is a safety vault. Alice locks a document with the text "a" into the vault, and gives the vault to Bob, but keeps the key. To prove that she committed to a, she gives the key to Bob who can then open the save and verify her statement. We give a more precise definition in the next section. For now we just have to note that the semi-honest model is not meaningful for commitment schemes. The whole point of a commitment is to make sure that Alice will not lie about her choice of a. If she were semi-honest, a commitment scheme would not be needed, since she can just tell Bob about a whenever needed. Instead we use a "hybrid" adversarial model, where we assume that Alice is semi-honest during the commitment phase, but that she tries to change her mind in the opening phase.

We will not study oblivious transfer in detail in this paper, but the strong relationship between commitment schemes, scalar products, and oblivious transfer makes it useful to have a short look at oblivious transfer. In one-out-of-two oblivious transfer (OT_2^1) Alice (the sender) has two input messages m_0 and m_1, and Bob (the receiver) wants to learn m_b. An OT_2^1 protocol is such that Bob learns exactly m_b, and nothing else, while Alice learns nothing at all (in particular she does not know which message Bob received). It can be shown that both commitment schemes and oblivious transfer can be implemented with scalar products. This suggests that secure scalar products may be implemented in models where secure oblivious transfer exists. Even though oblivious transfer cannot be unconditionally secure, secure implementations have been demonstrated under several assumptions. It is known that oblivious transfer based on different kinds of noisy channels exist [7,6,8]. Rivest showed that oblivious transfer can be implemented with the help of a trusted initiator – a trusted third party

[1] Scalar product, in our case.

who only participates in a setup phase [25]. In [21] the authors show how oblivious transfer is possible if the role of the sender is distributed amongst several parties. And, of course, many oblivious transfer protocols exist in the computational model [23,12,3].

All protocols presented in this paper work with inputs and outputs from a finite field $\mathbb{F}$, even though the section about commitment schemes is valid for any set. Throughout the paper we use the notation $x \in_R S$ to denote that x is chosen at random from the set S according to the uniform distribution.

2.3. Commitment schemes – formal definition

A commitment scheme consists of two protocols: (1) a commitment protocol, and (2) an opening protocol. In the commitment protocol Alice has an input "string" s from a finite field $\mathbb{F}$, and at the end of the commitment protocol Bob has some output state $commit(s)$. At the end of the opening protocol Bob learns s. We require two things of a commitment scheme:

Hiding. At the end of the commitment protocol Bob knows nothing about s (that he did not know in advance) – formally, there exists a simulator, S, such that $S(\perp) \equiv view_B$, where $\perp$ means *no input* (either perfect, statistically, or computationally).
Binding. At the end of the opening protocol Bob learns s exactly – formally, there exists a polynomial time verifier, V such that $V(commit(s), open(s')) = 1$, if and only if $s = s'$.

The hiding and binding properties can be either perfectly, statistically, or computationally bounded. It is well-known that at least one of the properties has to be computationally secure, but in the following theorem we give a quantitative bound of the information leaked to an computationally unbounded adversary:

Theorem 1. *For all commitment schemes, in all commitments there exists a set $S \subseteq \mathbb{F}$ known to both Alice and Bob, such that, after the commitment protocol:*

- *Bob knows that Alice committed to some $s \in S$, and*
- *Alice can open to any $s' \in S$.*

Proof. Suppose that Alice has committed to the string s. Let c describe the conversation between Alice and Bob during the commitment protocol (all messages sent forth and back), and let $view_B$ describe the view of Bob after the commitment protocol, but before the open protocol (the random choices made by Bob, his private knowledge, and the conversation c).

Let $V_B(s')$ be the set of all views Bob can have after a commitment to s'. Define $S_B = \{s' \in \mathbb{F} | view_B \in V_B(s')\}$ as the set of all strings which have a commitment that would give Bob $view_B$. Observe that

- If a string $s' \in S_B$, then there would be a commitment to s' where Bob would have $view_B$. Thus, Bob cannot distinguish if the current commitment is to s or s'.
- If Alice can open to a string $s' \neq s$, then $s' \in S_B$.

Now let $C(s')$ be the set of all possible conversations when committing to s', and let $S_A = \{s' \in \mathbb{F} | c \in C(s')\}$ be the set of all strings which have a commitment with conversation c. Observe that

- If $s' \in S_A$, for some $s' \neq s$, then c is a valid conversation for a commitment to s'. So, when opening, Alice can pretend that she had committed to s' and can thus open to s'.
- If Alice can open to $s' \neq s$, then c must be a possible conversation when committing to s' (since c is part of $view_B$), so $s' \in S_A$.

We finally show that $S = S_A = S_B$. First assume that $s' \in S_A$, then Alice can open to s', but this implies that $s' \in S_B$, so $S_A \subseteq S_B$. Now, let $s' \in S_B$, then there is commitment to s' which would give Bob $view_B$. But then the conversation which is part of $view_B$ is a valid conversation for both s and s', so $s' \in S_A$, implying that $S_B \subseteq S_A$. □

It follows from the theorem that any unconditionally hiding commitment scheme must be such that $S = \mathbb{F}$ (all strings are consistent with what Bob has seen during the commitment protocol). An unconditional binding commitment scheme, however, must be such that $S = \{s\}$ after committing to s (so that Alice cannot open to another string $s' \neq s$). It is a corollary to this theorem that if a commitment scheme has either unconditionally hiding or binding, the validity of the other property must rely on an extra assumption.

3. No unconditionally secure scalar product

In a *scalar product protocol* Alice and Bob have d-dimensional input vectors $\bar{v}$, $\bar{w} \in \mathbb{F}^d$, respectively. They wish to compute the scalar product

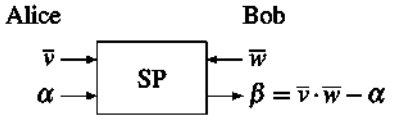

Fig. 1. Determined secret shared scalar product ($\alpha + \beta = \bar{v} \cdot \bar{w}$).

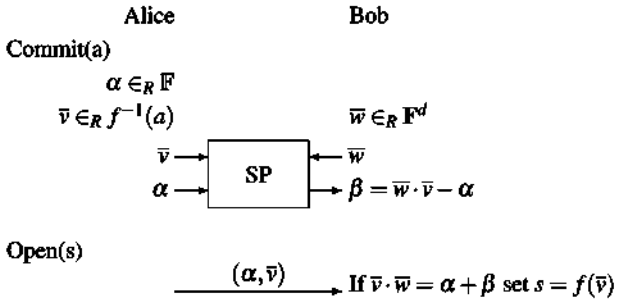

Fig. 2. Reducing commitments to scalar products.

$$\bar{v} \cdot \bar{w} = \sum_{i=1}^{d} v_i w_i, \tag{5}$$

without revealing "too much" about their input vectors.[2]

Several variants of scalar product protocols can be defined, depending on the outputs of the two parties. One approach is to give the scalar product as output to Bob (trivial SP). Alternatively, we can require that Alice and Bob each gets an *additive secret share* of the scalar product (secret shared SP or SSSP). In this paper we use a third approach where we let Alice choose her own part of the additive secret sharing in SSSP (so that, as output, Bob gets the scalar product minus Alice's share) – we call this approach determined SSSP. Fig. 1 shows the determined SSSP protocol.

It is easily seen that determined SSSP and SSSP are equivalent. We can turn an implementation of SSSP into a determined SSSP by letting Alice set her own secret share to a value α of her choice. She then computes $\beta' = \alpha - \alpha'$, where α' is the secret share she got from the SSSP, and sends β' to Bob. Bob sets his own secret share to be $\beta + \beta'$. Vice versa, given a determined SSSP, Bob chooses a random number α' and sets his own share to $\beta - \alpha'$. Bob sends α' to Alice, who sets her share to $\alpha + \alpha'$. The reason that we use determined SSSP in this paper is that it gives more natural and efficient implementations of scalar products in Section 5.

The special case of SSSP over the binary field is the computation of a Boolean *and* gate. It has long been known that it is impossible to compute and gates with unconditional security [5], however, in this paper we do not only show impossibility, but also a bound in the amount of information leaked by the more general scalar product protocols.

3.1. Commitment schemes from scalar products

We now show that a commitment scheme can be implemented by one call to determined SSSP. We reduce a commitment scheme over the field $\mathbb{G}$ to determined SSSP over $\mathbb{F}^d$, for any dimension $d > 1$. The base field of the vector space does not have to be the same as the field of the commitment scheme.

Let $A \in \mathbb{G}$ be the random variable describing the input of Alice to the commitment scheme, and let $\mathbb{F}^d$ be a vector space such that $|\mathbb{F}^d|$ is at least as large as the support of A. Furthermore, let $f : \mathbb{F}^d \to \mathbb{G}$ be an arbitrary surjective function. To commit to a string $a \in \mathbb{G}$, the sender chooses a vector $\bar{v} \in_R f^{-1}(a)$ at random, and a random value $\alpha \in \mathbb{F}$. The receiver chooses a vector $\bar{w} \in_R \mathbb{F}^d$ at random. Sender and receiver then call the scalar product protocol with vectors $\bar{v}$ and $\bar{w}$, respectively, and the sender sets his chosen secret share to α. The output, β, of the scalar product is the commitment.

To open the commitment the sender sends $(\alpha, \bar{v})$ to the receiver, who verifies that $\alpha + \beta = \bar{v} \cdot \bar{w}$. If the tests passes, he opens $a = f(\bar{v})$. An outline of this protocol can be seen in Fig. 2.

Lemma 1. *Given a scalar product protocol which is secure against semi-honest adversaries, the commitment scheme in Fig. 2 is perfectly hiding and binding when the parties are semi-honest in the commitment protocol.*

[2] Note that scalar products over finite fields are not inner products – they do not have the usual geometrical interpretation. In particular, a vector over a finite field can be "orthogonal" to itself.

Proof. Since the two parties are semi-honest in the commitment protocol the only information that the receiver gets is β, which is random, so the commitment is hiding.

Now assume that the sender can open to both $a \neq a' \in \mathbb{F}$, and let (α, v) and (α', v') be the two open messages. Since both will be accepted we have that $\alpha + \beta = v \cdot w$ and $\alpha' + \beta = v' \cdot w$. Subtracting the two equations gives us the scalar product $\alpha' - \alpha = (v' - v) \cdot w$. Since $f(v) = a \neq a' = f(v')$ the two vectors are different, so $v' - v \neq 0$. This means that the sender knows a non-trivial scalar product with the input vector of the receiver. This contradicts the security of the scalar product, and thus the sender can only open to one message. So the protocol is binding. □

Since the reduction does not rely on any assumptions, a secure implementation of determined SSSP will immediately give a commitment scheme with the same security. The fact that no unconditionally secure commitment scheme exists implies that no unconditionally secure SP exists either. We now see that it was no coincidence that all the scalar product protocols analysed in [14] were insecure. The following theorem, which is our first main result, shows how, with high probability, at least one of the two parties in a scalar product will be able to learn non-trivial information about the input vector of the other party.

Since the commitment protocol in Fig. 2 only consists in one call to a scalar product protocol, the information leakage of the commitment scheme can be directly translated into information leakage in the scalar product.

Theorem 2. *In any scalar product protocol, after each invocation, there exists a natural number* $0 < n \leqslant |\mathbb{F}^d|$ *and an algorithm* E *such that Alice learns at least* $n - 1$ *scalar products with* w *and* $\Pr[E(view_B) = v] \geqslant 1/n$.

Proof. Let a scalar product protocol be given, and let $V \in \mathbb{F}^d$ be the random variable describing the input vector of Alice. We implement a commitment scheme where Alice commits to a value from the same vector space, $\mathbb{F}^d$. Let f be any permutation of the vectors in $\mathbb{F}^d$. Alice's input to the commitment scheme is described by random variable A, where $\Pr[A = a] = \Pr[V = f^{-1}(a)]$, so that the distribution of the vector in the reduction will be the same as the original input distribution to the scalar product.

From Theorem 1 we know that there exists a set $S \subseteq \mathbb{F}$, known to both Alice and Bob, such that Bob knows that $a \in S$ and Alice can open to any $a' \in S$. Let $\{(\alpha_i, v_i)\}$ be the set of all opening messages, where $v_i = f^{-1}(a_i)$, for $a_i \in S$.

As in the proof of Lemma 1 above, we see that Alice learns $(v_i - v_j) \cdot w = \alpha_i - \alpha_j$ for all openings (α_i, v_i) and (α_j, v_j). Though some of these scalar products may be identical, by fixing i, and letting j vary over all other $|S| - 1$ values, we see that Alice learns at least $|S| - 1$ distinct scalar products.

Let E be the function which takes the view of Bob, computes the set S, and picks a random element $a' \in S$, and returns $f^{-1}(a')$. The probability, $\Pr[E(view_B) = v]$, that $a' = a$ is $1/|S|$, since $a \in S$, and a' is chosen uniformly at random, and independently of a, from S. Setting $n = |S|$ yields the desired result. □

No matter what the value of n is in a given invocation of a scalar product protocol, it can clearly never be unconditionally secure without extra assumptions.

Corollary 1. *No unconditionally secure scalar product exists.*

4. Application to previous scalar product protocol

In the paper [20] Malek and Miri propose a protocol for scalar products and prove that it is information theoretically secure over small sets. More precisely, they claim that the probability that computationally unbounded Bob can guess the input vector of Alice is $1/((p^2 - 1)(p^2 - p))$ and the probability that computationally unbounded Alice can guess the input vector of Bob is $1/p^{d-2}$, where input vectors are from the field $\mathbb{F}_{q^d}$, $q = p^n$, of characteristic p. However, we show that the probability that Bob can guess the input vector is $1/p^2 > 1/((p^2 - 1)(p^2 - p))$, which coincides with the bound of Theorem 2.

In the protocol by Malek and Miri vectors from $\mathbb{F}_q^d$ are mapped into elements from $\mathbb{F}_{q^d}$ (these two objects are, of course, isomorphic). A *basis* of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$ is a set of elements $\{\alpha_1, \ldots, \alpha_d\} \subset \mathbb{F}_{q^d}$ such that any element $u \in \mathbb{F}_{q^d}$ can be written as a unique sum $u = u_1\alpha_1 + \cdots + u_d\alpha_d$, for $u_1, \ldots, u_d \in \mathbb{F}_q$. Given a basis $\{\alpha_1, \ldots, \alpha_d\}$ we define the natural mapping from the vector space $\mathbb{F}_q^d$ to the field $\mathbb{F}_{q^d}$ as $h_{\{\alpha_1,\ldots,\alpha_d\}}(u) = \sum_{i=1}^{d} \alpha_i u_i$.

Let $\mathbb{F}_q$ be a finite field and let $\mathbb{F}_{q^d}$ be an extension field of $\mathbb{F}_q$. The *trace* of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$ is the function $Tr : \mathbb{F}_{q^d} \to \mathbb{F}_q$,

$$Tr(u) = \sum_{i=0}^{d-1} u^{q^i}, \tag{6}$$

The trace function gives rise to the definition of *dual bases*, where bases $\{\alpha_1, \ldots, \alpha_d\}$ and $\{\beta_1, \ldots, \beta_d\}$ are dual if $Tr(\alpha_i\beta_j) = \delta_{i,j}$ (where $\delta_{i,j}$ is Kronecker's delta). It follows (see [20] for details) that for two vectors $v, w \in \mathbb{F}_q^d$, and for two dual bases $\{\alpha_1, \ldots, \alpha_d\}$ and $\{\beta_1, \ldots, \beta_d\}$, the scalar product of v and w is:

$$v \cdot w = Tr(h_{\{\alpha_1,\ldots,\alpha_d\}}(v)h_{\{\beta_1,\ldots,\beta_d\}}(w)). \tag{7}$$

In the protocol by Malek and Miri the input vectors of Alice and Bob are from the vector field $\mathbb{F}_q^d$. The protocol uses (7) to compute the scalar product in the field $\mathbb{F}_{q^d}$. The protocol proceeds as shown in Fig. 3 (the notation is slightly difference from the notation in [20]).

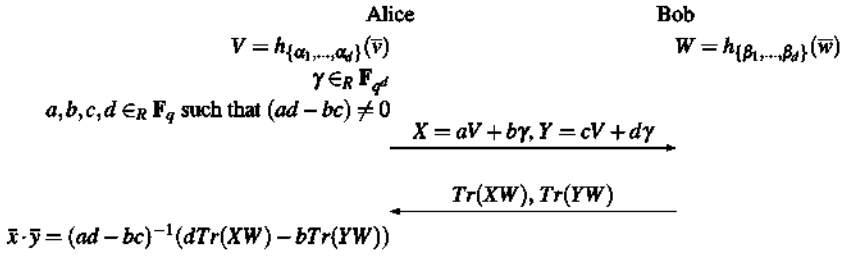

Fig. 3. Scalar product protocol from [20].

The information leakage of this protocol is almost exactly what is guaranteed by Theorem 2, except that Alice learns one more scalar product than guaranteed by the theorem. This shows that our information leakage bound is (almost) tight.

Theorem 3. *In the protocol from [20] Alice learns q^2 scalar products with $\bar{w}$ and Bob can compute $\bar{v}$ with probability $1/q^2$.*

Proof. First, note that

$$\frac{b^{-1}}{b^{-1}a - d^{-1}c}X - \frac{d^{-1}}{b^{-1}a - d^{-1}c}Y = \frac{b^{-1}aV - d^{-1}cV}{b^{-1}a - d^{-1}c} = V, \tag{8}$$

so V is a linear combination of the two elements X and Y. Since h is an isomorphism $\bar{v}$ is also a linear combination of $h^{-1}_{\{\alpha_1,\ldots,\alpha_d\}}(X)$ and $h^{-1}_{\{\alpha_1,\ldots,\alpha_d\}}(Y)$.

Next, note that

$$\alpha Tr(XW) + \beta Tr(YW) = Tr((\alpha X + \beta Y)W) = \left(\alpha h^{-1}_{\{\alpha_1,\ldots,\alpha_d\}}(X) + \beta h^{-1}_{\{\alpha_1,\ldots,\alpha_d\}}(Y)\right)\cdot\bar{w}, \tag{9}$$

for all $\alpha, \beta \in \mathbb{F}_q$.

We consider two cases: (1) $X \neq Y$, where both are non-zero, and (2) $X = Y$, or either X or Y is 0.

If $X \neq Y$ are non-zero, then (8) is a linear combination of two non-zero elements, and Bob has to guess the two coefficients to find $\bar{v}$. By choosing two coefficients at random, Bob will get the right ones with probability $1/q^2$. Alice can use (9) to compute q^2 distinct scalar products with $\bar{w}$.

If $X = Y$, or if either X or Y is 0, then (8) only has one unknown coefficient, so Bob can guess $\bar{w}$ with probability $1/q$. Similarly (9) only allows Alice to compute q scalar products. □

The nature of the information leakage in this protocol depends on the scalar field. In small scalar fields considerable information is leaked to Bob, while Alice only gains limited information. Vice versa, over large scalar fields, the probability that Bob can guess the input vector of Alice is small, while Alice a large number of scalar products.

5. Efficient and secure scalar products under various assumptions

5.1. Encryption based

A computationally secure implementation of the scalar product was given by Goethals et al. [14]. Their protocol can be based on any semantically secure additively homomorphic public-key encryption scheme ($E(x)E(y) = E(x+y)$). Let (G, E, D) be such an encryption scheme and assume that both Alice and Bob know the public key of Bob, but only Bob knows the corresponding secret key. If Alice and Bob have vectors $\bar{v} = (v_1, \ldots, v_d)$ and $\bar{w} = (w_1, \ldots, w_d)$, respectively, the scalar product proceeds as shown in Fig. 4.

A good candidate for the homomorphic encryption is the Paillier encryption scheme [22], which takes plaintexts from $\mathbb{Z}_n$ and gives ciphertexts in $\mathbb{Z}_{n^2}$ where n is an RSA prime. Goethals et al. use the Paillier scheme in their paper.

One of the primary arguments against using schemes based on homomorphic encryptions is the blowup in the message size. If we use Paillier encryption to perform scalar products over binary vectors, each bit is encrypted in 2048 bits, which is unacceptable when working with massive data sets. To overcome the large communication overhead, we need an alternative

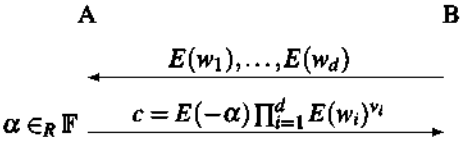

Fig. 4. Computationally secure scalar product protocol. Setting $\beta = D(c)$ gives $\alpha + \beta = \bar{v}\cdot\bar{w}$.

Table 1
Time consumption of scalar product based on ElGamal.

Bitsize	Alice (ms)	Bob (ms)	Brute-force (ms)
1	63	Negligible	Negligible
2	46	Negligible	Negligible
3	47	Negligible	62
4	62	16	406
5	62	16	1125
6	31	Negligible	11,875
7	31	Negligible	24,094
8	62	16	92,859

homomorphic encryption which has less overhead when computing scalar products over small fields. To this end we propose to use a modified version of the ElGamal encryption scheme[11] over elliptic curves which encrypts a 160-bit message to a 640-bit ciphertext for the same level of security as 1024-bit RSA.

The encryption of a message x in the elliptic curve ElGamal cryptosystem is defined as $(rP, rQ + xG)$ where P and G are two generators in the elliptic curve group, $r \in_R Z_n$ is a random integer, and n is the order of the elliptic curve group. The elliptic curve point $Q = sP$ is the public key of Alice while s is her private key. Therefore, the encryption of $(w_1, \ldots, w_d)$ by Bob will result in $(r_iP, r_iQ + w_iG)$ for $i \in \{1, \ldots, d\}$, which are sent to Alice. Upon receiving these elliptic curve point pairs, Alice performs elliptic curve point multiplication and obtains $(v_ir_iP, v_i(r_iQ + w_iG))$ for $i \in \{1, \ldots, d\}$. She then computes the encryption of her secret share $\alpha \in_R \mathbb{F}$: $(rP, rQ + \alpha G)$, and performs elliptic curve additions to obtain $((r + \sum v_ir_i)P, (r + \sum v_ir_i)Q + (\alpha + \sum v_iw_i)G)$. Decryption of this ciphertext by Bob will result in $(\sum v_iw_i)G$ from which the scalar product is calculated by using brute-force. Brute-force is necessary since discrete logarithm is difficult to compute in this elliptic curve. If the messages are chosen over a small field, the brute-force does not pose a problem as shown in the timing results below.

We conducted tests to see how practical it is to use the ElGamal encryption scheme as a building block for scalar products. We used the MIRACL library [19] to implement the secure scalar product protocol on an Intel Dual-Core Centrino PC with 2 MB cache, 2 GB RAM, and 1.83 GHz clock speed. We used vectors of length $d = 10$ over fields with different bit lengths. The timing results are listed in Table 1, where the term *brute-force* denotes the method to compute the discrete logarithm which is necessary to decrypt the message. The tests shows that for vector spaces with entries of up to 8 bits it is possible to use the ElGamal encryption scheme.

We implemented the Paillier scheme using the same experimental setting and found that Alice and Bob spends about 500 and 50 ms, respectively, in the secure scalar product computation, independent of the bit lengths of the vector entries (up to the maximum of 160 bits).

When the ElGamal scheme is used for scalar products of binary vectors, the overhead is a factor of 640. When using the Paillier scheme for vectors of 4-bit element, the overhead is $2048/4 = 512$. A hybrid scheme, which uses ElGamal for vectors over small fields, and Paillier for large fields, we can thus guarantee a worst case overhead of a factor of 640. Compared to the overhead of the suggested efficient protocol of Atallah and Du [2] which has a communication overhead of approximately 160 times of the original message size when computing scalar products in $\mathbb{Z}_{1024}^{80}$ (and is insecure), we see that scalar products based on encryption are not as inefficient as some might fear. By using a trick for "batch" computation of scalar products suggested by Goethals et al., the communication overhead can be further reduced.

5.2. Trusted third party

In this section we present a new secure scalar product protocol which uses a secure arithmetic circuit evaluation to compute field multiplication in $\mathbb{F}_{2^n}$. Under normal circumstances secure arithmetic circuit evaluation protocols are too inefficient to be of practical use. However, our protocol relies on three ideas which makes it efficient:

1. Use an arithmetic circuit evaluation protocol over shares inspired by the Boolean circuit evaluation of Goldreich [15, Section 7.1.3.3.]. The circuit evaluation protocol is secure for semi-honest players and relies on oblivious transfer to multiply bits as described below.
2. Use the Karatsuba multiplication algorithm [18] to reduce the number of and-gates.
3. Use the oblivious transfer by Rivest [25] which is considerably more efficient than other oblivious transfer protocols, but assumes the presence of a "trusted initializer" (this is the extra assumption we use to make a secure protocol possible).

The Boolean circuit evaluation protocol of Goldreich [15, Section 7.1.3.3.] is easily generalised to an arithmetic circuit evaluation protocol. An arithmetic circuit over a field $\mathbb{F}$ is an acyclic, directed graph where nodes are *arithmetic gates* (e.g. multiplication or addition) or *input/output gates* and edges are *wires*. Arithmetic gates have one or two *input wires* and multiple *output wires*. Input gates only have output wires, and an output gate only has a single input wire. The *value* of an arithmetic gate is the result of performing the arithmetic operation associated with that gate on the values of the source gates of the two input wires. To make it easier to refer to gate inputs we write *value of a wire* as shorthand for value of the source gate

of a wire. To prevent anyone from seeing the values of non-output gates, the value of each arithmetic gate is additively secret shared between the two players. This means that after evaluating a gate g Alice and Bob get values g_0 and g_1, respectively, such that the (secret) value of gate g is $g_0 + g_1$. Before starting the circuit evaluation we say that all wires are *inactive*. When a gate has been evaluated its output wires are said to be *active*. When all input wires of a gate are active, the gate can be evaluated, and in turn it's output wires become active. At the beginning of the circuit evaluation (when all wires are inactive) the input gates (which have no input wires) are evaluated to activate their output wires. The value of an input gate is the secret input value of either Alice or Bob. To evaluate an input gate g the value of which belongs to Alice, say, Alice chooses a random field element g_1 and sends it to Bob. Alice keeps $g_0 = a - g_1$, where $a \in \mathbb{F}$ is her secret input. Input gates that belong to Bob are teated in the same fashion.

For the purpose of this paper we distinguish between four kinds of arithmetic gates: addition of two secret field elements, multiplication between two secret field elements, multiplication of a secret field element from a subfield $\mathbb{G} \subseteq \mathbb{F}$ and a bit known to one party (bit-multiplication for short), and multiplication of a secret field element with a known constant. To evaluate an addition gate, each party simply adds his shares of the values of the input wires. To evaluate a gate which multiplies a known constant with a secret field element, each party multiplies his share of the value of the input wire with the known constant. To multiply two secret field elements, we use the Karatsuba multiplication algorithm explained below to replace the multiplication gate with a sub-circuit consisting of addition, multiplication by constants, and bit-multiplication gates.

To multiply a secret field element from $\mathbb{G} \subseteq \mathbb{F}$ with a bit known to one party we use one invocation of an oblivious transfer protocol. Suppose that Bob has an input bit $b \in \{0,1\}$, which we have to multiply with secret value $a = a_0 + a_1 \in \mathbb{G}$, where Alice knows a_0 and Bob knows a_1. Since $ab = a_0 b + a_1 b$, Bob can compute $a_1 b$ by himself, and all we have to do is to compute new values $z_0 + z' = a_0 b$, such that Alice only knows z_0 and Bob only knows z', and then set the secret sharing of the bit-multiplication gate to z_0 and $z_1 = z' + a_1 b$ for Alice and Bob, respectively. To this end we call a 1-out-of-2 oblivious transfer where Alice is the sender and Bob is the receiver. Alice first chooses a random field element $z_0 \in \mathbb{G}$, and computes messages $m_i = ia_0 - z_0$, for $i = 0, 1$. Alice uses inputs m_0, m_1 to the oblivious transfer, and Bob uses input b. As a result, Bob gets $z' = m_b = ba_0 - z_0$, and sets $z_1 = z' + ba_1$. Now Alice and Bob have the required additive secret sharing $z_0 + z_1 = z_0 + z' + ba_1 = ab$.

A multiplication gate of arbitrary field elements can be reduced to a sub-circuit consisting of addition, multiplication by constant, and bit-multiplication gates. Since each bit-multiplication requires one call to oblivious transfer, while the other gates require no interaction, we need a sub-circuit which requires as few bit-multiplications as possible. To this end we use a modified version of the Karatsuba multiplication algorithm which uses an expected number of $n^{\log_2(3)}$ bit-multiplications to multiply two elements from $\mathbb{F}_{2^n}$. To multiply $x,\ y \in \mathbb{F}_{2^n}$ we fix a basis $\{a, 1\}$ for $\mathbb{F}_{2^n}$ over $\mathbb{F}_{2^{n/2}}$ (for simplicity we assume that n is a power of 2), we can then split x and y into smaller parts, $x_h,\ x_l,\ y_h,\ y_l \in \mathbb{F}_{2^{n/2}}$, such that $x = ax_h + x_l$ and $y = ay_h + y_l$, and recursively compute the three multiplications $A = x_h y_h$, $B = x_l y_l$, and $C = (x_h + x_l)(y_h + y_l)$. The product is $xy = a^2 A + a(C - A - B) + B$, which can be computed with addition and multiplication with constant when the secret values A, B, and C have been computed. At the bottom of the recursion, the multiplication of two 1-bit integers is simply a bit-multiplication as described above.

While most implementations of oblivious transfers are too inefficient for our purpose, the oblivious transfer proposed by Rivest [25] gives a very efficient scalar product protocol. The oblivious transfer by Rivest relies on a *trusted initializer* – a third party who only participates in the protocol in an initialization phase, and does not collude with any of the other players. In the oblivious transfer by Rivest, the trusted initializer generates two random strings $x_0,\ x_1 \in \mathbb{G}$, and a random bit $b \in \{0,1\}$. He sends (x_0, x_1) to the sender, and (b, x_b) to the receiver. The sender and receiver now have a "random instance" of an oblivious transfer. To realize a real oblivious transfer of messages $m_0,\ m_1 \in \mathbb{G}$ the receiver sends the bit $c' = c \oplus b$ to the sender, where c is the index of the message he wants to learn. The sender replies with the message (m'_0, m'_1), where $m'_i = m_i + x_{c' \oplus i}$, for $i = 0, 1$. The receiver can now recover $m_c = m'_c - x_{c' \oplus c} = m'_c - x_b$. For proof of security and other details see [25].

Putting together the peaces above, we can compute the scalar product of two vectors $\bar{v} = (v_1, \ldots, v_d)$ and $\bar{w} = (w_1, \ldots, w_d)$ known to Alice and Bob, respectively. The scalar product of the two vectors is done by the arithmetic expression

$$\bar{v} \cdot \bar{w} = \sum_{i=1}^{d} v_i w_i, \tag{10}$$

which is easily converted into an arithmetic circuit. The circuit has $2d$ input gates, one for each entry in each of the two vectors. For each $i \in \{1, \ldots, d\}$ we have a multiplication gate where the input wires are the output wires of input gates v_i and w_i. Internally the d multiplication gates are translated into Karatsuba multiplication circuits as described above. The output wires of the d multiplication gates are connected to a binary tree of addition gates. The output of the root of the addition tree is connected to the final output gate of the circuit.

5.2.1. Security

The only interaction which takes place in our scalar product protocol is the initial additive secret sharing of the inputs, and during the oblivious transfer. If Alice and Bob are semi-honest, and the oblivious transfer protocol is secure, then their outputs from each oblivious transfer are additive secret sharings of bit-multiplications. Clearly this does not reveal

any information about the input at all (since Alice chose her share uniformly at random). The security of the protocol thus relies on the security of the oblivious transfer protocol used (Rivest oblivious transfer in our case).

5.2.2. Efficiency

To multiply two elements from the field $\mathbb{F}_{2^n}$, Karatsuba needs approximately $n^{\log_2(3)}$ bit-multiplications. Each bit-multiplication requires one Rivest bit-oblivious transfers, which requires 7 bits of communication (2 bits from the third party to each of Alice and Bob, one bit from Bob to Alice, and 2 bits from Alice to Bob). In the scalar product protocol we first have to share the $2d$ field elements, and then perform d multiplications, so we expect to send approximately $2dn + 7n^{\log_2(3)}d$ bits to compute the scalar product of two vectors over $\mathbb{F}_{2^n}^d$. As an example, it will take $1765d$ bits of communication to perform scalar products between vectors in $\mathbb{F}_{2^{32}}^d$.

Since the oblivious transfers only depend on the inputs of Alice and Bob, they can all be done in parallel, so we only need one round of communication.

The computational cost of the algorithm is minimal, since no cryptographic operations are involved – only simple field arithmetic.

5.3. Comparison of scalar product protocols

We evaluate the performance of the two proposed scalar product protocols in a real-world scenario, and compare them to the scalar product based on Paillier encryption [14], and a trivial (insecure) scalar product.

The apriori algorithm is commonly used in association rule mining. Clifton and Vaidya showed [28] how to securely implement the apriori algorithm on vertically partitioned data by using a secure implementation of scalar products. To give a realistic comparison of the performance of the compared scalar products we have implemented the apriori algorithm proposed in [28] and applied it to the "Congressional Voting Records" dataset from the UCI Machine Learning Repository [26]. The voting records dataset consists of 232 records, each with 17 binary records. When using the protocol from [28] on this dataset, the scalar products are computed on 232-dimensional binary vectors. We are aware that the case of binary vectors is the best case for both the ElGamal based scalar product presented in Section 5.1 and the third party based protocol presented in Section 5.2. However, the benefit earned by the ElGamal based protocol is only in the brute-force decoding step, and the benefit in the third party based protocol is only in the number of parallel executions of the Rivest oblivious transfer. In both cases, the benefit earned is small compared to the overall timings in distributed environment shown below.

Our experimental setting consists of four virtual machines running on a quad-core computer. One of the virtual machines is a software router, which we use to control the bandwidth and latency between the three remaining computers, which run Alice, Bob, and the third party, respectively. We use Debian linux installed on VMware for the three virtual application computers, and FreeBSD installed on VMware as software router. The application computers are located on each their virtual network, and use the FreeBSD software router as internet gateway, thus giving full control over bandwidth and latency. A tutorial for setting up the software router can be found at [10]. The host computer is an Intel Core 2 Quad Q6600 with 2.4 GHz CPU and 4 GB of ram memory running Debian. Each virtual machine is given only 256 MB of memory to prevent swapping on the host computer.

Fig. 5a shows the time it takes to perform the apriori algorithm on the voting records dataset for different bandwidth when the node to node latency is fixed at 50 ms. Fig. 5b shows the effect on latency on the timing when then bandwidth if fixed to 100 kbps. All timings are computed as average of five experiments.

Fig. 5a clearly shows that the improvement proposed in Section 5.1 (marked ElGamal in the figure) gives a considerable speedup over the method proposed in [14] which is based on Paillier encryption. It can also be seen that the method based on a trusted third party, proposed in Section 5.2, has the best performance among the secure protocols for bandwidth.

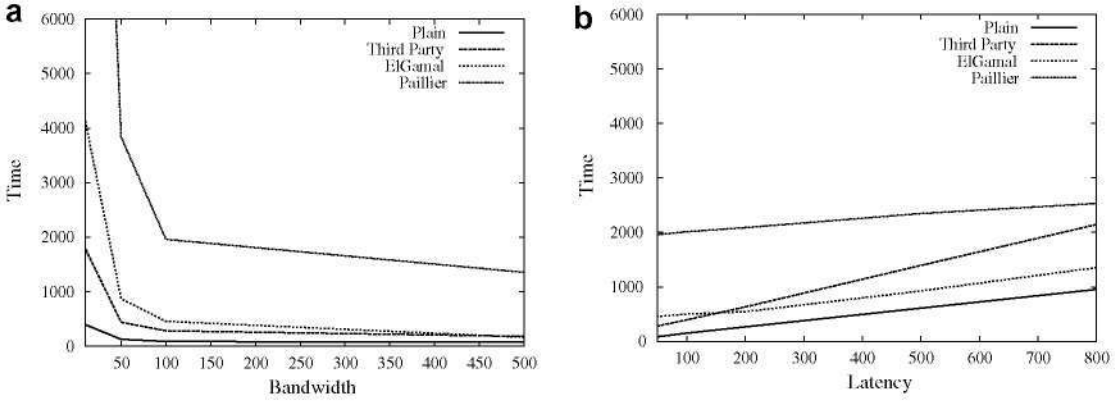

Fig. 5. Timing results. (a) Time vs. bandwidth. Latency fixed at 50 ms. (b) Time vs. latency. Bandwidth fixed at 100 kbps.

Fig. 5b shows similar results. However, the performance of the third party based protocol drops considerably for high latencies. This is due to the fact that each Rivest oblivious transfer in each scalar product needs four rounds of communication compared to only two rounds for all the other protocols. High latency therefor punishes the third party protocol. Notice that a more careful implementation of Rivest oblivious transfer, where Alice and Bob get all the randomness needed from the third party in one message at the beginning of the protocol would reduce the number of communication rounds to two per oblivious transfer, plus one large bulk message in the beginning of the protocol. With this modification the third party protocol would perform best in all scenarios.

6. Conclusion

We show that no unconditionally secure protocol for scalar product exists for two semi-honest parties without extra assumptions. It follows from this result that some of the scalar product protocols suggested in the literature are not secure. In particular, we show that in any attempt to implement a scalar product protocol without any extra assumptions, either Alice learns $n-1$ scalar products with Bob's input vector, or Bob learns the input vector of Alice with probability $1/n$.

On the other hand, we demonstrated two efficient scalar product protocols which are secure in alternative models. The first scalar product protocol is an improvement of the computationally secure protocol previously presented in [30,14,32]. The other scalar product protocol is a novel protocol which is very efficient compared to excising protocols, and whose security relies on a "trusted initializer".

References

[1] Rakesh Agrawal, Ramakrishnan Srikant, Privacy-preserving data mining, in: SIGMOD'00: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, ACM Press, 2000, pp. 439–450.
[2] Mikhail J. Atallah, Wenliang Du, Privacy-preserving cooperative statistical analysis, in: Proceedings of the 17th Annual Computer Security Applications Conference, 2001, ACSAC 2001, 2001, pp. 102–110.
[3] Mihir Bellare, Silvio Micali, Non-interactive oblivious transfer and applications, in: Advances in Cryptology – CRYPTO'89, A Conference on the Theory and Applications of Cryptographic Techniques, Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1990, pp. 547–557.
[4] Michael Ben-Or, Shafi Goldwasser, Avi Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, in: Proceedings of the 19th Annual ACM Conference on Theory of Computing (STOC), ACM Press, 1988, pp. 1–10.
[5] Benny Chor, Eyal Kushilevitz, A zero-one law for boolean privacy, in: STOC'89: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, Lecture Notes in Computer Science, ACM, 1989, pp. 62–72.
[6] Claude Crépeau, Efficient cryptographic protocols based on noisy channels, in: Advances in Cryptology – EUROCRYPT'97: International Conference on the Theory and Application of Cryptographic Techniques, Lecture Notes in Computer Science, vol. 1233, Springer-Verlag, 1997, pp. 306–317.
[7] Claude Crépeau, Joe Kilian, Achieving oblivious transfer using weakened security assumptions, in: Proceedings of the 29th annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 1988, pp. 42–52.
[8] Ivan Damgård, Joe Kilian, Louis Salvail, On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions, Technical Report RS-98-37, BRICS, University of Aarhus, 1998.
[9] Kamalika Das, Kanishka Bhaduri, Kun Liu, Hillol Kargupta, Distributed identification of top-l inner product elements and its application in a peer-to-peer network', IEEE Transactions on Knowledge and Data Engineering 20 (4) (2008) 475–488.
[10] Dummynet in wmware. <http://www.codefromthe70s.org/dummynet.aspx>.
[11] Taher ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, in: Advances in Cryptology: Proceedings of CRYPTO 84, Lecture Notes in Computer Science, vol. 196, Springer-Verlag, 1985.
[12] Shimon Even, Oded Goldreich, Abraham Lempel, A randomized protocol for signing contracts, Communications of the ACM 28 (6) (1985) 637–647.
[13] Michael J. Freedman, Kobbi Nissim, Benny Pinkas, Efficient private matching and set intersection, in: Advances in Cryptology – EUROCRYPT 2004, Lecture Notes in Computer Science, 2004, pp. 1–19.
[14] Bart Goethals, Sven Laur, Helger Lipmaa, Taneli Mielikäinen, On private scalar product computation for privacy-preserving data mining, in: Information Security and Cryptology – ICISC 2004, Lecture Notes in Computer Science, vol. 3506, Springer-Verlag, 2005, pp. 104–120.
[15] Oded Goldreich, The foundations of cryptography, Basic Applications, vol. 2, Cambridge University Press, 2004.
[16] Oded Goldreich, Silvio Micali, Avi Wigderson, How to play any mental game, in: Proceedings of the 19th Annual ACM Conference on Theory of Computing (STOC), ACM Press, 1987, pp. 218–229.
[17] Ioannis Ioannidis, Ananth Grama, Mikhail J. Atallah, A secure protocol for computing dot-products in clustered and distributed environments, in: Proceedings of the International Conference on Parallel Processing (ICPP'02), December 2002, pp. 379–384.
[18] A. Karatsuba, Yu Ofman, Multiplication of many-digital numbers by automatic computers, Nauk SSSR 145 (1962) 293–294 (Translation in Physics-Doklady 7 (1963) 595–596).
[19] Shamus Software LTD. Miracl, 2005. <http://www.shamus.ie>.
[20] Behzad Malek, Ali Miri, Secure dot-product protocol using trace functions, in: Proceedings of the 2006 IEEE International Symposium on Information Theory, July 2006, pp. 927–931.
[21] Moni Naor, Benny Pinkas, Distributed oblivious transfer, in: Advances in Cryptology – ASIACRYPT 2000: Sixth International Conference on the Theory and Application of Cryptology and Information Security, Lecture Notes in Computer Science, vol. 1976, Springer-Verlag, 2000.
[22] Pascal Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Advances in Cryptology – EUROCRYPT'99, International Conference on the Theory and Application of Cryptographic Techniques, Lecture Notes in Computer Science, Springer-Verlag, May 1999, pp. 223–238.
[23] Michael O. Rabin, How to exchange secrets by oblivious transfer, Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.
[24] Pradeep Ravikumar, William W. Cohen, Stephen E. Fienberg, A secure protocol for computing string distance metrics, in: Workshop on Privacy and Security Aspects of Data Mining, 2004, pp. 40–46.
[25] Ronald L. Rivest, Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer, August 1999. <http://people.csail.mit.edu/rivest/publications.html>.
[26] UCI machine learning repository. <http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records>.
[27] Jaideep Vaidya, Privacy-preserving linear programming, in: Proceedings of the 2009 ACM Symposium on Applied Computing, 2009, pp. 2002–2007.
[28] Jaideep Vaidya, Chris Clifton, Privacy preserving association rule mining in vertically partitioned data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 639–644.
[29] Jaideep Vaidya, Christopher W. Clifton, Privacy-preserving kth element score over vertically partitioned data, IEEE Transactions on Knowledge and Data Engineering 21 (2) (2009) 253–258.

[30] Rebecca Wright, Zhiqiang Yang, Privacy-preserving bayesian network structure computation on distributed heterogeneous data, in: KDD'04: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM, 2004, pp. 713–718.
[31] Weijia Yang, Shangteng Huang, Data privacy protection in multi-party clustering, Data and Knowledge Engineering 67 (1) (2008) 185–199.
[32] Zhiqiang Yang, Rebecca Wright, Privacy-preserving computation of bayesian networks on vertically partitioned data, IEEE Transactions on Knowledge and Data Engineering 18 (9) (2006) 1253–1264.
[33] Andrew C. Yao, Protocols for secure computations (extended abstract), in: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, IEEE, 1982, pp. 160–164.

Thomas B. Pedersen received his M.S. (2002) and Ph.D (2006) degrees in computer science from the University of Aarhus, Denmark. From 2006 to 2008 he was a postdoctural fellow at Sabanci Universtity, where he is now a visiting faculty member. His main research interests are cryptographic protocols, security, and privacy.

Erkay Savaş received the BS (1990) and MS (1994) degrees in electrical engineering from the Electronics and Communications Engineering Department at Istanbul Technical University. He completed the Ph.D. degree in the Department of Electrical and Computer Engineering (ECE) at Oregon State University in June 2000. He had worked for various companies and research institutions before he joined Sabanci University as an assistant professor in 2002. He is the director of the Cryptography and Information Security Group (CISec) of Sabanci University. His research interests include cryptography, data and communication security, privacy in biometrics, trusted computing, security and privacy in data mining applications, embedded systems security, and distributed systems. He is a member of IEEE, ACM, the IEEE Computer Society, and the International Association of Cryptologic Research (IACR).