On Private Scalar Product Computation for Privacy-Preserving Data Mining

Bart Goethals[1], Sven Laur[2], Helger Lipmaa[2], and Taneli Mielikäinen[1]

[1] HIIT Basic Research Unit
Department of Computer Science
University of Helsinki, Finland
{goethals,tmielika}@cs.helsinki.fi
[2] Laboratory for Theoretical Computer Science
Department of Computer Science and Engineering
Helsinki University of Technology, Finland
{slaur,helger}@tcs.hut.fi

Abstract. In mining and integrating data from multiple sources, there are many privacy and security issues. In several different contexts, the security of the full privacy-preserving data mining protocol depends on the security of the underlying private scalar product protocol. We show that two of the private scalar product protocols, one of which was proposed in a leading data mining conference, are insecure. We then describe a provably private scalar product protocol that is based on homomorphic encryption and improve its efficiency so that it can also be used on massive datasets.
Keywords: Privacy-preserving data mining, private scalar product protocol, vertically partitioned frequent pattern mining.

1 Introduction

Within the context of privacy-preserving data mining, several private (shared) scalar product protocols [DA01b,DA01a,DZ02,VC02] have been proposed. The goal is that one of the participants obtains the scalar product of the private vectors of all parties. Additionally, it is often required that no information about the private vectors, except what can be deduced from the scalar product, will be revealed during the protocol. Moreover, since data mining applications work with a huge amount of data, it is desirable that the scalar product protocol is also very efficient. A secure scalar product protocol has various applications in privacy-preserving data mining, starting with privacy-preserving frequent pattern mining on vertically distributed database [VC02] and ending with privacy-preserving cooperative statistical analysis [DA01a].

To give an idea of how such a protocol can be used, let us look at the protocol by Vaidya and Clifton for computing frequent itemsets from vertically partitioned transaction database [VC02]. A transaction database is a multi-set of subsets (*transactions*) of some finite set (of *items*). A transaction database can be seen also as a binary matrix where each row corresponds to a transaction, each column corresponds to an item, and there is one in the entry (i, j) if and only if the transaction i contains the item j. An itemset is a subset of items. The frequency of an itemset in a transaction database is

the fraction of transactions containing the itemset as their subset. (The support of an itemset is its frequency multiplied by the number of transactions in the database.) The σ-frequent itemsets (i.e., the frequent itemsets with minimum frequency threshold σ) in a transaction database are the itemsets with frequency at least σ. Thus, mining the σ-frequent itemsets is equivalent to finding all subsets of columns of the binary matrix where at least a σ-fraction of rows have only ones in those columns. In a frequent itemset mining protocol for a vertically partitioned transaction database one party, Alice, has the projection of the database onto some items and another party, Bob, has the projection of database onto the rest of the items. The frequent itemset mining protocol of Vaidya and Clifton is based on the property that an itemset can be frequent only if all of its subsets are frequent. The candidate itemsets are generated and tested level-wise as in the APRIORI algorithm [AMS$^+$96].

If an itemset contains items of only one party, then the party can compute the frequency privately and share it with the other parties without any additional privacy problems. The main challenge occurs when the support of a candidate itemset containing items from both parties needs to be computed. In that case, each party first computes which of the transactions contain the itemset within their own part of the database. This kind of information can be conveniently represented as binary vectors in which the ith entry represents whether or not the itemset is contained in the ith transaction. The number of transactions containing the itemset in the combined transaction database amounts to the scalar product between the corresponding binary vectors of Alice and Bob. A protocol, given by Vaidya and Clifton [VC02], attempts to compute the scalar product in a secure manner, by computing the scalar product on scrambled versions of the binary vectors, such that in the end of the protocol, both parties obtain the joint support without ever seeing each others vector. Their protocol reveals the supports of some infrequent itemsets, as not all candidate itemsets are frequent; this can be avoided by combining private shared scalar product protocols and Yao's circuits for frequency testing.

In this paper, we show that the private scalar product protocol of Vaidya and Clifton [VC02] is not private. Additionally, we are able to break another private (shared) scalar product protocol which was recently proposed by Du and Atallah [DA01a]. Our attacks against the Vaidya-Clifton and Du-Atallah protocols work in the simplest cryptographic model: namely, they enable one of the two parties to retrieve the private input of another party with probability, very close to 1, after the two parties have executed the corresponding protocol once.

While the attacks do not work for all possible private vectors of Alice and Bob, they show that before applying the Vaidya-Clifton and Du-Atallah protocols, one must carefully analyse whether it is safe to apply these protocols in any concrete case. Moreover, the provided attacks can be readily generalised to work for a much larger fraction of private vectors in a more complex model where attack's success probability does not have to be 1 (but just large enough for practical purposes, say 0.001) and/or when Alice and Bob re-execute the corresponding scalar product protocols from [DA01a,VC02] with similar private vectors. (Scalar product protocol from [DA01b] was recently analysed in [LL04b].)

As a positive result, we describe a cryptographic protocol for computing scalar product. We prove that the new scalar product protocol is private—in a strong cryptographic

sense—under standard cryptographic assumptions. More specifically, no probabilistic polynomial time algorithm substituting Alice (resp., Bob) can obtain a non-negligible amount of information about Bob's (resp., Alice's) private input, except what can be deduced from the private input and private output of Alice (resp., Bob). This means, in particular, that this protocol can be used a polynomial number of times (in the security parameter) with *any* private vectors of Alice and Bob in *any* context. In practice, the latter means "an arbitrary number of times". Finally, we show that by using some optimisation tricks, the proposed protocol can be made very efficient: we show how to separately optimise for Alice's and Bob's computation, and for the communication of the new protocol. In particular, the communication-optimal version is more communication-efficient than either of the Vaidya-Clifton or the Du-Atallah protocols.

Road-map. In Section 2, we describe the necessary cryptographic preliminaries. In Section 3, we analyse some previous private scalar product protocols. In Section 4, we propose a new scalar product protocol, prove its security and propose some important optimisations. We finish with conclusions and acknowledgements.

2 Cryptographic Preliminaries

Secure Multi-Party and Two-Party Computation. To guarantee that a protocol is secure in as many applications as possible, one should use the secure multi-party and two-party techniques [Gol04]. Briefly, a two-party protocol between Alice and Bob is *secure* when privacy and correctness are guaranteed for both Alice and Bob. It is said that a protocol *protects privacy*, when the information that is leaked by the distributed computation is limited to the information that can be learned from the designated output of the computation [Pin02].

There are several different security models where one can prove the security of a protocol in. The simplest setting is the *semi-honest* model, where it is assumed that both Alice and Bob follow the protocol, but they are also curious: that is, they store all exchanged data and try to deduce information from it. In the *malicious model*, no assumption is made about the behaviour of Alice and Bob, and it is required that the privacy of one party is preserved even in the case of an arbitrary behaviour of the second party. Most of the papers on privacy-preserving data mining provide only security in the semi-honest model. Such a protocol can be made secure in the malicious model when accompanied with zero-knowledge proofs that both parties follow the protocol. However, such proofs are usually too inefficient to be used in data mining applications.

Homomorphic public-key cryptosystems. A public-key cryptosystem Π is a triple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic polynomial-time algorithms for key-generation, encryption and decryption. The security of a public-key cryptosystem is determined by a security parameter k. For a fixed k, it should take more than polynomial in k operations to break the cryptosystem. Together with increased security, larger k means also larger keys and ciphertexts. The key generation algorithm generates, on input $1^k = 1 \ldots 1$ (k ones) a valid pair $(\mathsf{sk}, \mathsf{pk})$ of private and public keys that corresponds to the security parameter k. For a fixed key pair $(\mathsf{sk}, \mathsf{pk})$, let $P(\mathsf{sk})$ denote the plaintext space of

Π. The encryption algorithm Enc takes as an input a plaintext $m \in P(\mathsf{sk})$, a random value r and a public key pk and outputs the corresponding ciphertext $\mathsf{Enc}_{\mathsf{pk}}(m;r)$. The decryption algorithm Dec takes as an input a ciphertext c and a private key sk (corresponding to the public key pk) and outputs a plaintext $\mathsf{Dec}_{\mathsf{sk}}(c)$. It is required that $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m;r)) = m$ for any $m \in P(\mathsf{sk})$, pk and r.

A public-key cryptosystem is *semantically secure* (IND-CPA secure) when a probabilistic polynomial-time adversary cannot distinguish between random encryptions of two elements, chosen by herself. We denote the encryption of a message m by $\mathsf{Enc}_{\mathsf{pk}}(m;r)$, where pk is the corresponding public key and r is the used random string. A public-key cryptosystem is *homomorphic* when $\mathsf{Enc}_{\mathsf{pk}}(m_1;r_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2;r_2) = \mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2; r_1 \cdot r_2)$, where $+$ is a group operation and $\cdot$ is a groupoid operation. This means that a party can add encrypted plaintexts by doing simple computations with ciphertexts, *without* having the secret key. Usually, $P(\mathsf{sk}) = \mathbb{Z}_m$ for some large m. One of the most efficient currently known semantically secure homomorphic cryptosystems was proposed by Paillier cryptosystem [Pai99] and then improved by Damgård and Jurik [DJ01]. In Paillier's case, $P(\mathsf{sk}) = \mathbb{Z}_m$ with $m \geq 2^{1024}$. One can effectively assume that m is as large as say 2^{4096}, when using the Damgård-Jurik cryptosystem [DJ01]. We will assume that k is the bit length of the plaintexts, thus $k \geq 1024$.

Oblivious transfer. In an $\binom{n}{1}$-oblivious transfer protocol, Bob has a database $(\mathcal{D}_1, \ldots, \mathcal{D}_n)$ and Alice has an index $i \in [n]$. The goal is for Alice to retrieve the element $\mathcal{D}_i$ without revealing her index i to Bob, and Bob does not want Alice to get to know anything about the other elements in his database apart from the element she asks for. Recently, Lipmaa [Lip04] proposed an asymptotically efficient $\binom{n}{1}$-oblivious transfer protocol with communication $\Theta(\log^2 n)k$.

3 Cryptanalysis of Proposed Private SP Protocols

Before cryptanalysing some of the previously proposed private scalar product and private shared scalar product protocols, we must define what does it mean to attack one. Next, we will give a somewhat intuitive definition. For simplicity, we will require that all arithmetic is done in $\mathbb{Z}_m$ for some m.

We call a protocol between Alice and Bob a *scalar product* (SP) protocol when Bob obtains, on Alice's private input $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{Z}_m^N$ and on Bob's private input $\boldsymbol{y} = (y_1, \ldots, y_N) \in \mathbb{Z}_m^N$, the scalar product $\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^N x_i y_i$. A protocol is a *shared scalar product (SSP) protocol* when Alice receives a uniformly distributed random value $s_A \in \mathbb{Z}_m$ and Bob receives a dependent uniformly distributed random value $s_B \in \mathbb{Z}_m$, such that $s_A + s_B \equiv \boldsymbol{x} \cdot \boldsymbol{y} \pmod m$. A scalar product protocol is *private* when after executing the protocol, Bob obtains no more knowledge than $\boldsymbol{x} \cdot \boldsymbol{y}$ and Alice obtains no new knowledge at all. In particular, Alice gets to know nothing new about Bob's vector and Bob gets to know nothing about Alice's vector that is not implied by $\boldsymbol{x}$ and $\boldsymbol{x} \cdot \boldsymbol{y}$. A *private shared scalar product protocol* is defined analogously.

Recently, several researchers from the data mining community have proposed private SSP and SP protocols [DA01b,DA01a,DZ02,VC02], that were primarily meant to

PRIVATE INPUT OF ALICE: $\boldsymbol{x} \in \{0,1\}^N$
PRIVATE INPUT OF BOB: $\boldsymbol{y} \in \{0,1\}^N$
PRIVATE OUTPUT OF BOB: Scalar product $\boldsymbol{x} \cdot \boldsymbol{y} \mod m$.

1. Alice and Bob jointly do:
   Generate a random invertible $N \times N$ matrix C.
2. Alice does:
   Generate a random vector $\boldsymbol{p} \in \mathbb{Z}_m^N$.
   Send $\boldsymbol{u} \leftarrow \boldsymbol{x} + C\boldsymbol{p}$ to Bob.
3. Bob does:
   Generate ℓ random values $s_1, \ldots, s_\ell \in \mathbb{Z}_m$.
   Send $\boldsymbol{v} \leftarrow C^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{r}$, where $\boldsymbol{r}[\boldsymbol{i}] \leftarrow s_i\mathbf{1}$, to Alice.
4. Alice does:
   Set $t_0 := \boldsymbol{v} \cdot \boldsymbol{p}$.
   For $i \in \{1, \ldots, \ell\}$, set $t_i := \sum_{j=1}^{n} p[i]_j$.
   Send $(t_0, t_1, \ldots, t_\ell)$ to Bob.
5. Bob does:
   Return $\boldsymbol{u} \cdot \boldsymbol{y} - t_0 + \sum_{i=1}^{\ell} s_i t_i$.

Protocol 1: Vaidya-Clifton private shared scalar product protocol. (All computations are done modulo a public m.)

be used in the context of privacy-preserving data mining. Most of the proposed solutions try to achieve information-theoretical security—that is, without relying on any computational assumption—by using additive or linear noise to mask the values. In almost all such solutions, one can construct a system of linear equations based on the specification of the protocol, and solve it for the secret values. We will next demonstrate that explicitly in the case of the protocols from [DA01a,VC02].

3.1 Vaidya-Clifton Private Scalar Product Protocol

First, we analyse the Vaidya-Clifton private SP protocol [VC02], depicted by Protocol 1. For the sake of simplicity, we assume that the database size is $N = \ell n$, where n is a block size and ℓ is the number of blocks. We represent each N-dimensional vector $\boldsymbol{z}$ either as $\boldsymbol{z} = (z_1, \ldots, z_N)$ or $\boldsymbol{z} = (\boldsymbol{z}[\mathbf{1}], \ldots, \boldsymbol{z}[\boldsymbol{\ell}])$, where $\boldsymbol{z}[\boldsymbol{i}] = (z_{(i-1)n+1}, \ldots, z_{in})$. We denote the n-dimensional vectors $(1, \ldots, 1)$ and $(0, \ldots, 0)$ by $\mathbf{1}$ and $\mathbf{0}$.

Protocol 1 is a slight modification of the original Vaidya-Clifton protocol. Namely, in the original protocol all scalars belong to $\mathbb{R}$, while in Protocol 1 they belong to $\mathbb{Z}_m$ with $m > N$. Our modifications make the protocol more applicable and also more secure for the next reasons. First, as computers can use only limited precision, there will be stability and correctness problems when computing over real numbers. Second, adding random noise r from $\mathbb{R}$ to value x from $\mathbb{R}$ does not perfectly hide x since it is impossible to choose r uniformly at random from $\mathbb{R}$, or even from $\mathbb{N}$. Therefore, cryptanalysis of the original Vaidya-Clifton protocol is simpler and attacks against it are more dangerous when we consider their protocol as working in $\mathbb{R}$.

In the following, we explicitly assume that m is prime. Proposed attacks also work with composite m, but then one would have to tackle many insubstantial

yet technical details. We will also establish some additional notation. First, for any $\mathcal{I} = \{i_1, \ldots, i_j\} \subseteq [N]$ with $|\mathcal{I}| = j$, any vector $\boldsymbol{x}$ and any matrix M, let $\boldsymbol{x}_\mathcal{I} = (x_{i_1}, \ldots, x_{i_j})$ and $M_\mathcal{I}$ denote the sub-matrix of M that consists of the rows $\mathcal{I} = \{i_1, \ldots, i_j\}$. Second, C is invertible and known to both Alice and Bob. Therefore, define $\boldsymbol{a_i} := (C^{\mathrm{T}})^{-1}\boldsymbol{e_i} \mod m$, where $\boldsymbol{e_i}[\boldsymbol{j}] = \mathbf{1}$ if $i = j$ and $\boldsymbol{e_i}[\boldsymbol{j}] = \mathbf{0}$, otherwise. Define $\boldsymbol{\omega} := (C^{\mathrm{T}})^{-1}\boldsymbol{v}$. Then $(C^{\mathrm{T}})^{-1}\boldsymbol{r} \equiv (C^{\mathrm{T}})^{-1}(s_1\mathbf{1}, \ldots, s_\ell\mathbf{1}) \equiv \sum_{i=1}^{\ell} s_i\boldsymbol{a_i} \pmod m$, $\boldsymbol{\omega} \equiv \boldsymbol{y} + \sum_{i=1}^{\ell} s_i\boldsymbol{a_i} \pmod m$ and $t_i \equiv \boldsymbol{e_i} \cdot \boldsymbol{p} \equiv \boldsymbol{a_i} \cdot C\boldsymbol{p} \pmod m$ for $i \geq 1$.

First, we show that if the vector $\boldsymbol{y}$ has a low support then Alice is guaranteed to learn half coefficients y_i—and with a high probability the whole vector $\boldsymbol{y}$—after just executing Protocol 1 once.

Lemma 1. *As previously, let* $\mathrm{supp}(\boldsymbol{y}) := |\{y : y_i \neq 0\}|$ *be the support of* $\boldsymbol{y}$*. Assume that* $N \geq (2\,\mathrm{supp}(\boldsymbol{y}) + 1)\ell$*. After just executing Protocol 1 once, a semi-honest Alice obtains at least half of the coefficients of* $\boldsymbol{y}$*, with probability* 1*, by solving* $2\,\mathrm{supp}(\boldsymbol{y})+1$ *systems of linear equations in* ℓ *variables.*

Proof. Let M be the matrix with column vectors $\boldsymbol{a_1}, \ldots, \boldsymbol{a_\ell}$. Let $\boldsymbol{s} = (s_1, \ldots, s_\ell)$. The attack is based on the observation that the equality $M\boldsymbol{s} \equiv \boldsymbol{\omega} - \boldsymbol{y} \pmod m$ gives Alice a system of N linear equations in ℓ unknowns s_j. The values v_i and vectors $\boldsymbol{a_1}, \ldots, \boldsymbol{a_\ell}$ are known to Alice; the values $y_i \in \{0, 1\}$ are unknown. Alice partitions the set $[N]$ iteratively into $\geq N/\ell$ (non-empty) parts $\mathcal{I}_k$ as follows: Denote $\mathcal{J}_k := [N] \setminus \bigcup_{i<k} \mathcal{I}_k$. Alice chooses an $\mathcal{I}_k \subseteq \mathcal{J}_k$, such that the matrix $M_{\mathcal{I}_k}$ has the maximal possible rank with respect to $\mathcal{J}_k$ and $\mathcal{I}_k$ is minimal unless the rank of $M_{\mathcal{J}_k}$ is zero. In particular, $M_{\mathcal{J}_k} = D_k M_{\mathcal{I}_k}$ for some matrix D_k. If rank of $M_{\mathcal{J}_k}$ is zero then Alice chooses a random index from $\mathcal{J}_k$. Note that $M_{\mathcal{J}_k} = D_k M_{\mathcal{I}_k}$ still holds for an appropriate zero matrix D_k.

Now, there are at least $N/\ell \geq 2\,\mathrm{supp}(\boldsymbol{y}) + 1$ parts $\mathcal{I}_k$. For a majority of indices k (we say that such indices k are "good"), $\boldsymbol{y}_{\mathcal{I}_k}$ is a zero vector. Therefore, in the majority of the cases, Alice obtains the correct values $\boldsymbol{s}_{\mathcal{I}_k}$ by solving the equation $M_{\mathcal{I}_k}\boldsymbol{s} = \boldsymbol{\omega}_{\mathcal{I}_k}$. Since $M_{\mathcal{J}_k}\boldsymbol{s} = D_k M_{\mathcal{I}_k}\boldsymbol{s}$, the value of $\boldsymbol{y}_{\mathcal{J}_k}$ is uniquely determined by $\boldsymbol{s}_{\mathcal{I}_k}$. Moreover, the smallest "good" $k = k_0$ satisfies $k_0 \leq \mathrm{supp}(\boldsymbol{y}) + 1$. The solution $\boldsymbol{s}$ of $M_{\mathcal{I}_{k_0}}\boldsymbol{s} = (\boldsymbol{\omega})_{\mathcal{I}_{k_0}}$ is consistent with the solutions that correspond to other "good" k's, that is, $M_{\mathcal{I}_k} \cdot \boldsymbol{s}_{\mathcal{I}_{k_0}} = \boldsymbol{\omega}_{\mathcal{I}_k}$ for all "good" indices $k > k_0$. Therefore, Alice can find all "good" indices k by majority voting. She also obtains all coordinates of $\boldsymbol{y}_{\mathcal{J}_{k_0}}$. □

If $|\mathcal{I}_{k_0}| = \ell$ then all coordinates of $\boldsymbol{y}$ are revealed, otherwise coefficients are revealed for all sets $|\mathcal{I}_k| \leq |\mathcal{I}_{k_0}|$, as any solution to $M_{\mathcal{I}_{k_0}}\boldsymbol{s} = \boldsymbol{\omega}_{\boldsymbol{\mathcal{I}_{k_0}}}$ uniquely determines $\boldsymbol{y}_{\boldsymbol{\mathcal{J}_{k_0}}} = \boldsymbol{\omega}_{\boldsymbol{\mathcal{J}_{k_0}}} - D_{k_0}\boldsymbol{\omega}_{\boldsymbol{\mathcal{I}_{k_0}}}$. The next result shows that $\boldsymbol{y}$ is revealed almost certainly.

Lemma 2. *Let* $\mathcal{I}_k$ *be defined as in the proof of the previous lemma. Then* $\Pr[\,|\mathcal{I}_k| = |\mathcal{I}_{k+1}|\,] = \prod_{i=0}^{d-1}\left(1 - m^{-|\mathcal{J}_k|+i}\right)$*. Thus, the probability that all coefficients are revealed is approximately* $(1 - m^{-N/2})^{\mathrm{supp}(\boldsymbol{y})\ell} \approx 1 - \mathrm{supp}(\boldsymbol{y})\ell m^{-N/2}$.

Proof. Consider all possible vector assignments of $\boldsymbol{a_1}, \ldots, \boldsymbol{a_\ell}$ that are consistent with the choice of $\mathcal{I}_1, \ldots, \mathcal{I}_k$; that is, such assignments, for which $M_{\mathcal{J}_k} = D'_k M_{\mathcal{I}_k}$ for some D'_k. The latter is equivalent to the assumption that rows of $M_{\mathcal{J}_k}$ are randomly

sampled from a vector space of dimension $|\mathcal{I}_k|$. By a standard result [vLW92, p. 303], the probability that $\mathsf{rank}(M_{\mathcal{J}_k}) = |\mathcal{I}_k|$ is equal to $\prod_{i=0}^{|\mathcal{I}_k|-1}(1 - m^{-|\mathcal{J}_k|+i})$. Hence, the first claim is proven. Now, $\boldsymbol{y}$ is completely determined if $\left|\mathcal{I}_{\mathrm{supp}(\boldsymbol{y})+1}\right| = \ell$. As $|\mathcal{I}_1| = \ell$ by the protocol construction and for $k < \mathrm{supp}(\boldsymbol{y})$, $\left|\mathcal{J}_{\mathrm{supp}(\boldsymbol{y})}\right| > N/2$, the second claim follows from a straightforward calculation. □

If we give more power to Alice, she will be able to do much better. Assume that Protocol 1 is run twice with the same input vector $\boldsymbol{y}$; let $\boldsymbol{a_1}, \ldots, \boldsymbol{a_\ell}$ and $\boldsymbol{a'_1}, \ldots, \boldsymbol{a'_\ell}$ be vectors, computed from the random matrices C and C' as previously. Then, $\boldsymbol{\omega} - \boldsymbol{\omega'} = \sum_{i=1}^{\ell} s_i \boldsymbol{a_i} - \sum_{i=1}^{\ell} s'_i \boldsymbol{a'_i}$. With high probability, this determines $\boldsymbol{s}$ and $\boldsymbol{s'}$ uniquely. To avoid similar attacks, Bob must never run Protocol 1 twice with the same input $\boldsymbol{y}$ but different matrices C. The next lemma shows that also Alice must never run Protocol 1 twice with the same input $\boldsymbol{x}$ but different matrices C.

Lemma 3. *If Protocol 1 is re-executed $k > N/\ell$ times with the same $\boldsymbol{x}$, Bob obtains $\boldsymbol{x}$ with probability higher than $\prod_{i=0}^{N-1}(1 - m^{-k\ell+i})$.*

Proof. Each execution of Protocol 1 provides ℓ linear equations $\boldsymbol{a_i} \cdot \boldsymbol{u} = \boldsymbol{a_i} \cdot \boldsymbol{x} + \boldsymbol{a_i} \cdot C\boldsymbol{p} = \boldsymbol{a_i} \cdot \boldsymbol{x} + t_i$ for $i \in \{1, \ldots, \ell\}$. As $\boldsymbol{a_1}, \ldots, \boldsymbol{a_\ell}$ are chosen randomly, similar argumentation as in Lemma 2 gives the probability estimate. □

Finally, we get another efficient attack when we consider itemsets with almost the same support. For example, assume that Alice knows that $\mathrm{supp}(\boldsymbol{y} - \boldsymbol{y'}) < N/(4\ell) - 1/2$. Then, by using Lemma 1, Alice can determine $\boldsymbol{s}$ and $\boldsymbol{s'}$ from the equation $\boldsymbol{\omega} - \boldsymbol{\omega'} = \boldsymbol{y} - \boldsymbol{y'} + \sum_{i=1}^{\ell} s_i \boldsymbol{a_i} - \sum_{i=1}^{\ell} s'_i \boldsymbol{a'_i}$; therefore, she obtains $\boldsymbol{y}$ and $\boldsymbol{y'}$. This attack works with any choice of C. The condition $\mathrm{supp}(\boldsymbol{y} - \boldsymbol{y'}) \ll N$ is not so rare in the context of frequent itemset mining. Moreover, several optimisations of APRIORI are devised to exploit such shortcuts. To analyse the applicability of low support attacks, we need additional notations. Let $\mathrm{supp}(I)$ denote the support of the itemset I and $\boldsymbol{y_I}$ the corresponding vector, i.e. $y_{I,k} = 1$ iff the kth row contains items I. We say that I is a closed frequent itemset, iff $\mathrm{supp}(I)$ is over frequency threshold and for any proper superset $J \supsetneq I$, $\mathrm{supp}(I) > \mathrm{supp}(J)$. Now, if the frequent itemset I is not closed, then the APRIORI algorithm discovers $J \supset I$ such that $\mathrm{supp}(\boldsymbol{y_I} - \boldsymbol{y_J}) = 0$ and Alice can apply the attack. The ratio ρ between frequent and frequent closed sets describes the average number of vectors revealed by a single closed set. Empirical results [PHM00] on standard data mining benchmarks indicate that ρ can range from 2 to 100 depending on the frequency threshold, when the database contains some highly correlated items.

The analysis can be extended further by using notion of frequent δ-free sets. A itemset I is δ-free iff for any proper subset $J \subsetneq I$, $\mathrm{supp}(\boldsymbol{y_I} - \boldsymbol{y_J}) > \delta$. In other words, if I is not δ-free but frequent, then two vectors $\boldsymbol{y_I}$ and $\boldsymbol{y_J}$ with $\mathrm{supp}(\boldsymbol{y_I} - \boldsymbol{y_J}) \leq \delta$ appear in the APRIORI algorithm. Again, empirical results [BBR03,BB00] on standard data mining benchmarks show that the number of frequent δ-free sets with $\delta \in [0, 20]$ is several magnitudes smaller than the number of frequent sets, when database contain highly correlated items. To conclude, a low support differences are quite common for many practical data sets and thus the Vaidya-Clifton scalar product protocol is insecure for frequent itemset mining.

Remark on [VC02, Section 5.2]. In [VC02, Section 5.2], Vaidya and Clifton note that the fact that x_i and y_i belong to $\{0, 1\}$ can create a disclosure risk. They propose two solutions. The first consists of "*cleverly*" selecting the matrix C so that it is not evident which of the values of x_i and y_i are 1's. Lemma 1 states that such a "*clever*" choice is impossible in general since at least a half of $\boldsymbol{y}$'s coordinates is revealed for every matrix C. Besides, the solution is not fully spelled out and no security proofs are given. Another solution from [VC02, Section 5.2] is said to increase the security of Bob but decrease the security of Alice, but again, no security proofs are given. Thus, it is difficult to estimate the exact security of the proposed solutions. It seems that neither of these mentioned solutions is secure against our attacks.

Communication and computation of Vaidya-Clifton protocol. Alice and Bob must both know C, thus the communication of the Vaidya-Clifton protocol is approximately $N^2 \log m$ bits. In the version of the scalar product protocol where no privacy is guaranteed, Alice just sends her vector (N bits) to Bob, who returns the scalar product ($\lceil \log_2 N \rceil$ bits). Define the communication overhead of a private scalar protocol P to be equal to $C(P)/N$, where $C(P)$ is the number of bits communicated in the protocol P. Thus, the communication overhead of the Vaidya-Clifton private SP protocol is Nm. Computation is dominated by $\Theta(N^2)$ multiplications and additions in $\mathbb{Z}_m$. The new scalar product protocol, that we will propose in this paper, is both more secure and more efficient.

3.2 Du-Atallah Private Scalar Product Protocol

Du and Atallah proposed another private SSP protocol [DA01a], depicted by Protocol 2. We show that also this protocol cannot handle binary vectors with low support.

Since Protocol 2 chooses the values r_i randomly, s_A is a random value and therefore Alice does not learn anything about $\boldsymbol{y}$. To learn $\boldsymbol{x}$, Bob must guess correctly the values ℓ_i for all i. Since the probability of a random guess is p^{-d}, Du and Atallah argue that this protocol is secure when $p^d > 2^{80}$. Bob can do much better, however.

Lemma 4. *Assume $N \geq (2\operatorname{supp}(\boldsymbol{x}) + 1)pd$. Then, with probability 1, Bob finds at least $N/2$ coordinates of $\boldsymbol{x}$ by solving $\operatorname{supp}(\boldsymbol{x}) + 1$ systems of linear equations, each having dimension $pd - 1$. With high probability $\approx (1 - m^{-N/2})^{\operatorname{supp}(\boldsymbol{x})(pd-1)} \approx 1 - \operatorname{supp}(\boldsymbol{x})(pd - 1)m^{-N/2}$, Bob obtains the whole vector $\boldsymbol{x}$.*

Proof. Bob knows that $\sum_{i=1}^{d} \boldsymbol{h_{ij_i}} = \boldsymbol{x}$ for some values j_i. Equivalently,

$$\sum_{i=1}^{d} \sum_{j=1}^{p} c_{ij} \boldsymbol{h_{ij}} = \boldsymbol{x} \ ,$$

where $c_{ij} = 1$ if $j = j_i$ and $c_{ij} = 0$, otherwise. Exactly as Alice did in the proof of Lemma 1, Bob iteratively partitions $[N]$ into subsets $\mathcal{I}_k$ with maximal possible rank. Hence, a solution to $\sum_{i,j} c_{ij} (\boldsymbol{h_{ij}})_{\mathcal{I}_{k_0}} = \boldsymbol{0}$ uniquely determines $\boldsymbol{x}_{\mathcal{I}_k} = \sum_{i,j} c_{ij} (\boldsymbol{h_{ij}})_{\mathcal{I}_k}$ for $k > k_0$. On the other hand, Bob creates at least $2\operatorname{supp}(\boldsymbol{x}) + 1$ partitions $\mathcal{I}_k$. Thus,

PRIVATE INPUTS: Vectors $\boldsymbol{x} \in \{0,1\}^N$ and $\boldsymbol{y} \in \{0,1\}^N$.
PRIVATE OUTPUTS: Shares $s_A + s_B \equiv \boldsymbol{x} \cdot \boldsymbol{y} \mod m$.

1. Alice does:
   Generate random $\boldsymbol{v_1}, \ldots, \boldsymbol{v_{d-1}} \leftarrow \mathbb{Z}_m^N$.
   Set $\boldsymbol{v_d} := \boldsymbol{x} - \sum_{i=1}^{d-1} \boldsymbol{v_i}$ and $s_A := 0$.
2. For $i = 1$ to d do
   (a) Alice does:
   Generate random $\ell_i \in \{1, \ldots, p\}$.
   Set $\boldsymbol{h_{i\ell_i}} := \boldsymbol{v_i}$.
   For $j \in \{1, \ldots, \ell_i - 1, \ell_i + 1, \ldots, p\}$: Generate random $\boldsymbol{h_{ij}} \in \mathbb{Z}_m^n$.
   Send $(\boldsymbol{h_{i1}}, \ldots, \boldsymbol{h_{ip}})$ to Bob.
   (b) Bob does:
   Generate random $r_i \in \mathbb{Z}_m$.
   For $j \in \{1, \ldots, p\}$: Set $z_{ij} := \boldsymbol{h_{ij}} \cdot \boldsymbol{y} + r_i$.
   (c) Alice does:
   Use $\binom{p}{1}$-oblivious transfer to retrieve $z_{i\ell_i}$ from $(z_{i1}, \ldots, z_{ip})$.
   Set $s_A := s_A + z_{i\ell_i}$.
3. Alice outputs s_A, Bob outputs $s_B = -\sum_{i=1}^{d} r_i$.

Protocol 2: Du-Atallah private SSP protocol. Here, $m > N$ is a public modulus

there exists a $k \leq \mathrm{supp}(\boldsymbol{x}) + 1$, such that $\boldsymbol{x}_{\mathcal{I}_k} = \boldsymbol{0}$. As in the proof of Lemma 1, we can determine the first "good" $k_0 \leq \mathrm{supp}(\boldsymbol{x}) + 1$ by using majority voting.

To reduce the amount of computations, Bob can ignore all sets $|\mathcal{I}_k| = pd$. For any "good" k, $|\mathcal{I}_k| \leq pd-1$, as $\boldsymbol{x}_{\mathcal{I}_k} = \boldsymbol{0}$ and the homogeneous system $\sum_{i,j} c_{ij} (\boldsymbol{h_{ij}})_{\mathcal{I}_k} = \boldsymbol{0}$ has a nontrivial solution.

The proof of the second claim is similar to the proof of Lemma 2, since it is sufficient that $pd - 1$ random vectors are linearly independent, and $|\mathcal{I}_1| \geq pd - 1$ by construction. □

This protocol has another serious weakness, since with high probability slightly more than pd coordinates of $\boldsymbol{x}$ allow to determine correct c_{ij} and thus also reveal other coordinates. Therefore, a leakage of pd database entries, can reveal the whole vector (database) and thus pd must be large, say more than 200. On the other hand, this protocol is very inefficient when pd is large.

Communication and computation complexity. Assume $p^d > 2^{80}$. Then the communication of the Du-Atallah private SSP protocol is $dpN + dt_p$, where t_p is the communication complexity of the $\binom{p}{1}$-oblivious transfer protocol. This is minimal when d is maximised, i.e., when $p = 2$. Taking the efficient $\binom{p}{1}$-oblivious transfer protocol from [AIR01], one has $t_2 = 3k$, where $k \approx 1024$ is the security parameter. Then the communication is $2dN + 3dk$ bits for $d \geq 80$ and $k \geq 1024$. Taking $d = 80$ and $k = 1024$, we get communication $160N + 245760$ bits. However, Lemma 4 indicates that for the security of the Du-Atallah protocol, one should pick p and d such that pd

PRIVATE INPUTS: Private vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_\mu^N$.
PRIVATE OUTPUTS: Shares $s_A + s_B \equiv \boldsymbol{x} \cdot \boldsymbol{y} \mod m$

1. Setup phase. Alice does:
   Generate a private and public key pair $(\mathsf{sk}, \mathsf{pk})$.
   Send pk to Bob.
2. Alice does for $i \in \{1, \ldots, N\}$:
   Generate a random new string r_i.
   Send $c_i = \mathsf{Enc}_{\mathsf{pk}}(x_i; r_i)$ to Bob.
3. Bob does:
   Set $w \leftarrow \prod_{i=1}^N c_i^{y_i}$.
   Generate a random plaintext s_B and a random nonce r'.
   Send $w' = w \cdot \mathsf{Enc}_{\mathsf{pk}}(-s_B; r')$ to Alice.
4. Alice does: Compute $s_A = \mathsf{Dec}_{\mathsf{sk}}(w') = \boldsymbol{x} \cdot \boldsymbol{y} - s_B$.

Protocol 3: Private homomorphic SSP protocol

is quite large. For example, picking $p = 2^{11}$ and $d = 8$ might result in an acceptable security level, but then the communication of the protocol will be $2^{14} \cdot N + dt_p$ bits.

4 Cryptographic Private SSP Protocol

In this section we describe a private SSP protocol (Protocol 3) that is based on homomorphic encryption. Note that a private SP protocol can be obtained from it by defining $s_B \leftarrow 0$.

Theorem 1. *Assume that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure homomorphic public-key cryptosystem with $P(\mathsf{sk}) = \mathbb{Z}_m$ for some large m. Set $\mu := \lfloor \sqrt{m/N} \rfloor$. Protocol 3 is a secure SSP protocol in the semi-honest model, assuming that $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_\mu^N$. Alice's privacy is guaranteed when Bob is a probabilistic polynomial-time machine. Bob's privacy is information-theoretical.*

Proof. Clearly, the protocol is correct if the participants are honest. Since the cryptosystem is semantically secure, Bob only sees N random ciphertexts, for which he cannot guess the plaintexts. In particular, this holds even when Bob has given two candidate vectors $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ to Alice and Alice has randomly chosen one of them, $\boldsymbol{x} := \boldsymbol{x_b}$. Even after a polynomial number of protocol executions with Alice's input, Bob will gain only an insignificant amount of information about $\boldsymbol{x_b}$ that will not help him in guessing the value of b. (This roughly corresponds to the standard notion of semantic security.) On the other hand, Alice only sees a random encryption of $s_A = \boldsymbol{x} \cdot \boldsymbol{y} - s_B$, where s_B is random. But Alice has the key anyways, so she can decrypt this message. Thus, Alice obtains no information at all. □

(Note that if $m > 2^{1024}$ and $N \approx 2^{16}$ then $\mu \geq 2^{504}$.) In Appendix A, we describe an extension of this protocol to more than two parties.

Practical considerations. Note that when Alice and Bob need to execute this protocol several times, they can reuse public and private keys and thus the setup phase can be executed only once. Public key cryptography is computationally demanding. To estimate the computational cost of the new scalar product protocol, we must count encryptions, decryptions and multiplications of ciphertexts. Bob must perform N exponentiations and 1 encryption. Alice has to perform N encryptions and 1 decryption.

In the specifically interesting case when $x_i, y_i \in \{0, 1\}$ (e.g., when $\boldsymbol{x}$ and $\boldsymbol{y}$ correspond to characteristic functions of two sets X and Y; then $\boldsymbol{x} \cdot \boldsymbol{y} = |X \cap Y|$), this protocol can be further optimised. Namely, Alice can pre-compute and then store a large table of random encryptions of 0's and 1's. Then every "encryption" just corresponds of fetching a new element from the correct table; this can be done very quickly. Bob has to perform 1 encryption and $\mathrm{supp}(\boldsymbol{y})$ multiplications, since the exponents y_i are all Boolean. (When $y_i = 0$ then $c_i^{y_i} = 1$ and otherwise $c_i^{y_i} = c_i$.)

The current hardware allows to do approximately 10^5 multiplications per seconds and thus the computational complexity of both Alice and Bob is tolerable. A similar analysis applies for Protocol 4. Here, Alice and Bob must pre-compute N encryptions. Hence, we can conclude that the computational complexity is not a serious downside of the proposed protocols. Similar, although not as efficient, optimisation tricks can also be used to speed up Protocol 3 when $\boldsymbol{x}$ and $\boldsymbol{y}$ are not binary.

Estimated communication complexity. The only serious drawback of the new protocols is the communication overhead: since Alice sends N ciphertexts c_i, the overhead is k'/μ, where k' is just the size of each ciphertext in bits. When using any of the currently known most efficient semantically secure homomorphic cryptosystems (e.g., the one from [Pai99]), $k' \approx 2048$. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_{m'}$ with very small m'—say, $m' \leq 13$, this compares non-favourably with the overhead of the (insecure) Du-Atallah protocol which has the overhead of approximately 160 times with $d = 80$ and $k = 1024$. For a large m', the described protocol is already more communication-efficient than the Du-Atallah protocol.

Comparison with Freedman-Nissim-Pinkas protocol. Recently, Freedman, Nissim and Pinkas proposed a related cryptographically secure protocol for computing the set intersection cardinality [FNP04], a task that is equivalent to privately computing the scalar product of two binary vectors. In the non-shared case, the Freedman-Nissim-Pinkas protocol is more efficient than the new one, but then the participants also learn the values $\mathrm{supp}(\boldsymbol{x})$ and $\mathrm{supp}(\boldsymbol{y})$. However, recall that in the data mining applications it is preferable that both parties will get only shares $s_A + s_B = \boldsymbol{x} \cdot \boldsymbol{y} \mod m$ of the scalar product, otherwise frequency of some infrequent sets is revealed. Moreover, sometimes only a list of frequent sets without frequencies is required.

Freedman, Nissim and Pinkas proposed also a solution for shared version, but their protocol requires a secure circuit evaluation. Briefly, secure evaluation means that first Alice and Bob obtain $\mathrm{supp}(\boldsymbol{x})$ different shares

$$s_i + t_i = \left\{ \begin{array}{l} 0, \text{ if } x_i = 1, y_i = 1 \\ r_i, \text{ if } x_i = 1, y_i = 0 \end{array} \right\} \mod m$$

where $r_i \in \mathbb{Z}_m$ is a random value and m is (say) a 1024-bit number. To securely compute $\boldsymbol{x} \cdot \boldsymbol{y}$ by secure circuit evaluation, one therefore needs to execute oblivious transfer for each $1024 \cdot \mathrm{supp}(\boldsymbol{x})$ input bit pairs (s_i, t_i). Since a $\binom{2}{1}$-oblivious transfer protocol requires sending at least three encryptions, the communication overhead of the Freedman-Nissim-Pinkas protocol is lower than the communication overhead of Protocol 3 only if $\mathrm{supp}(\boldsymbol{x}) \leq N/(3 \cdot 1024)$, i.e., if the candidate set is very infrequent.

Reducing communication overhead. We shall now discuss how to reduce the overhead if it is known that $\boldsymbol{x}$ and $\boldsymbol{y}$ are Boolean. Again, similar optimisation techniques can be used when $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_{\mu'}$ for some $2 < \mu' \ll \mu$. In the following we assume that the plaintext space of the cryptosystem Π is a residue ring $\mathbb{Z}_m$ such that $\log m \geq 1024$. This is the case for all widely known homomorphic cryptosystems. When we assume that $x_i, y_i \in \{0, 1\}$, every ciphertext c_i in Protocol 3 only transfers a single bit x_i, which results in communication overhead.

The next technique for packing several bits into one plaintext is fairly standard in cryptography (it has been used at least in the context of electronic voting [CGS97,DJ01], electronic auctions [LAN02] and oblivious transfer [Lip04]). To pack k entries into a single message—recall that the plaintext length is k bits—, we fix a radix $B > N$, such that $B^k < m$, and work implicitly with B-ary numbers. Let $[v_k, \ldots, v_2, v_1] = v_1 + v_2 B + \cdots + v_k B^{k-1}$. Our method works only in the case when Alice and Bob do patch computation of scalar products, more precisely, when Alice and Bob need to compute $\boldsymbol{x_i} \cdot \boldsymbol{y}$ for several vectors $\boldsymbol{x_i}$, $i \in \{1, \ldots, k\}$, owned by Alice. (This is exactly what happens in the context of frequent itemset mining.)

The new patch scalar product protocol looks exactly like Protocol 3, except that Alice computes c_i as

$$\begin{aligned} c_i =& \mathsf{Enc}_{\mathsf{pk}}([x_{ki}, \ldots, x_{2i}, x_{1i}]; r_i) \\ =& \mathsf{Enc}_{\mathsf{pk}}(1;0)^{x_{1i}} \mathsf{Enc}_{\mathsf{pk}}(B;0)^{x_{2i}} \cdot \cdots \cdot \mathsf{Enc}_{\mathsf{pk}}(B^{k-1};0)^{x_{ik}} \mathsf{Enc}_{\mathsf{pk}}(0;r_i) \ . \end{aligned}$$

It takes at most k multiplications to compute c_i. Again, the encryptions $\mathsf{Enc}_{\mathsf{pk}}(B^j; 0)$ can be computed in the setup phase. Hence, during the first step, Alice's computation is N encryptions and $O(kN)$ multiplications.

At the second step of the protocol, Bob computes

$$\begin{aligned} w &= \prod_{i=1}^{N} \mathsf{Enc}_{\mathsf{pk}}(y_i[x_{ki}, \ldots, x_{2i}, x_{1i}]; r_i) \mathsf{Enc}_{\mathsf{pk}}(-s_B, r') \\ &= \mathsf{Enc}_{\mathsf{pk}}\left([\boldsymbol{x_k} \cdot \boldsymbol{y}, \ldots, \boldsymbol{x_1} \cdot \boldsymbol{y}] - s_B; r''\right) \ . \end{aligned}$$

Hence, if Bob reveals s_B, Alice can restore *all* scalar products $\boldsymbol{x_j} \cdot \boldsymbol{y}$. Sometimes it is also needed that Alice be able only to compute $\boldsymbol{x_j} \cdot \boldsymbol{y}$ for $j \in I$, where I is a proper subset of $\{1, \ldots, k\}$. One can do this efficiently by using standard cryptographic techniques.

Therefore, when using the Paillier cryptosystem, the resulting protocol for privately computing the scalar product of two binary vectors has almost optimal communication

overhead of $\lceil \log N \rceil$ times. (When using the Damgård-Jurik cryptosystem, the communication overhead might even be smaller.) This should be compared to the 160 times overhead of the insecure Du-Atallah protocol.

Security in malicious model. Protocol 3 can be made in the malicious model by letting Alice to prove in zero-knowledge, for every i, that c_i encrypts a value from $\mathbb{Z}_\mu$. This can be done efficiently in the random oracle (or common reference string) model [Lip03]. An alternative is to use conditional disclosure of secrets [AIR01] modified recently to the setting of Paillier's cryptosystem in [LL04a]. Both methods guarantee that at the end of a protocol run, Alice is no better of than mounting the next *probing attack*: Alice creates a suitable valid input vector $\boldsymbol{x'}$, executes the protocol with Bob, and obtains $\boldsymbol{x'} \cdot \boldsymbol{y}$. If $\boldsymbol{x'}$ is suitably chosen (e.g., $x'_i = 1$ and $x'_j = 0$ for $j \neq i$), this may result in privacy leakage. However, this probing attack is unavoidable, no matter what private scalar product protocol is used instead of Protocol 3. The only way to tackle this attack is to let Alice to prove that her input $\boldsymbol{x}$ is "correctly" computed, whatever "correctly" means in the concrete application (e.g., in frequent itemset mining on vertically distributed databases). While such a functionality can be added to Protocol 3, it is not a part of the definition of a "private scalar product" protocol, but highly application-dependent (and thus should be left to be specified on a higher level), and very often, highly costly.

5 Conclusions

The secure computation of a scalar product is an important task within many data mining algorithms that require the preservation of privacy. Recently, several protocols have been proposed to solve this task. We have shown, however, that they are insecure. Moreover, we presented a private scalar product protocol based on standard cryptographic techniques and proved that it is secure. Furthermore, we described several optimisations in order to make it very efficient in practice.

Acknowledgements. We would like to thank Benny Pinkas for useful comments. This work was partially supported by the Finnish Defence Forces Institute for Technological Research and by the Finnish Academy of Sciences.

References

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In Birgit Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, 6–10 May 2001. Springer-Verlag.

[AMS+96] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast Discovery of Association Rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI/MIT Press, 1996.

[BB00] Jean-Franois Boulicaut and Artur Bykowski. Frequent Closures as a Concise Representation for Binary Data Mining. In *PADKK 2000*, volume 1805 of *Lecture Notes in Computer Science*, pages 62–73. Springer, 2000.

[BBR03] Jean-Franois Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-Sets: A Condensed Representation of Boolean Data for the Approximation of Frequency Queries. *Data Mining and Knowledge Discovery*, 7(1):5–22, 2003.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118, Konstanz, Germany, 11–15 May 1997. Springer-Verlag.

[DA01a] Wenliang Du and Mikhail J. Atallah. Privacy-Preserving Statistical Analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, pages 102–110, New Orleans, Louisiana, USA, December 10–14 2001.

[DA01b] Wenliang Du and Mikhail J. Atallah. *Protocols for Secure Remote Database Access with Approximate Matching*, volume 2 of *Advances in Information Security*, page 192. Kluwer Academic Publishers, Boston, 2001. http://www.wkap.nl/prod/b/0-7923-7399-5.

[DJ01] Ivan Damgård and Mads Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In Kwangjo Kim, editor, *Public Key Cryptography 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136, Cheju Island, Korea, 13–15 February 2001. Springer-Verlag.

[DZ02] Wenliang Du and Zhijun Zhan. A Practical Approach to Solve Secure Multi-party Computation Problems. In Carla Marceau and Simon Foley, editors, *Proceedings of New Security Paradigms Workshop*, pages 127–135, Virginia Beach, virginia, USA, September 23–26 2002. ACM Press.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient Private Matching and Set Intersection. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19, Interlaken, Switzerland, 2–6 May 2004. Springer-Verlag.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[LAN02] Helger Lipmaa, N. Asokan, and Valtteri Niemi. Secure Vickrey Auctions without Threshold Trust. In Matt Blaze, editor, *Financial Cryptography — Sixth International Conference*, volume 2357 of *Lecture Notes in Computer Science*, pages 87–101, Southhampton Beach, Bermuda, 11–14 March 2002. Springer-Verlag.

[Lip03] Helger Lipmaa. On Diophantine Complexity and Statistical Zero-Knowledge Arguments. In Chi Sung Laih, editor, *Advances on Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 398–415, Taipei, Taiwan, 30 November–4 December 2003. Springer-Verlag.

[Lip04] Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Total Communication. Technical Report 2004/063, International Association for Cryptologic Research, February 25 2004.

[LL04a] Sven Laur and Helger Lipmaa. Additive Conditional Disclosure of Secrets Or: Good-Bye, Random Oracles. Manuscript, November 2004.

[LL04b] Sven Laur and Helger Lipmaa. On Private Similarity Search Protocols. In Sanna Limatainen and Teemupekka Virtanen, editors, *Proceedings of the Ninth Nordic Workshop on Secure IT Systems (NordSec 2004)*, pages 73–77, Espoo, Finland, November 4–5, 2004.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*,

volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, 2–6 May 1999. Springer-Verlag.

[PHM00] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000.

[Pin02] Benny Pinkas. Cryptographic Techniques for Privacy-Preserving Data Mining. *KDD Explorations*, 4(2):12–19, 2002.

[VC02] Jaideep Vaidya and Chris Clifton. Privacy Preserving Association Rule Mining in Vertically Partitioned Data. In *Proceedings of The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 23–26 2002. ACM.

[vLW92] Jacobus H. van Lint and Richard M. Wilson. *A Cource in Combinatorics*. Cambridge University Press, 1992.

A Private Generalised Scalar Product Protocol

Next, we propose a secure generalised scalar product protocol (Protocol 4) for

$$\langle \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_k} \rangle = \sum_{i=1}^{N} x_{1i} \cdots x_{ki} \ .$$

For the sake of simplicity, we consider only the three-party case but the protocol can be easily generalised. Again, Alice has a private key; Bob and Carol know only the corresponding public key. The security of the generalised scalar product protocol depends on Alice. Namely, when Alice colludes with other parties then privacy can be compromised. For example, colluding Alice and Carol can reveal y_i, unless $x_i = 0$, since $\mathsf{Dec}_{\mathsf{sk}}(d_i) = x_i y_i$. Thus, we get the following result.

Theorem 2. *Assume that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure homomorphic public-key cryptosystem with $P(\mathsf{sk}) = \mathbb{Z}_m$ for some large m. Protocol 4 is a secure generalised scalar product protocol. In particular, it is secure against all possible coalitions provided that Alice does collude with other parties.*

The proof is a simple generalisation of the previous proof. Bob must re-randomise c_i's as $d_i = c_i \cdot \mathsf{Enc}_{\mathsf{pk}}(0; r_i')$, since otherwise the values of y_i's can be detected only by comparing the ciphertext that he receives from Alice with the one he sends to Carol. The sharing step 4 allows combine the outcome with other cryptographic protocols.

The assumption that Alice does not collude with other parties is quite strong. When we modify the protocol so that $(\mathsf{sk}, \mathsf{pk})$ is generated jointly by Alice, Bob and Carol and that on the step 4, they do threshold decryption of w, we get a private SP protocol with the next security result:

Theorem 3. *Assume $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure homomorphic threshold public-key cryptosystem. Then Protocol 4, generalised to κ parties, is secure against coalitions by $< \kappa/2$ parties.*

PRIVATE INPUTS: Private vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{Z}_{\mu}^{N}$.
PRIVATE OUTPUTS: Shares $s_A + s_B + s_C \equiv \langle \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \rangle \mod m$

1. Alice does:
   Generate a key-pair $(\mathsf{sk}, \mathsf{pk})$.
   Send the public key pk to Bob and Carol.
2. Alice does for $i \in \{1, \ldots, N\}$:
   Send $c_i = \mathsf{Enc}_{\mathsf{pk}}(x_i; r_i)$ to Bob.
3. Bob does for $i \in \{1, \ldots, N\}$:
   Set $d_i = c_i^{y_i} \mathsf{Enc}_{\mathsf{pk}}(0; r_i')$.
   Send d_i to Carol.
4. Carol does:
   Set $w \leftarrow \prod_{i=1}^{N} c_i^{z_i}$.
   Generate a random plaintext s_C and a random nonce r'.
   Send $w' \leftarrow w \cdot \mathsf{Enc}_{\mathsf{pk}}(-s_C; r')$ to Bob.
5. Bob does:
   Generate a random plaintext s_B and a random nonce r''.
   Send $w'' \leftarrow w' \cdot \mathsf{Enc}_{\mathsf{pk}}(-s_B; r'')$ to Alice.
6. Alice computes $s_A \leftarrow \mathsf{Dec}_{\mathsf{sk}}(w'') = \boldsymbol{x} \cdot \boldsymbol{y} - s_B - s_C$.

Protocol 4: Private generalised homomorphic SSP protocol