A Weakness in Some Oblivious Transfer and Zero-Knowledge Protocols

Ventzislav Nikov[1], Svetla Nikova[2], and Bart Preneel[2]

[1] venci.nikov@gmail.com
[2] Department Electrical Engineering, ESAT/COSIC,
Katholieke Universiteit Leuven, Kasteelpark Arenberg 10,
B-3001 Heverlee-Leuven, Belgium
svetla.nikova, bart.preneel@esat.kuleuven.be

Abstract. We consider oblivious transfer protocols and their applications that use underneath semantically secure homomorphic encryption scheme (e.g. Paillier's). We show that some oblivious transfer protocols and their derivatives such as private matching, oblivious polynomial evaluation and private shared scalar product could be subject to an attack. The same attack can be applied to some non-interactive zero-knowledge arguments which use homomorphic encryption schemes underneath. The roots of our attack lie in the additional property that some semantically secure encryption schemes possess, namely, the decryption also reveals the random coin used for the encryption, and that the (sender's or prover's) inputs may belong to a space, that is very small compared to the plaintext space. In this case it appears that even a semi-honest chooser (verifier) can derive from the random coin bounds for all or some of the sender's (prover's) private inputs with non-negligible probability. We propose a fix which precludes the attacks.

Keywords: Oblivious Transfer, Homomorphic Semantically Secure Cryptosystems, Paillier's Public-Key Cryptosystem, Non-Interactive Zero-Knowledge Arguments.

1 Introduction

Oblivious Transfer (OT) [4, 30] protocols allow one party, called *the sender* to send part of its inputs to a second party, called *chooser*, in such a manner that the chooser does not receive more information than it is entitled and the sender does not learn which part of the inputs the chooser received. Oblivious transfer is used as a key component in many applications of cryptography.

Naor and Pinkas [26] proposed a way to use OT for *polynomial evaluation*. Another application known as *private matching* solves the problem of two parties who possess lists of items and want to compute their set-intersection or to approximate the size of the intersection. Freedman *et al.* [16] have shown that a simple reduction from oblivious transfer to private matching exists. The authors of [16] used oblivious polynomial evaluation in their solution for the private matching set intersection problem.

In this paper we will work in the *semi-honest security model*, in which the parties follow the protocol, but may be curious. We do not consider malicious parties who may deviate from the protocol. Often, there is no guarantee for the privacy of the sender if the chooser is malicious, but we do not consider this issue.

Our Contribution: We first describe an attack against an oblivious transfer protocol; subsequently we apply the attack to certain protocols derived from oblivious transfer, such as oblivious polynomial evaluation, private matching (set cardinality and subset inclusion) and private (shared) scalar product. For our attack we exploit the additional property that some semantically secure encryption schemes possess, namely that the decryption reveals the random coin used for encryption. We consider the case when the (sender's) inputs belong to a very small space compared to the plaintext space of the Paillier cryptosystem. We show that from the random coin the chooser can derive certain information (bounds) for all (or some) of the sender's private inputs with non-negligible probability. We extend the attack to certain non-interactive zero-knowledge protocols. We introduce the so-called irrational behavior of the chooser, meaning that a semi-honest but curious chooser is "bluffing" in order to get the sender's inputs, i.e. the chooser is putting his privacy at risk. To the best of our knowledge some of the protocols from the following papers [6, 12, 17, 19, 31] could be subject to this attack when applied in this scenario. Finally we propose a fix which precludes the attacks.

Organization of the paper: In the next section we introduce the notions of homomorphic semantically secure cryptosystems, oblivious transfer, and different applications of the oblivious transfer. Section 3 provides description of several known protocols and in Sect. 4 the attack against them is proposed. We conclude in Sect. 5.

2 Preliminary

Homomorphic Semantically Secure Cryptosystems.
Let $\Pi = (G_\Pi, E, D)$ be a public-key encryption cryptosystem, where G_Π is the key generation algorithm, E is the encryption algorithm and D is the decryption algorithm. Let k be the security parameter, then the key generation algorithm G_Π on input 1^k generates a valid key pair (SK, K) of private and public keys that corresponds to the security parameter k. The encryption algorithm E takes as input a plaintext m, a random coin r and the public key K and outputs the corresponding ciphertext $E_K(m, r)$. The decryption algorithm takes as input a ciphertext c and the private key SK and outputs the corresponding plaintext $D_{SK}(c)$. More formal: $G_\Pi : 1^k \mapsto (SK, K)$; $E_K : (m, r) \mapsto E_K(m, r)$, $D_{SK} : c \mapsto D_{SK}(c)$ and $D_{SK}(c) = m$ if $c = E_K(m, r)$. It is required that $D_{SK}(E_K(m, r)) = m$ for any random coin r, key pair (SK, K) and plaintext m. It is said that Π is *homomorphic*, if $E_K(m_1, r_1) \cdot E_K(m_2, r_2) = E_K(m_1 + m_2, r_1 \cdot r_2)$. It then follows that $E_K(m, r)^s = E_K(s \cdot m, r^s)$.

For an algorithm A, define $Adv^{sem}_{\Pi,k}(A)$ to be the advantage that A has over random guessing when trying to distinguish random encryption of two elements, chosen by herself. It is said that Π is *semantically* secure under an chosen plaintext attack (IND-CPA secure) if for all PPT (probabilistic polynomial time) algorithms A, the advantage $Adv^{sem}_{\Pi,k}(A)$ is negligible in k.

Several homomorphic probabilistic encryption schemes are known: ElGamal [14], Goldwasser and Micali [18], Benaloh [2], Okamoto and Uchiyama [28], Naccache and Stern [25], Paillier [29] and its modifications [5, 13].

For the sake of simplicity, we will describe the protocols with the Paillier cryptosystem (some of the protocols which we consider are indeed designed for the Paillier cryptosystem), although most of the homomorphic semantically secure cryptosystems can be used instead of Paillier's. We present the Paillier cryptosystem for completeness, but omit the number-theoretic justifications.

Key Generation: let N be an RSA modulus $N = pq$, where p, q are large primes. The public key K is N and the secret key SK is $\lambda(N) = lcm((p-1),(q-1))$, where $\lambda(N)$ is the Carmichael function. One can assume w.l.o.g. that $N > 2^k$, where the security parameter $k \geq 1024$.

Encryption: to encrypt a plaintext $m \in \mathbb{Z}_N$, compute the ciphertext

$$c = E_K(m,r) = (1+mN)r^N \bmod N^2, \text{ with } r \in_R \mathbb{Z}_N^*.$$

Decryption: to decrypt a ciphertext $c \in \mathbb{Z}_{N^2}$, compute the plaintext

$$m = D_{SK}(c) = \frac{L(c^{\lambda(N)} \bmod N^2)}{\lambda(N)} \bmod N, \text{ where } L(u) = \frac{u-1}{N}.$$

The Paillier cryptosystem possesses the following useful properties:

$$\begin{aligned} E_K(m_1,r_1)E_K(m_2,r_2) \bmod N^2 &= E_K(m_1+m_2 \bmod N, r_1r_2 \bmod N) \\ E_K(m,r)^s \bmod N^2 &= E_K(sm \bmod N, r^s \bmod N) \\ E_K(m,r)(1+N)^c \bmod N^2 &= E_K(m+c \bmod N, r). \end{aligned}$$

In order to re-randomize a ciphertext $c = E_K(m,r)$, simply multiply it by a random encryption of 0, i.e. compute $cr_1^N \bmod N^2 = E_K(m, rr_1 \bmod N)$ for $r_1 \in_R \mathbb{Z}_N^*$.

It is well known (see [5]) that for Paillier's cryptosystem $D_{SK}(c) = (m,r)$ if $c = E_K(m,r)$, i.e. the result of the decryption of a ciphertext is the corresponding plaintext and the random coin used for the encryption (usually the random coin cannot be recovered efficiently). Indeed as Catalano *et al.* have shown there is an alternative decryption process based on the observation that the ciphertext $c = E_K(m,r)$ satisfies $c = r^N \bmod N$. The latter can disclose r by an RSA decryption (modulo N, with public exponent N). Now putting r in the original ciphertext equation provides the plaintext m.

We stress here that the ability to efficiently disclose the random coin used for the encryption, forms an essential point for our attack. We pose as an open problem whether our attack can be extended to some of the other homomorphic semantically secure cryptosystems.

2.1 $\binom{n}{1}$-Oblivious Transfer and Zero-Knowledge Arguments

During an $\binom{n}{1}$-Oblivious Transfer the *sender* maintains n items and the *chooser* receives one item chosen by him. The sender does not know which item was transferred. The security of an OT is usually defined in two parts. We will follow the definitions of [22, 27]. Let $\tilde{k}$ be the security parameter.

Chooser-Privacy: Consider an algorithm A that executes the sender's part of the OT protocol; define $Adv^{OT}_{Cho,\tilde{k}}(A)$ to be the probability that after observing an execution of the protocol, A can predict which choice was made by the chooser. An OT protocol is said to be (computationally) chooser-private if $Adv^{OT}_{Cho,\tilde{k}}(A)$ is negligible for any PPT algorithm A. In all this protocols the chooser-privacy (which holds even against a malicious sender) will be based on the indistinguishability implied by the underlying semantically secure encryption scheme.

Sender-Privacy: Consider an algorithm A executing the chooser's part of the OT protocol; define a simulator S that generates an output that is statistically indistinguishable from the view of A that interacts with the honest sender. More precisely, for an algorithm S define $Adv^{OT}_{Sen,\tilde{k}}(A, S)$ to be the statistical difference of the distributions of the S output and the view of A. An OT protocol is called (statistically) sender-private if for every (not necessarily PPT) A there exists a (not necessarily PPT) S, such that $Adv^{OT}_{Sen,\tilde{k}}(A, S)$ is negligible in $\tilde{k}$. The sender-privacy is called *perfect* if $Adv^{OT}_{Sen,\tilde{k}}(A, S) = 0$. In all this cases the sender-privacy is based on a comparison with the ideal model.

Recently Damgård *et al.* [12] have proposed a method to build non-interactive zero-knowledge protocols from homomorphic encryption. Namely the authors described a method for compiling a class of Σ-protocols (3-move public-coin protocols) into non-interactive zero-knowledge arguments. In a zero-knowledge proof system a *prover* convinces a *verifier* via an interactive protocol that some statement is true. The verifier should learn nothing beyond the fact that the assumption is valid. Σ-protocols are three-move protocols where conversations are tuples of the form (a, e, z) where e is a random challenge sent by the verifier, a is the prover's input and z is the proof. There are several well-known techniques for making Σ-protocols non-interactive [11, 15].

2.2 Applications of OT

As shown by Kilian [21] most cryptographic protocols can be based on oblivious transfer. In this section we will describe several protocols built on top of OT.

An $\binom{n}{1}$-OT protocol sometimes needs to be *sender-verifiable* (or *committed*) [7, 10] in the following sense: the sender commits to every item and sends these commitments to the chooser; these commitments later can be used in various zero-knowledge proofs and arguments.

The notion of *conditional oblivious transfer* (COT) was introduced by Di Crescenzo *et al.* [9]. It is a variant of OT in which the two participants have private inputs, say x and y respectively, and share a public predicate $Q(\cdot, \cdot)$. The sender has a secret s, which is transferred to the chooser if and only if $Q(x, y) = 1$. If $Q(x, y) = 0$, no information about s is transferred to the chooser. The chooser's private input and the value of the predicate remain computationally hidden from the sender.

The notion of *strong conditional oblivious transfer* (SCOT) has been first introduced by Di Crescenzo [8]; later Blake and Kolesnikov [3] have independently defined the same notion. SCOT strengthens the COT definition, in the SCOT setting – unlike the COT "all-or-nothing" approach – the sender possesses two secrets s_0 and s_1 and transfers s_i if $Q(x, y) = i$ (where $i = 0$ or 1). In addition to the COT requirement that the chooser private input has to be computationally hidden from the sender, the value of the predicate should also remain hidden for both participants.

Consider the following problem: two parties possess lists (sets) of items and they want to compute their set-intersection. Related problems are to approximate the size of the intersection or to decide whether the intersection size is greater than a threshold. Such problems are called *private matching* (PM) in [16]. That is, if the chooser inputs $X = \{x_1, \ldots, x_{k_c}\}$ and the sender inputs $Y = \{y_1, \ldots, y_{k_s}\}$ then the chooser learns $X \cap Y = \{x_u : \exists v, x_u = y_v\} \leftarrow PM(X, Y)$. The related variants are as follows: the chooser learns $|X \cap Y| \leftarrow PM_C(X, Y)$ for the *intersection size* problem or for the *threshold intersection size* problem he gets $1 \leftarrow PM_t(X, Y)$ if $PM_C(X, Y) > t$ and 0 otherwise. As shown by Freedman *et al.* [16] a simple reduction from oblivious transfer to private matching exists.

In a simpler form of PM both lists contain just one item, thus the two parties want to compare their private inputs without leaking it. *Private equality test* (PET) allows the chooser to know whether his private input and the sender's private input are equal [16, 22].

Another kind of PM is the *private subset inclusion.* Namely, both participants have sets X and Y as inputs and the chooser gets 0 if $X \subseteq Y$ or 1 otherwise. Laur *et al.* [24] have proposed a private subset inclusion protocol, based on an improvement of the intersection size protocol by Freedman *et al.* [16].

Naor and Pinkas [26] proposed a way to use OT for *polynomial evaluation* (OPE). Freedman *et al.* [16] used OPE in their solution for the PM set intersection problem. Recently Freedman *et al.* [17] proposed another OPE protocol which is used as a building block for a *keyword search* protocol.

A protocol between two parties is called a *scalar product* (SP) protocol when on private inputs of both parties $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ it outputs their scalar product $<\mathbf{x}, \mathbf{y}>= \sum_{i=1}^{n} x_i y_i$. A protocol is called a *shared scalar product* (SSP) protocol [19] when both parties receive as output of the protocol uniformly distributed additive shares of the scalar product, i.e., the chooser gets $s_c \in \mathbb{Z}_N$ and the sender gets $s_s \in \mathbb{Z}_N$ such that $s_c + s_s =<\mathbf{x}, \mathbf{y}> \bmod N$. These protocols are called *private* if the inputs (i.e. $\mathbf{x}$ and $\mathbf{y}$) are not disclosed.

3 The Protocols

This section describes the protocols to which our attacks can be applied. The reader who is familiar with these protocols can skip this section and continue with the attack described in Sect. 4.

Consider the standard OT setting, i.e. the chooser and the sender have their private inputs. The chooser encrypts his input and sends it to the sender. The sender applies a transformation to the encrypted chooser's input and to his own input (which could be also encrypted). The value obtained in this way is returned to the chooser.

3.1 $\binom{n}{1}$-Oblivious Transfer

We will start with a short description of Homomorphic Oblivious Transfer and the AIR protocol [1, 22].
Private Inputs:

- The sender has a vector $\mu = (\mu_1, \ldots, \mu_n)$, $\mu_i \in \mathbb{Z}_T$ and $T \leq N$.
- The chooser has made a choice $\sigma \in \{1, \ldots, n\}$.

Private Output: The chooser gets μ_σ.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. Then generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c \leftarrow E_K(\sigma, r)$. He sends K and c to the sender.
2. For $i = 1, \ldots, n$ the sender performs the following: generates random coins $r_i, s_i \in_R \mathbb{Z}_N^*$ and computes $c_i \leftarrow E_K(\mu_i, 1)\,(c\,E_K(-i, 1))^{s_i}\,E_K(0, r_i) \bmod N^2$. He sends $c_1, \ldots, c_n$ to the chooser.
3. The chooser obtains $\mu_\sigma \leftarrow D_{SK}(c_\sigma)$.

Homomorphic $\binom{n}{1}$-Oblivious Transfer:
Aiello *et al.* [1] have proposed an OT protocol, which provides perfect sender-privacy and computational chooser-privacy (AIR protocol, in short). This protocol has been slightly modified and generalized by Lipmaa [22] to a *homomorphic oblivious transfer* (HOT) protocol. In [23] the authors fix some problems with the scheme from [22] .

Since the encryption scheme is semantically secure, the sender cannot derive σ from the ciphertext c (step 1), which guarantees the chooser-privacy. Using the homomorphic property of the encryption scheme it is easy to verify that in step 2 the sender computes $c_i \leftarrow E_K(\mu_i + (\sigma - i)s_i \bmod N,\ r_i\ r^{s_i} \bmod N)$. Then in step 3 the chooser can obtain $\mu_i + (\sigma - i)s_i \bmod N$. But since the s_i are random coins, the values μ_i are perfectly hidden, except μ_σ. This guarantees the correctness of the scheme and the sender-privacy. The HOT protocol is further used in [22] to build committed OT and PET protocols.

Stern's $\binom{n}{1}$-Oblivious Transfer:
Now we present the OT protocol proposed by Stern [31]; this protocol has later

been rediscovered by Chang [6]. The original protocol uses a homomorphic semantically secure encryption scheme and a homomorphic commitment scheme. The Paillier encryption scheme, proposed one year after the publication of [31], is not used in the original scheme.
Private Inputs:

- The sender has a vector $\mu = (\mu_1, \ldots, \mu_n)$, $\mu_i \in \mathbb{Z}_T$ and $T \leq N$.
- The chooser has made a choice $\sigma \in \{1, \ldots, n\}$.

Private Output: The chooser gets μ_σ.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. He chooses an n-tuple $(x_1, \ldots, x_n)$ such that $x_\sigma = 1$ and $x_i = 0$ for $i \neq \sigma$. Then generates n random coins $r_i \in_R \mathbb{Z}_N^*$ and computes $c_i \leftarrow E_K(x_i, r_i)$ for $i = 1, \ldots, n$. He sends K and $c_1, \ldots, c_n$ to the sender. Last he provides zero-knowledge proofs that all x_i except one are equal to 0 and the nonzero one is equal to 1.
2. The sender generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c \leftarrow (\prod_{i=1}^n c_i^{\mu_i})\ E_K(0, r) \bmod N^2$. He sends c to the chooser.
3. The chooser obtains $\mu_\sigma \leftarrow D_{SK}(c)$.

Using the homomorphic property of the encryption scheme it is easy to verify that in step 2 the sender computes $c \leftarrow E_K(\sum_{i=1}^n \mu_i x_i \bmod N,\ r \prod_{i=1}^n r_i^{\mu_i} \bmod N)$. Then in step 3 the chooser can obtain $\overline{\mu} = \sum_{i=1}^n \mu_i x_i \bmod N$. But since $(x_1, \ldots, x_n)$ is such that $x_\sigma = 1$ and $x_i = 0$ for $i \neq \sigma$ the decrypted value is $\overline{\mu} = \mu_\sigma$.

Note that in both OT protocols [31] and [1, 22] the sender uses an encryption of 0 (step 2) to re-randomize the ciphertext.

3.2 Oblivious Polynomial Evaluation

Recall that oblivious polynomial evaluation protocol is a building block for other more complex protocols, for example private matching. The protocol given by Freedman *et al.* [17] can be described as follows.
Private Inputs:

- The chooser input is a value $\overline{x} \in \mathbb{Z}_T$.
- The sender input is a polynomial $P(x) = \sum_{i=0}^n a_i x^i$, $a_i \in \mathbb{Z}_T$.
- T is chosen such that $\max(|P(x)|) \leq N$.

Private Output: The chooser gets $P(\overline{x})$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. Then he generates random coins $r_j \in_R \mathbb{Z}_N^*$ and computes $c_j \leftarrow E_K(\overline{x}^j, r_j)$ for $j = 1, \ldots, n$. The chooser sends K and $c_1, \ldots, c_n$ to the sender.
2. The sender generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c = E_K(a_0, r)(\prod_{j=1}^n c_j^{a_j}) \bmod N^2$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts, i.e. he computes $z = D_{SK}(c)$.

Observe that $c = E_K(\sum_{j=0}^n a_j \overline{x}^j \bmod N,\ r \prod_{j=1}^n r_j^{a_j} \bmod N)$, thus $z = \sum_{j=0}^n a_j \overline{x}^j \bmod N$, i.e. $z = P(\overline{x})$. Note that the sender re-randomizes the ciphertext (step 2) in a slightly non standard way – by encrypting a_0 with a random r instead of encrypting 0 afterwards.

3.3 Private Shared Scalar Product

In [19] Goethals *et al.* proposed a private SSP protocol. As pointed out by the authors of [19] a private SP can be obtained immediately from the private SSP protocol by defining $s_s \leftarrow 0$. We present here the private SSP protocol.
Private Inputs:

– The chooser input is a vector $\mathbf{x} = (x_1, \ldots, x_n)$, $x_i \in \mathbb{Z}_T$ and $T \leq \lfloor \sqrt{N/n} \rfloor$.
– The sender input is a vector $\mathbf{y} = (y_1, \ldots, y_n)$, $y_i \in \mathbb{Z}_T$.

Private Output:

– The chooser gets a share $s_c \in \mathbb{Z}_N$.
– The sender gets a share $s_s \in \mathbb{Z}_N$.
– Such that $s_c + s_s = <\mathbf{x}, \mathbf{y}> \bmod N$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. He generates a random coin $r_i \in_R \mathbb{Z}_N^*$ and computes $c_i \leftarrow E_K(x_i, r_i)$ for $i = 1, \ldots, n$. The chooser sends K and $c_1, \ldots, c_n$ to the sender.
2. The sender performs the following: generates a random coin $r \in_R \mathbb{Z}_N^*$, a random share $s_s \in_R \mathbb{Z}_N$ and computes $c \leftarrow E_K(-s_s, 1)(\prod_{i=1}^n c_i^{y_i}) E_K(0, r) \bmod N^2$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts and sets it as his share s_c, i.e. he computes $s_c = D_{SK}(c)$.

Note that $c = E_K(-s_s + \sum_{i=1}^n x_i y_i \bmod N,\ r \prod_{i=1}^n r_i^{y_i} \bmod N)$, thus $s_c = -s_s + <\mathbf{x}, \mathbf{y}> \bmod N$, i.e. the protocol is correct. Again the semantic security of the encryption scheme guarantees the chooser-privacy. The sender-privacy is preserved since the chooser only sees a random encryption of $-s_s + <\mathbf{x}, \mathbf{y}>$, where s_s is random. Note again that the sender uses encryption of 0 (step 2) to re-randomize the ciphertext.

The authors of [19] give an interesting application of an SP protocol: if $x_i, y_i \in \{0, 1\}$, i.e. $\mathbf{x}$ and $\mathbf{y}$ are the characteristic vectors of two sets X and Y, then $<\mathbf{x}, \mathbf{y}> = |X \cap Y|$. In other words such an SP protocol provides solution for the private matching intersection set size problem.

3.4 Private Matching

We first describe the private subset inclusion protocol given by Laur *et al.* [24]. Then we propose a modification to this protocol, which is more efficient.
Laur's Private Subset Inclusion:
The authors of [24] use the fact that $X \subseteq Y$ if and only if $|X| = |X \cap Y|$. Instead of using directly the sets, their characteristic functions (denoted with the same letters) are used in the protocol, where $X[i] = 1$ if $i \in X$ and $X[i] = 0$ otherwise ($Y[i]$ is defined in a similar way).
Private Inputs:

– The chooser input is a set $X \subseteq \{1, \ldots, n\}$.
– The sender input is a set $Y \subseteq \{1, \ldots, n\}$.

Private Output: The chooser gets 0 if $X \subseteq Y$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_{\Pi}(1^k)$. Then he generates a random coin $r_j \in_R \mathbb{Z}_N^*$ and computes $c_j \leftarrow E_K(X[j], r_j)$ for $j = 1, \ldots, n$. The chooser sends K and $c_1, \ldots, c_n$ to the sender.
2. The sender generates random coins $r, s \in_R \mathbb{Z}_N^*$ and computes $c = (\prod_{j:Y[j]=0} c_j)^s \ E_K(0, r) \bmod N^2$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts, i.e. he computes $z = D_{SK}(c)$ and accepts that $X \subseteq Y$ if $z = 0$.

Note that $c = E_K(s \sum_{j:Y[j]=0} X[j] \bmod N,\ r \prod_{j:Y[j]=0} r_j^s \bmod N)$. Thus the chooser gets $z = s \sum_{j:Y[j]=0} X[j] \bmod N$, which is zero only if all $X[j] = 0$ when $Y[j] = 0$. The last relation implies that $X \subseteq Y$.

Private Subset Inclusion:

We also do not use directly the sets in our protocol, but their characteristic functions redefined as follows $X[i] = s_i$ if $i \in X$ (for a random nonzero $s_i \in_R \mathbb{Z}_T^*$ and $T \leq \lfloor N/n \rfloor$) and $X[i] = 0$ otherwise.

Private Inputs:

- The chooser input is a set $X \subseteq \{1, \ldots, n\}$.
- The sender input is a set $Y \subseteq \{1, \ldots, n\}$.

Private Output: The chooser gets 0 if $X \subseteq Y$.

1. The chooser generates a (private, public) key-pair $(SK, K) \leftarrow G_{\Pi}(1^k)$. Then he generates a random coin $r_j \in_R \mathbb{Z}_N^*$ and computes $c_j \leftarrow E_K(X[j], r_j)$ for $j = 1, \ldots n$. The chooser sends K and $c_1, \ldots, c_n$ to the sender.
2. The sender generates a random coin $r \in_R \mathbb{Z}_N^*$ and computes $c = (\prod_{j:Y[j]=0} c_j) \ E_K(0, r) \bmod N^2$. He sends c to the chooser.
3. The chooser decrypts the received ciphertexts, i.e. he computes $z = D_{SK}(c)$ and accepts that $X \subseteq Y$ if $z = 0$.

Note that $c = E_K(\sum_{j:Y[j]=0} X[j] \bmod N,\ r \prod_{j:Y[j]=0} r_j \bmod N)$. Thus $z = \sum_{j:Y[j]=0} X[j]$, which is zero only if all $X[j] = 0$ when $Y[j] = 0$. The last relation implies that $X \subseteq Y$. Obviously this protocol is more efficient than the original protocol of [24] since the sender does not need to compute a random power of $\prod_{j:Y[j]=0} c_j$. Note again that the standard way to re-randomize the ciphertext (step 2) is used in both protocols, i.e. the sender uses an encryption of 0.

3.5 Zero-Knowledge Arguments

Consider the following protocol for equality of double base discrete logarithms. We consider another Σ-protocol than the one in [12] which is for the equality of discrete logarithms, where the prover should prove that indeed $h_1 = g_1^w \bmod p$ and $h_2 = g_2^w \bmod p$ for some w. Let $\tilde{k}$ be the security parameter.

Input:

- The system setting is the tuple $(p, p', g_1, g_2, h_1, h_2)$ where p, p' are prime, p' is $\tilde{k}$-bit long, $p = 2p' + 1$, $g_1 \in \mathbb{Z}_p^*$ has order p' and $g_2, h_1, h_2 \in < g_1 >$.
  In addition $g_2 = g_1^y$ for some secret $y \in \mathbb{Z}_p^*$ and $h_1 = g_1^w g_2^{w_1}, h_2 = g_1^w g_2^{w_2}$ for some $w, w_1, w_2 \in \mathbb{Z}_p^*$.
- The tuple $(p, p', g_1, g_2, h_1, h_2)$ is a common input to the prover and the verifier.
- The prover gets w, w_1, w_2 as private input.

Output: The verifier checks whether $\log_{g_1}(h_1) \bmod y = \log_{g_2}(h_2) \bmod y$.

1. The prover chooses random $3\tilde{k}$-bit integers r, r_1, r_2 and sends $a_1 = g_1^r g_2^{r_1} \bmod p$ and $a_2 = g_1^r g_2^{r_2} \bmod p$ to the verifier.
2. The verifier chooses the challenge e at random in $\mathbb{Z}_{p'}$ and sends it to the prover.
3. The prover computes $z = r + ew$, $z_1 = r_1 + ew_1$, $z_2 = r_2 + ew_2$ and sends them to the verifier who checks that $g_1^z g_2^{z_1} = a_1 h_1^e \bmod p$ and $g_1^z g_2^{z_2} = a_2 h_2^e \bmod p$.

4 The Proposed Attack

4.1 Attack against Oblivious Transfer

We first specify the information that the chooser possesses after finishing the protocol.

- Consider the Stern's OT protocol described in Section 3.1. Denote by $\overline{r} = r\prod_{i=1}^n r_i^{\mu_i} \bmod N$ and recall that $D_{SK}(c) = (\overline{\mu}, \overline{r})$, where $\overline{\mu} = \sum_{i=1}^n \mu_i x_i \bmod N$. Thus the chooser obtains $\overline{\mu}$ and $\overline{r}$.
- Consider the OPE protocol described in Section 3.2. Denote by $\overline{r} = r\prod_{j=1}^n r_j^{a_j} \bmod N$ and recall that $D_{SK}(c) = (z, \overline{r})$, where $z = P(\overline{x})$. Thus the chooser obtains z and $\overline{r}$.
- Consider the private SSP protocol described in Section 3.3. Recall that $D_{SK}(c) = (\overline{m}, \overline{r})$, where $\overline{m} = -s_s + \sum_{i=1}^n x_i y_i \bmod N$ and $\overline{r} = r\prod_{i=1}^n r_i^{y_i} \bmod N$. Thus the chooser obtains $\overline{m}$ and $\overline{r}$.
- Consider the modified Subset Inclusion protocol described in Section 3.4. Recall that $D_{SK}(c) = (z, \overline{r})$, where $z = \prod_{j:Y[j]=0} X[j] \bmod N$ and $\overline{r} = r\prod_{j:Y[j]=0} r_j \bmod N$. Thus the chooser obtains $\overline{z}$ and $\overline{r}$.

Notice that in all these cases $\overline{r}$ has a common form, which we will further unify as $\overline{r} = r\prod_{i=1}^n r_i^{y_i} \bmod N$.

Scenario:

Now we describe the scenario in which our attack can be mounted by the chooser. Recall that the sender's inputs to the protocol are $y_i \in \mathbb{Z}_T$. We consider the case when $T \ll N$, i.e. is very small; how small will be specified later. In this case a semi-honest chooser with *irrational* behavior can try to get some information on the sender's inputs with a non-negligible probability. For the sake of simplicity we only consider the case of a uniform probability distribution for $(y_1, \ldots, y_n)$, but our results hold for any probability distribution.

Attack - Phase 1:
Let the chooser select r_i in step 1 to be small prime numbers, e.g. $2 \leq p_1 \leq p_2 \leq \ldots \leq p_n \ll N$. Thus the probability that $gcd(r_i, r) \neq 1$ for some i is $\frac{1}{r_i}$, when $r \in_R \mathbb{Z}_N^*$ is chosen (independently) by the sender in step 2. Hence the probability $P_i = Pr[r_i = p_i \text{ and } r \in_R \mathbb{Z}_N^* : gcd(r, r_i) \neq 1] = 1/p_i$.

Consider the random coin $\overline{r}$ obtained by the chooser after decrypting the sender's reply. Denote by $\tilde{r} = r \prod_{i=1}^n r_i^{y_i}$ thus $\tilde{r} = \overline{r} + \ell N$, where $\ell = 0, 1, \ldots$. Recall that $y_i \in \mathbb{Z}_T$, $r \in_R \mathbb{Z}_N^*$ and $r_i = p_i$. Denote by $\overline{N} = (\prod_{i=1}^n p_i)^{T-1}$ hence $\ell < \overline{N}$. Denote by $x = \frac{N}{\prod_{i=1}^n p_i^{y_i}}$ (assuming the y_i's are fixed) then $Pr[r \in_R \mathbb{Z}_N^* : r < x] = \frac{x}{N}$ and since the probability that $(y_1, \ldots, y_n)$ is the concrete sender's input is $\frac{1}{T^n}$ we obtain that

$$P[\ell = 0] = Pr[y_i \in_R \mathbb{Z}_T, r \in_R \mathbb{Z}_N^* : r \prod_{i=1}^n p_i^{y_i} < N] \tag{1}$$

$$= \sum_{(\overline{y_1}, \ldots, \overline{y_n})} Pr[(y_1, \ldots, y_n) = (\overline{y_1}, \ldots, \overline{y_n})] \; Pr[r \in_R \mathbb{Z}_N^* : r \prod_{i=1}^n p_i^{\overline{y_i}} < N]$$

$$= \frac{1}{T^n} \sum_{(\overline{y_1}, \ldots, \overline{y_n})} \frac{1}{\prod_{i=1}^n p_i^{\overline{y_i}}} = \frac{1}{T^n} \frac{\prod_{i=1}^n (p_i^T - 1)}{\prod_{i=1}^n p_i^{T-1}(p_i - 1)} > \frac{1}{T^n}.$$

Notice that $2x < N$ when $(y_1, \ldots, y_n) \neq (0, \ldots, 0)$ and $x = N$ when $(y_1, \ldots, y_n) = (0, \ldots, 0)$, thus we obtain $Pr[r \in_R \mathbb{Z}_N^*, x \neq N : x \leq r < 2x] = \frac{x}{N}$. It can be observed that $P[\ell = 0] > P[\ell = i]$ for any $i > 0$, for example:

$$P[\ell = 1] = Pr[y_i \in_R \mathbb{Z}_T, r \in_R \mathbb{Z}_N^* : N \leq r \prod_{i=1}^n p_i^{y_i} < 2N]$$

$$= \frac{1}{T^n} \sum_{(\overline{y_1}, \ldots, \overline{y_n}) \neq (0, \ldots, 0)} \frac{1}{\prod_{i=1}^n p_i^{\overline{y_i}}} = \frac{1}{T^n} \left(\frac{\prod_{i=1}^n (p_i^T - 1)}{\prod_{i=1}^n p_i^{T-1}(p_i - 1)} - 1 \right).$$

Hence $P[\ell = 1] = P[\ell = 0] - \frac{1}{T^n}$. More importantly $P[\ell = 0]$ depends only on the primes selected by the chooser and the system parameters n and T.

Attack - Phase 2:
Now we explain further how the attack works. The protocol is executed just once with the exception that the chooser does not generate the r_i at random but instead selects them as described above. At the end of the execution the chooser possesses $\overline{r}$ and with probability $P[\ell = 0]$ he guesses $\tilde{r}$. Note that $r_i^{y_i}$ is a factor of $\tilde{r}$ co-prime with the other factors, except maybe with r. Let the attacker target some of the secrets y_i for $i \in I$ ($I \subseteq [n] = \{1, \ldots, n\}$). We stress that the chooser should select different prime numbers p_i for $i \in I$. Thus from $\tilde{r}$ the chooser can find $\overline{y_i}, i \in I$, by simple division. Hence $y_i \leq \overline{y_i}$ holds, moreover

the difference between $\overline{y_i}$ and y_i is equal to the power m_i of p_i, such that $p_i^{m_i}$ divides r but $p_i^{m_i+1}$ doesn't. Thus an irrational semi-honest chooser can derive from $\tilde{r}$ upper bounds for all y_i for $i \in I$ with probability $P[\ell = 0]$.

Stern's OT protocol and the modified Subset Inclusion protocol give the chooser additional information namely μ_σ (z respectively) which can be used to verify the derived values $\overline{y_i}$. If there is a mismatch between them (e.g. $\mu_\sigma > \overline{y_\sigma}$) then the chooser tries the next $\tilde{r}$ for $\ell = 1$ (with probability $P[\ell = 1]$) and so on.

Setting:

We are in position now to clarify the setting of our attack (i.e. when it is feasible at all) and more precisely what we mean by T to be small (i.e. $T \ll N$). We recall here that the security parameter $\tilde{k}$ for the OT is the logarithm of $\frac{1}{T^n}$. The other security parameter k ensures only that the Paillier cryptosystem is secure and in this case $\frac{1}{2^k} = \frac{1}{N} \ll \frac{1}{T^n}$ (i.e. $k \geq \tilde{k}$) holds, i.e. we consider the case when $\frac{1}{T^n}$ is non-negligible in k. Note that in some protocols it is implicitly assumed that $T = N$, but sometimes this requirement is not imposed. We want to point out that for all four protocols T is allowed to be small, moreover for the private SSP (used for PM intersection set problem) and the modified Subset Inclusion protocols we have explicitly $T = 2$.

Recall that the chooser derives with probability $P[\ell = 0]$ upper bounds for all y_i for $i \in I$, i.e. $y_i \leq \overline{y_i}$. Hence to break the security of the protocol, namely the sender's privacy, it is sufficient that $P[\ell = 0] > \frac{1}{T^n}$ (i.e. for $I = [n]$). Indeed the inequality holds, see (1). Thus the attacker obtains upper bounds for the secrets, which contradicts the security goal of the protocol.

Now we will show that if the attacker tries to find the exact values y_i for some set I his success probability is negligible. The attack success probability P of finding the exact values y_i is the product of the probability $P[\ell = 0]$ and the probabilities of $gcd(r_i, r) = 1$ for those $y_i, i \in I$, i.e.

$$P = P[\ell = 0] \prod_{i \in I}(1 - P_i) = \frac{1}{T^n} \prod_{i=1}^{n} \frac{p_i^T - 1}{p_i^{T-1}(p_i - 1)} \prod_{i \in I} \frac{p_i - 1}{p_i}$$

$$= \frac{1}{T^n} \prod_{i \in I} \frac{p_i^T - 1}{p_i^T} \prod_{i \in [n] \setminus I} \frac{(p_i^T - 1)}{p_i^{T-1}(p_i - 1)} = \frac{1}{T^n} \prod_{i=1}^{n}(1 - \frac{1}{p_i^T}) \prod_{i \in [n] \setminus I} \frac{p_i}{p_i - 1}.$$

Obviously the higher P is, the more successful the attack. In order to get the exact values of $y_i, i \in I$, it is sufficient that $P > \frac{1}{T^{|I|}}$ (the random guessing), but it is easy to verify that

$$P = \frac{1}{T^n} \prod_{i=1}^{n}(1 - \frac{1}{p_i^T}) \prod_{i \in [n] \setminus I} \frac{p_i}{p_i - 1} \leq \frac{1}{T^n} \prod_{i=1}^{n}(1 - \frac{1}{p_i^T}) 2^{n-|I|} < \frac{1}{T^{|I|}}$$

because $T \geq 2$ and taking $p_i = 2$ for $i \in [n] \setminus I$. Hence the success probability of this attack is indeed negligible.

But the attacker still can mount a stronger attack than finding upper bounds for the secrets. Since the probability $\overline{P_i} = Pr[r_i = p_i \text{ and } r \in_R \mathbb{Z}_N^* : gcd(r, r_i) \neq$

1 and $gcd(r/r_i, r_i) \neq 1] = \frac{1}{p_i^2}$ the attacker obtains with probability $\overline{P} = P[\ell = 0] \prod_{i \in I}(1 - \overline{P_i})$ that $y_i \in \{\overline{y_i} - 1, \overline{y_i}\}$ for $i \in I$. This is the probability that $m_i \in \{0, 1\}$. Indeed when $I = [n]$ it is easy to check that

$$\overline{P} = \frac{1}{T^n} \frac{\prod_{i=1}^n (p_i^T - 1)}{\prod_{i=1}^n p_i^{T-1}(p_i - 1)} \prod_{i=1}^n \frac{(p_i^2 - 1)}{p_i^2} > \frac{1}{T^n}$$

holds since $p_i \geq 2$ and $T \geq 2$. Thus with probability $\overline{P}$ better than random guessing the attacker derives sets with two values for each of the secrets, which is particularly interesting when $T > 2$.

To summarize, we have proved the inequalities: $P[\ell = 0] > \overline{P} > \frac{1}{T^n} > P$; and we have shown that $y_i \leq \overline{y_i}$ with probability $P[\ell = 0]$ and $y_i \geq \overline{y_i} - 1$ with probability $\overline{P}$.

Example:
Let the system parameters are $T = 5$ and $n = 2$. If the attacker selects $p_1 = 2$ and $p_2 = 3$ the success probabilities of the attacks are as follows: $P[\ell = 0] = \frac{2.89429}{25}$, $P[\ell = 1] = \frac{1.89429}{25}$, $\overline{P} = \frac{1.92953}{25}$ and $P = \frac{0.96476}{25}$ while the random guessing has probability $\frac{1}{25}$. Thus with approximately three times better probability than random guessing the attacker obtains the upper bound and with approximately twice better probability the lower bound for each secret.

Discussion:
A natural question is why this attack doesn't apply to the HOT and AIR protocols? Recall that $D_{SK}(c_i) = (\overline{\mu_i}, \overline{r_i})$, where $\overline{\mu_i} = \mu_i + (\sigma - i)s_i \bmod N$ and $\overline{r_i} = r_i r^{s_i} \bmod N$. But now since the sender chooses r and s_i at random in $\mathbb{Z}_N$ the chooser can not derive s_i from $\overline{r_i}$. The same trick precludes the attack in the original Subset Inclusion protocol described in Section 3.4.

Now we clarify why we call the chooser *irrational*. Note that in order to mount the attack the chooser puts his privacy at risk. This happens because the Paillier cryptosystem is not secure if the used "random coin" is not random. It can be easily verified that if the attacker knows r then he can efficiently reveal m from $E_k(m, r)$. Thus if the sender knows that he is subject to an attack he can reveal the chooser's private input(s). Thus in order to get the sender's inputs the chooser has to bluff, which we call *irrational* behavior.

Our attack does not contradict the semantic security of the Paillier cryptosystem since the attack is performed by the owner of the private key. More precisely the owner of the private key encrypts a message which is then modified by another entity and returned back to the owner, who decrypts it and tries to figure out what the modification was. We would like to point out that the additional information from the random coins affects OT protocols because of their specific nature.

4.2 Attack against Non-Interactive Zero-Knowledge

We apply the compilation technique from [12] to obtain from the zero-knowledge protocol described in Section 3.5 a non-interactive one. Then we show that in this different (compare to OT protocols) scenario our attack can be mounted too.

Setting

Input:

- The system setting is the tuple $(p, p', g_1, g_2, h_1, h_2)$ where p, p' are prime, p' is $\tilde{k}$-bit long, $p = 2p' + 1$, $g_1 \in \mathbb{Z}_p^*$ has order p' and $g_2, h_1, h_2 \in < g_1 >$. In addition $g_2 = g_1^y$ for some secret $y \in \mathbb{Z}_p^*$ and $h_1 = g_1^w g_2^{w_1}, h_2 = g_1^w g_2^{w_2}$ for some $w_1, w_2 \in \mathbb{Z}_p^*$. Let $w \in \mathbb{Z}_T$ and $T \ll N$.
- The tuple $(p, p', g_1, g_2, h_1, h_2)$ is a common input to the prover and the verifier.
- The prover gets w, w_1, w_2 as private input.
- The verifier generates a (private, public) key-pair $(SK, K) \leftarrow G_\Pi(1^k)$. Then he generates a random challenge $e \in_R \mathbb{Z}_N^*$, a random coin $s \in_R \mathbb{Z}_N^*$ and computes $\tilde{c} \leftarrow E_K(e, s)$.
- The prover gets $\tilde{c}$ and K as input.

Output: The verifier checks whether $\log_{g_1}(h_1) \bmod y = \log_{g_2}(h_2) \bmod y$.

Protocol Compile

1. The prover chooses random $3\tilde{k}$-bit integers r, r_1, r_2 and computes $a_1 = g_1^r g_2^{r_1} \bmod p$ and $a_2 = g_1^r g_2^{r_2} \bmod p$.
2. The prover computes $c = E_K(r, \tilde{s})\tilde{c}^w$, $c_1 = E_K(r_1, \tilde{s}_1)\tilde{c}^{w_1}$, $c_2 = E_K(r_2, \tilde{s}_2)\tilde{c}^{w_2}$ with some random coins $\tilde{s}, \tilde{s}_1, \tilde{s}_2 \in_R \mathbb{Z}_N^*$. His proof is the tuple (a_1, a_2, c, c_1, c_2).

Verification

1. The verifier decrypts c, c_1, c_2 obtaining $D_{SK}(c) = (z, \overline{r})$, $D_{SK}(c_1) = (z_1, \overline{r}_1)$, $D_{SK}(c_2) = (z_2, \overline{r}_2)$. Where $z = r + ew$, $z_1 = r_1 + ew_1$, $z_2 = r_2 + ew_2$.
2. Then the verifier checks that $g_1^z g_2^{z_1} = a_1 h_1^e \bmod p$ and $g_1^z g_2^{z_2} = a_2 h_2^e \bmod p$.

Note that the ciphertexts c, c_1, c_2 are randomized by the prover. The verifier can mount the attack as follows. Let us emphasize that we explicitly require $w \in \mathbb{Z}_T$ and $T \ll N$. It is easy to compute that $\overline{r} = \tilde{s}s^w$, $\overline{r}_1 = \tilde{s}_1 s^{w_1}$, $\overline{r}_2 = \tilde{s}_2 s^{w_2}$. In the setting phase the verifier chooses s to be a small prime number e.g. p_1. Thus the probability that $\gcd(s, \tilde{s}) \neq 1$ is $\frac{1}{p_1}$. Moreover since $\tilde{s} \in_R \mathbb{Z}_N^*$ the probability $Pr[\tilde{s}, w : \tilde{s}s^w < N] = \frac{1}{T} \frac{(p_1^T - 1)}{p_1^{T-1}(p_1 - 1)}$ is larger than $\frac{1}{T}$. Hence the same type of attack can again be mounted by the verifier in the verification phase if the space from which w is selected is small. Bound for w can be derived, but not the exact value. Note that we intentionally modified the protocol from [12] to the Pedersen commitment, since the Pedersen commitment can perfectly hide any (even a small) secret w (by w_1 and w_2).

4.3 Precluding the Attack

Finally we propose an easy fix to the protocols in order to resist our attack. Note that all these protocols use an encryption of 0 to re-randomize the ciphertext. If more than one re-randomization is applied (e.g. two) then the probability $\overline{P}[\ell =$

$0] = Pr[r, s \in_R \mathbb{Z}_N^* : rs \prod_{i=1}^{n} p_i^{y_i} < N]$ is smaller than the probability $P[\ell = 0]$ (OT case) multiplied by $\frac{\sum_{i=1}^{N} 1/i}{N} \approx \frac{\ln(N)}{N}$ and therefore becomes negligible. The probability can be computed by taking into account that $Pr[r, s \in_R \mathbb{Z}_N^* : rs < x] = \sum_{i=1}^{x-1} Pr[s = i]\ Pr[r < x/i] = \frac{1}{N} \sum_{i=1}^{x-1} \frac{x}{iN} = \frac{x}{N^2} \sum_{i=1}^{x-1} \frac{1}{i} < \frac{x}{N^2} \sum_{i=1}^{N} \frac{1}{i}$. Thus we have shown that in these settings just one re-randomization is not sufficient, but two (or more) suffice.

5 Conclusion

We have described an attack against several OT protocols and protocols derived from OT such as private matching, oblivious polynomial evaluation and private shared scalar product, which are based on semantically secure homomorphic encryption scheme (e.g. Paillier's). Some semantically secure encryption schemes possess the additional property (e.g. Paillier's) – that they also decrypt the random coin used for the encryption. We have shown that in certain cases the information which can be derived from the random coin is sufficient even for a semi-honest chooser to obtain bounds for the sender's private inputs with non-negligible probability.

The following protocols could be subject to this attack: Stern at Asiacypt'98, Goethals *et al.* at ICISC'04, Chang at ACISP'04, Freedman *et al.* at TCC'05, Damgard *et al.* at TCC'06 if applied in the scenario, when the secrets belong to a space very small compared to the (Paillier's) plaintext space. A fix which precludes the attacks is proposed.

Acknowledgements. We would like to thank the anonymous reviewers of AC 2006 for the very helpful comments and suggestions.

References

1. B. Aiello, Y. Ishai, O. Reingold. Priced Oblivious Transfer: How to Sell Digital Goods, *EUROCRYPT'01*, LNCS 2045, 2001,B. Pfitzmann (Ed.), pp. 119–135.
2. J. Benaloh. Verifiable Secret-Ballot Elections, *Ph.D. Thesis*, Yale's Univ., 1987.
3. I. Blake, V. Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals, *ASIACRYPT'04*, LNCS 3329, 2004, Lee, Pil Joong (Ed.), pp. 515–529.
4. G. Brassard, C. Crepeau, J. Robert. All-or-nothing disclosure of secrets, *Crypto'86*, LNCS 263, 1987, pp. 234–238.
5. D. Catalano, R. Gennaro, N. Howgrave-Graham, P. Nguyen. Paillier's Cryptosystem Revisited, *ACM Conf. on Comp. and Commun. Security*, 2001, pp. 206–214.
6. Y. Chang. Single Database Private Retrieval with Logarithmic Communication, *ACISP'04*, LNCS 3108, 2004, C. Boyd, J. Gonzlez (Eds.), pp. 50–61.
7. C. Crepeau, J. van de Graaf, A. Tapp. Committed Oblivious Transfer and Private Multi-Party Computation, *CRYPTO'95*, LNCS 963, 1995, D. Coppersmith (Ed.), pp. 110–123.
8. G. Di Crescenzo. Private selective payment protocols. *Financial Crypto*'00, LNCS 1962, 2001, Y. Frankel (Ed.), pp. 72–89.
9. G. Di Crescenzo, R. Ostrovsky, S. Rajagopalan. Conditional Oblivious Transfer and time released encryption, *CRYPTO'99*, LNCS 1592, 1999, J. Stern, (Ed.), pp. 74–89.

10. R. Cramer, I. Damgard. Linear zero-knowledge - a note on efficient zero-knowledge proofs and arguments, *ACM Symp. on Theory of Computing*, 1997, pp. 436–445.
11. R. Cramer, I. Damgard. Secret-key Zero-Knowledge, *TCC'04*, LNCS 2951, 2004, M. Naor (Ed.), pp. 223–237.
12. I. Damgard, N. Fazio, A. Nicolosi. Non-Interactive Zero-Knowledge from Homomorphic Encryption, *TCC'06*, LNCS 3876, 2006, S. Halevi and T. Rabin (Eds.), pp. 41–59.
13. I. Damgard, M. Jurik. A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System, *PKC'01*, LNCS 1992, 2001, K. Kim (Ed.), pp. 119–136.
14. T. ElGamal. A public-key Cryptosysytem and a Signature Scheme Based on Discrete Logarithms, *CRYPTO'84*, LNCS 196, 1984, G. Blakley, D. Chaum (Eds.), pp. 10–18.
15. A. Fiat, A. Shamir. How to prove yourself: Practical solutions to Identification and Signature Problems, *CRYPTO'86*, LNCS 263, 1987, A. Odlyzko (Ed.) pp. 186-194.
16. M. Freedman, K. Nissim, B. Pinkas. Efficient Private Matching and Set Intersection, *EUROCRYPT'04*, LNCS 3027, 2004, C. Cachin, J. Camenisch (Eds.), pp. 1–19.
17. M. Freedman, Y. Ishai, B. Pinkas, O. Reingold. Keywords Search and Oblivious Pseudorandom Functions, *TCC'05*, LNCS 3378, 2005, J. Kilian (Ed.), pp. 303–324.
18. S. Goldwasser, S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information, *ACM Symp. on Theory of Computing*, 1982, pp. 365–377.
19. B. Goethals, S. Laur, H. Lipmaa, T. Mielikainen. On Private Scalar Product Computation for Privacy-Preserving Data Mining, *ICISC'04*, LNCS 3506, 2004, pp. 104–120.
20. O. Goldreich, S. Micali, A. Wigderson. How to play any mental game. *ACM Symp. on Theory of Computing*, 1987, pp. 218–229.
21. J. Kilian. Founding Cryptography on Oblivious Transfer, *ACM Symp. on Theory of Computing*, 1988, pp. 20–31.
22. H. Lipmaa. Verifiable Homomorphic Oblivious Transfer and Private Equality Test, *ASIACRYPT'03*, LNCS 2894, 2003, C. Laih (Ed.), pp. 416–433.
23. S. Laur, H. Lipmaa. Additive Conditional Disclosure of Secrets and Applications, *Cryptology ePrint Archive*: Report 2005/378.
24. S. Laur, H. Lipmaa, T. Mielikainen. Private Itemset Support Counting, *ICICS'05*, LNCS 3783, 2005, S. Qing et al. (Eds.), pp. 61–71.
25. D. Naccache, J. Stern. A new public-key cryptosysytem based on higher residues, *ACM Conf. on Computer and Commun. Security*, 1998, pp. 59–66.
26. M. Naor, B. Pinkas. Oblivious Transfer and Polynomial Evaluation, *ACM STOC*, 1999, pp. 245–254.
27. M. Naor, B. Pinkas. Efficient Oblivious Transfer Protocols, *ACM-SIAM Symp. on Discrete Algorithms*, 2001, pp. 448–457.
28. T. Okamoto, S. Uchiyama. A new public-key cryptosysytem as secure as factoring, *EUROCRYPT'98*, LNCS 1403, 1998, K. Nyberg (Ed.), pp. 308–318.
29. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, *EUROCRYPT'99*, LNCS 1592, 1999, J. Stern (Ed.), pp. 223–238.
30. M. Rabin. How to exchange secrets by oblivious transfer, *Technical Report* TR-81, Harvard Aiken Computation Laboratory, 1981.
31. J. Stern. A New and Efficient All-Or-Nothing Disclosure of Secrets Protocol, *ASIACRYPT'98*, LNCS 1514, 1998, K. Ohta, D. Pei (Eds.), pp. 357–371.