

Efficient Committed Oblivious Transfer of Bit Strings

Mehmet S. Kiraz, Berry Schoenmakers, and José Villegas

Dept. of Mathematics and Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
m.kiraz@tue.nl, berry@win.tue.nl, j.a.villegas@tue.nl

Abstract. Oblivious transfer (OT) is a powerful primitive in modern cryptography, often used in a context of semi-honest adversaries. Committed oblivious transfer (COT) is an enhancement involving the use of commitments, which can be used in many applications of OT covering particular malicious adversarial behavior. For OT, many protocols are known that cover the transfer of bit strings rather than just single bits. For COT, though, the known protocols only cover the transfer of bits. In this paper, we thus present efficient COT protocols for transferring (long) bit strings, which perform quite well in comparison to the most efficient COT protocols for bits. We prove the security of our protocols following the simulation paradigm in the cryptographic model, also assuming the random oracle model for efficient non-interactive proofs. Also, as a motivation for the use of COT instead of OT, we point out that a protocol which uses OT as a subprotocol may have subtle security issues in the presence of malicious adversaries.

Keywords: Committed oblivious transfer; Commitments; Homomorphic encryption

1 Introduction

Oblivious transfer is a fundamental primitive in modern cryptography. After Rabin introduced oblivious transfer [Rab81] a huge number of papers appeared regarding possible extensions, variants and applications of oblivious transfer. In Rabin's original oblivious transfer, the sender has a secret and sends it to a chooser who receives the secret with probability $1/2$ while the sender does not know whether the secret has been received. Later, Even, Goldreich and Lempel [EGL85] presented 1-out-of-2 Oblivious Transfer (OT), where the sender has two values s_0 and s_1 and the chooser has a selection bit b . Upon completion of the protocol, the chooser holds the value s_b while the sender does not know which of the two values s_0 and s_1 the chooser got who, in turn, learns nothing about s_{1-b} . Crépeau [Cré87] showed that Rabin's OT and 1-out-of-2 OT are equivalent. 1-out-of-2 OT will be called standard OT throughout the rest of the paper.

Committed Oblivious Transfer (COT) is obtained as a natural combination of 1-out-of-2 oblivious transfer and bit commitments. This notion was first introduced by Crépeau [Cré90] under the name Verifiable Oblivious Transfer. Later, Crépeau *et al.* [CvdGT95] presented a more efficient COT protocol and showed that from COT one can construct a protocol for general secure multi-party computation in the malicious case. Briefly, in these variants of oblivious transfer, the parties running the protocol are committed to their input values prior to the protocol. That is, if the sender has two private values to be obliviously transferred to the chooser who has a selection bit, all these input values must have been committed to by the respective parties before the transfer starts. At the end of the protocol, the chooser will receive one of the corresponding private values of the sender together with a (public) commitment by the chooser.

For many applications in which OT is used as a subprotocol, the security of the overall protocol is considered only in the semi-honest model. However, there may be subtle security issues when such protocols using OT are extended to the malicious case. We highlight this in this paper and we describe how COT helps to overcome these problems. Namely, the link between OT and the surrounding protocol can be securely done with the use of COT.

Since efficient COT protocols for transferring bits are known, one may thus replace applications of OT of bits by COT of bits. An interesting question is how to do the same when transferring bit strings. This is the starting point of this work. In this paper, we present *efficient* protocols for string COT based on any (2,2)-threshold homomorphic cryptosystem.

Related Work

Garay *et al.* [GMY04] present the most efficient COT protocol to date realizing COT functionality in the Universal Composable (UC) framework of Canetti [Can00]. However, this protocol only works for bits whereas our protocol allow for bit strings of arbitrary length (up to the length of plaintexts of the underlying threshold homomorphic cryptosystem, or a multiple thereof). In this paper, we will just consider a stand-alone setting, noting that the efficiency of our protocols improves the efficiency of the UC-protocols of [GMY04] when trimmed down to a stand-alone setting (replacing, e.g., the use of Ω -protocols by Σ -protocols).

If the parties are committed to the inputs of the OT protocol but there is no commitment to chooser's output we refer to this variant as Verifiable OT (VOT) in this paper. In this direction, Cachin and Camenisch [CC00] as well as Jarecki and Shmatikov [JS07] present protocols for VOT in 2 rounds. These protocols can be converted into COT by requesting the chooser to recommit to its received value and to prove the validity of this commitment w.r.t. the commitments for the inputs. In general, this incurs one extra communication round.

Lipmaa [Lip03] also presents a protocol under the name *verifiable homomorphic oblivious transfer* for strings. However, verifiability is defined in a different sense. The chooser will get commitments to all inputs of the sender which can later be used and referred to by the surrounding protocol. Similarly, the sender gets an encryption of the chooser's input. Hence, this is yet another form of OT,

which is related to COT and VOT, and is somewhat similar to the notion of “committing OT”, introduced later in [KS06].

Recently, Camenisch *et al.* [CNs07] presented a protocol for adaptive OT, in which a sender has a list of messages and a receiver adaptively chooses one message after the other. To prevent the so-called selective-failure problem mentioned in [CNs07] (which is similar to the problem discussed in [KS06], see below), the sender is required to commit to its input list of messages and to prove consistency w.r.t. this list in all ensuing runs of adaptive OT.

More generally, we note that standard OT protocols, which are secure in a stand-alone setting, must be carefully dealt with when used as subroutines in higher level protocols. Kiraz and Schoenmakers [KS06] show that there are actually several protocols in the literature (e.g., [Pin03,MNPS04,MF06]) where the use of standard OT compromises the overall security of the protocol. Namely, a malicious sender may put ‘bogus’ values instead of the correct messages, and by doing so, compromise the privacy of the surrounding protocol. The use of COT or VOT protocols may prevent such problems.

Our contributions

We present a protocol that implements COT, assuming that a (2,2)-threshold homomorphic cryptosystem has been setup before (as in, e.g., [CDN01]). This setting also allows for multiple (sequential) runs of the COT protocol, amortizing the initial cost of setting up the (2,2)-threshold cryptosystem. Our COT protocol efficiently transfers bit strings. Using the random oracle model our protocol achieves 2 rounds of interaction.

Compared to the COT protocol of [GMY04], which works for bits only, the cost of transferring $O(k)$ -bit strings (for security parameter k) using our protocol is comparable to the cost of transferring a single bit using the protocol of [GMY04]. Compared to the 2-round VOT protocol of [JS07] for bit strings, which can be turned into a 3-round COT protocol (see above), our protocol uses one round less and is also computationally more efficient. However, [JS07] only assumes a common reference string (CRS) containing an RSA modulus (among other things), while we assume that a (2,2)-threshold homomorphic cryptosystem has been setup.

The security analysis of our COT protocol is done using the simulation paradigm, in the model described by [Lip03]. Although the privacy for both parties is computational (as the commitments in our protocol are public key encryptions), we show a simulation which produces a statistically indistinguishable view of the COT protocol for both parties. Hence, the COT protocol does not divulge any information beyond what can be inferred from the encryptions (which are used as computationally hiding commitments).

Organization of the paper

The rest of the paper is as follows. In Section 2, we give some notation and definitions which are used throughout the paper. In Section 3, we present our

COT protocol together with a proof of security. In Section 4, we discuss the complexity of our protocol and compare with previous solutions. In Section 5, we discuss some applications which have some issues with the use of OT and motivate COT instead and finally we conclude the paper.

2 Preliminaries

Threshold homomorphic cryptosystems. Our results apply to any threshold homomorphic cryptosystem. Briefly, let $E(m, r)$ denote the encryption of value m using randomness r for a semantically secure public key encryption scheme. Often the randomness is omitted in the notation, writing $E(m)$. A cryptosystem is additively homomorphic if the product $E(m_1)E(m_2)$ results in $E(m_1 + m_2)$. As a consequence, for any public constant c , $E(m)^c$ is an encryption whose plaintext is cm .

In a (t, n) -threshold cryptosystem there are n parties, each of them holds a share of the overall secret key. There is a public key which allows anyone to encrypt messages. If at least t parties cooperate, any encryption can be successfully decrypted, whereas any collusion of less than t parties cannot get any information about the plaintext.

There are various instances of threshold homomorphic cryptosystems. The most widely used are (based on) ElGamal or Paillier. Threshold homomorphic ElGamal has the drawback of only allowing decryption of values belonging to a relatively small set, for which it is feasible to compute discrete logs. On the other hand, Paillier does not have this problem and allows decryption of encrypted values in an arbitrarily large set (e.g., 1024-bit integers). However, the distributed key generation protocol for threshold Paillier is very expensive compared to that for threshold ElGamal. It is also possible to use an amalgam of ElGamal and Paillier cryptosystems: the key generation protocol it is that of ElGamal, while allowing decryption of full-size plaintexts like in Paillier. One drawback of the latter is that the security of the cryptosystem relies on two computational assumptions (see [DJ03]).

Σ -protocols. A Σ -protocol for a relation $R = \{(v; w)\}$ is a 3-round protocol between a prover and a verifier, where the prover acts first. Both parties have the value v as common input, and the prover has a *witness* w as private input, where $(v; w) \in R$. A Σ -protocol is a proof of knowledge for relation R which satisfies special soundness and (special) honest-verifier zero-knowledge. See [CDS94] for further details. Moreover, non-interactive Σ -proofs are easily obtained in the random oracle model.

We will also use the fact that both for homomorphic ElGamal encryptions and for Paillier encryptions, there are efficient Σ -protocols for the relation $R_{\text{enc}} = \{(e; m, r) : e = E(m, r)\}$, proving knowledge of the message m and randomness r for a given encryption $e = E(m, r)$.

(Non-Interactive) Public and Private Threshold Decryption. Given a ciphertext in the (t, n) -threshold cryptosystem, at least t parties willing to decrypt, produce

shares of the decryption, based on their respective shares of the secret key. This information is broadcast and with this, everyone can simply recover the plaintext by using a reconstruction algorithm. Putting this more formally, on ciphertext c , at least t parties broadcast $c_i = D_{sk_i}(c)$, where sk_i denotes the secret key share for the i -th party. Later, everyone can perform $m = R(c_1, \dots, c_t)$ where $c = E(m)$, where R denotes the public reconstruction algorithm.

In order to withstand malicious adversaries, parties have to prove that the decryption share c_i is correctly computed. For this, they use a Σ -protocol for the relation $R_{\text{tdec}} = \{(c_i, c; sk_i) : c_i = D_{sk_i}(c)\}$.

For the security of this process, and for later use in our security proofs, we assume that if $t - 1$ parties are corrupted, then there is a simulator that on inputs $e = E(m, r)$, the message m , and the $t - 1$ shares of the private key for the corrupted parties, it can produce a statistically indistinguishable view of the decryption protocol. The concrete details on how to do this depend on the specific threshold encryption scheme used. For examples, see [ST04, Section 2] for the homomorphic threshold ElGamal, and [DJ01, Section 4.1] for the threshold Paillier cryptosystem.

In our protocol, we consider a variant of the threshold decryption protocol, the so-called *private threshold decryption* [CDN01, full version]. Here, the requirement is that one of the t parties will be the only party who will recover the secret. This is easily achieved: all $t - 1$ other parties follow the protocol, and broadcast their shares (along with the proofs of correctness). The party who will learn the plaintext proceeds with the decryption process privately, collects all decryption shares from the $t - 1$ other parties, and privately reconstructs the message. Note that the remaining parties will not get any information about this message.

Secure multi-party computation from threshold homomorphic cryptosystems.

Cramer, Damgård and Nielsen [CDN01] present a framework to build secure multiparty computation of any functionality that can be expressed as an arithmetic circuit (or formulae). Roughly, on inputs $e_1 = E(m_1)$ and $e_2 = E(m_2)$ where m_1 and m_2 may be unknown to everybody, parties can compute $E(m_1 + m_2)$ without interaction because of homomorphic properties. To compute $E(m_1 m_2)$ parties must engage in a *secure multiplication* protocol. If in the latter case one of the values, say m_1 , is private to one of the parties, this party can compute $e = e_2^{m_1} E(0, r)$ proving that this is the case. Clearly, e encrypts $m_1 m_2$. This protocol is usually referred to as *private-multiplier* (see, e.g., [ST04]). For later use in the paper, the relation for the proof given in the private-multiplier gate is denoted as $R_{\text{pm}} = \{(e_1, e_2, e; m_1, r_1, r) : e_1 = E(m_1, r_1) \wedge e = e_2^{m_1} E(0, r)\}$.

For later use in the simulation of our protocols, given $E(m_1)$, $E(m_2)$ and $E(m_1 m_2)$, the private-multiplier gate can be statistically simulated when there are at most $t - 1$ corrupted parties in a (t, n) -threshold homomorphic cryptosystem. For details, see [CDN01, DJ01, ST04, DN03].

Encryptions as Commitments. A probabilistic public key encryption scheme can be used as a non-interactive commitment scheme. One party commits to a

message by encrypting it. The opening is done by disclosing the message and the randomness used.

In this scenario we have to be careful: the holder of the private key can *always* see the contents of any commitment of this type and, depending on the encryption scheme used, this party might recover the randomness and therefore virtually open *any* commitment.¹ This compromises the hiding property for the committer that do not know the secret key.

We can resolve this issue with the following two possible actions: using the encryption scheme as a commitment without allowing any of the parties to know the secret key; while another suitable alternative could be to set up a threshold encryption scenario. In this way, the ability to decrypt can be distributed in a threshold fashion (possibly letting the threshold be the total number of parties).

Given a commitment $e = E(m, r)$, its committer in this scenario is the party that knows both the message m and the randomness r . Note that parties can run private threshold homomorphic decryption w.r.t. one party to retrieve the message behind e , but this not always allows the recipient to obtain the randomness used in e (e.g., ElGamal), and therefore this party will not be able to open e as a commitment. If party P is the committer of $e = E(m, r)$ we denote it by $e = \text{commit}_P(m, r)$.

Committed Oblivious Transfer. In 1-out-of-2 OT there are two parties: the sender S , who inputs two private values s_0 and s_1 ; and the chooser C who has a selection bit b . At the end, C receives the value s_b . The main security requirement of any protocol implementing OT is that after running the protocol S will not gain any information about the C 's selection bit b ; and C will be ignorant about the value of s_{1-b} .

Definition 1. A COT protocol is run between the parties S and C . At the beginning there public commitments $\text{commit}_S(s_0, r_0)$, $\text{commit}_S(s_1, r_1)$, and $\text{commit}_C(b, r)$. S inputs s_0, s_1 and r_0, r_1 , while C inputs b, r . At the end of the protocol, C receives s_b and a fresh commitment $\text{commit}_C(s_b, u)$ is publicly available. S learns nothing about b while C has no clue about s_{1-b} . (See Figure 1)

Now we point out the difference between COT and VOT in more detail. COT and VOT are identical except that in VOT the commitment by the chooser to its selected value s_b is not required. Keeping this in mind, we notice that [Cré90, CvdGT95, CD97, GMY04] are papers that present COT (of bits). Instead, in [CC00, JS07] only VOT protocols are presented. However, the different use of these terms causes some confusion: Crépeau [Cré90] introduces COT under the name of VOT, Jarecki and Shmatikov [JS07] present protocols for VOT, while they use the term COT. In the latter paper, they present a UC-secure VOT protocol (which for them is a COT), modifying the definition of the ideal functionality for COT by Garay *et al.* [GM04] to make it into VOT. It is straightforward to see that in line with our definitions COT implies VOT by

¹ For Paillier encryptions, one is able to recover both the plaintext and the randomness used if one knows the private key. Whereas, for ElGamal encryptions recovering the randomness is impossible under the DL assumption.

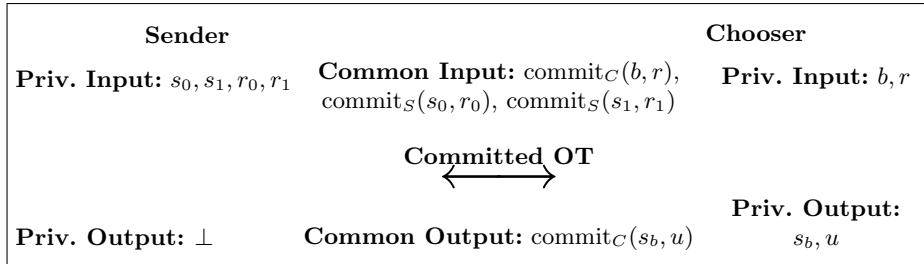


Fig. 1. Committed Oblivious Transfer

just ignoring the output commitment. Vice versa, a VOT protocol can be turned into a COT protocol by adding a round in which the chooser commits to the received bit string and proves the validity of the commitment.

Security definitions. The main security obligation is to show that our protocol achieves the privacy requirements for COT. There are protocols in the literature that achieve unconditional privacy for one of the parties (e.g., [NP01, Tze02, Lip03]) while the privacy for the other party on a computational assumption. As our commitments are encryptions of the underlying threshold public key cryptosystem, we can only give computational privacy to both parties. However, our protocol achieves more than computational privacy: we show that for any corrupted party (sender or chooser) there exists a simulator that produces a view of the protocol which is statistically indistinguishable from the view of the corrupted party executing a real instance of the protocol. This has clear consequences in the framework of [CDN01]: a successful attacker to our protocol is an attacker to the security of underlying cryptosystem without loss in its success probability. This results in modular security proofs of higher level protocols that use our COT as a subroutine.

To carry out such simulations, we proceed as follows. Assuming that one party is corrupted, we build an efficient simulator that has access to the public input, private secret shares of secret key and, as done in [Lip03], the private output in the case that the chooser is corrupted. Besides, the simulator knows the public output.

3 Committed Oblivious Transfer Protocol

In this section, we will present our COT protocol. A $(2,2)$ -threshold homomorphic cryptosystem is assumed to be set up. We let E denote the encryption algorithm of this cryptosystem, and as explained above, we also use E as a non-interactive commitment scheme.

Let $e_0 = E(s_0, r_0)$ and $e_1 = E(s_1, r_1)$ be the commitments to the sender's input strings s_0 and s_1 , and $e = E(b, r)$ be the commitment to the chooser's selection bit b .

Using the general approach to secure multiparty computation of [CDN01], the COT protocol corresponds to the secure evaluation of an arithmetic circuit given

by $t = b(s_1 - s_0) + s_0$ which clearly returns s_0 if $b = 0$ and s_1 when $b = 1$. This approach is so general that even s_0 , s_1 and b need not be known to any party. Note that the output of the evaluation will be an encryption $e' = E(t) = E(s_b)$. If inputs (s_0, s_1) and/or b are known to the respective parties then one can securely compute e' using a private-multiplier gate (instead of a secure multiplication gate), resulting in a more efficient protocol.

Once e' is obtained, according to one of the COT requirements, only the chooser must recover the plaintext. For this, we use *private decryption*, where the chooser is the one who will learn the plaintext inside e' .

To complete the COT protocol, the chooser needs to commit to the received value s_b , and to prove that it does so correctly. In principle, this can be done using some proofs of knowledge. However, we will use the fact that our commitments are encryptions for a threshold cryptosystem: to prove that a fresh commitment e'' to output s_b is correct, we observe that this proof equivalent to show that e''/e' is an encryption of 0. The latter statement is proved by actually decrypting e''/e' .

As a final remark, we see that if the chooser starts producing e' , it turns out that it has to wait for the decryption share of it from the sender, so that it later can produce the fresh commitment as just explained. This results in at least 3 rounds of communication. However, if the sender starts, it produces e' and at the same time the decryption share for e' , which reduces the overall strategy to at least 2 rounds of communication. In both cases, the computational cost is actually the same. For this reason, we only go into the details of this second approach, as it results in a more efficient way of doing COT.

2-round COT protocol

We now present our protocol for COT. This protocol has two rounds and it is quite efficient compared to the state of the art. In the beginning of the protocol, we take advantage of the fact that the values for the commitments e_0 , e_1 and e are known to the respective parties. The protocol is as follows.

Step 1. The sender produces $e' = E(b(s_1 - s_0) + s_0) = e^{(s_1 - s_0)} \cdot e_0 \cdot E(0, r')$ and the Σ -proof for relation R_{pm} on $(e, e_1/e_0, e'/e_0; s_1 - s_0, r_1 - r_0, r')$. The sender also produces its decryption share s_S of e' , along with the Σ -proof for relation R_{tdec} on $(e', s_S; sk_S)$.

Step 2. After checking the two proofs given by the sender, and if they pass, the chooser then produces its corresponding decryption share for e' , denoted as s_C . Combining s_S and s_C , the chooser gets s_b . Immediately, the chooser produces a fresh encryption $e'' = E(s_b, u)$ for a fresh random u , and generates its decryption share for e''/e' , denoted as \hat{s}_C . Then, e'' and \hat{s}_C are sent along with the Σ -proofs for R_{enc} and R_{tdec} on inputs $(e''; s_b, u)$ and $(e''/e', \hat{s}_C; sk_C)$ respectively.

Step 3. Finally, upon receiving e'' , the sender produces its decryption share for e''/e' , denoted as \hat{s}_S . This is combined with \hat{s}_C to check whether the resulting decrypted value is 0. If so, the sender accepts e'' as a valid commitment for the chooser's output. Otherwise, the sender rejects.

The protocol is sketched in Figure 2.

Committed OT of bit strings

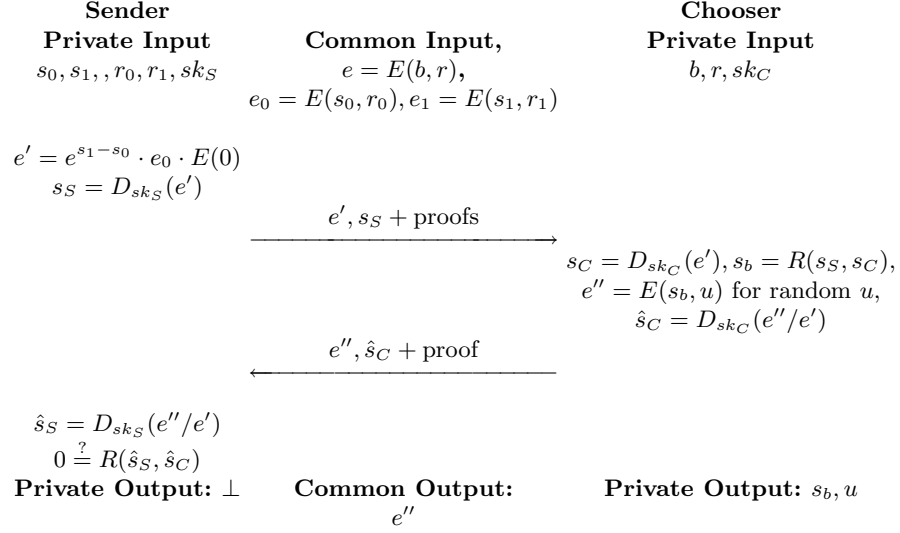


Fig. 2. Sketch of the committed OT protocol

The value s_b denotes the output of the chooser after privately decrypting e' . When this value has been computed, a fresh commitment to s_b (denoted as e'') by the chooser has to be sent in order to fulfill the COT requirement that the chooser's output must be committed. Notice here that without the fresh commitment to s_b the protocol fulfill the VOT requirement in one round only.

Security analysis

For the security analysis, we are going to prove that this protocol fulfills the privacy requirements for COT. We are going to show that given a party is corrupted, there exists a simulator that can produce a view which is statistically indistinguishable from the view of that party interacting with the other honest party.

Before starting the proof we make some remarks in the security model to make the proof precise. As we mentioned earlier, before the simulation is run the simulator already knows the shares of the secret key of the corrupted party. The reason is that the threshold cryptosystem is set up before the protocol starts, and therefore we assume that the simulator extracts this information when the distributed key generation is run.

Also, in case the chooser is corrupted, we use the approach in [Lip03]: the simulator will be given access to the received value by the chooser. From this and public information, we construct a simulator that produces an indistinguishable view for the adversary w.r.t. the view in the real execution.

Finally, we remind that the protocol gives computational privacy to both parties, the sender and the chooser, because of the semantic security of the underlying cryptosystem. Going a bit further than computational privacy, we now show that the protocol is simulatable for both parties and those simulations produce views which are statistically indistinguishable from the views in the real protocol executions.

Theorem 1. *On the sender's inputs s_0, s_1 (and randomness r_0 and r_1), the chooser's private selection bit b (and randomness r), where public commitments to the parties' inputs $e_0 = E(s_0, r_0)$, $e_1 = E(s_1, r_1)$, and $e = E(b, r)$ are available, the COT protocol privately gives s_b (and a fresh randomness u) to the chooser, along with a public commitment $e'' = E(s_b, u)$.*

Proof. As we argued before, we assume that one of the parties is corrupted. Based on public information besides of its private decryption share, we show a simulation which produces a view to the adversary that is statistically indistinguishable from the view in the real protocol execution.

In all cases, a set of valid public inputs is available: e is a commitment to the chooser's selection bit, and e_0, e_1 are respective commitments to the sender's inputs. Also, the simulator is assumed to get the public output commitment e'' which is a valid commitment to chooser's received value.

Case 1- The chooser is corrupted. We first prove the security for the case that the chooser is corrupted. The simulator has the chooser's private key share sk_C , and received value s_b , apart from the public commitments. From this information, the simulator constructs a view for the chooser which is statistically close to the one when interacting with the honest sender.

The simulator proceeds as follows:

1. The simulator computes $e' = e'' \cdot E(0)$. The value e' together with a simulation of the private-multiplier gate (over multiplicands e and e_1/e_0 and result e'/e_0) are output.
2. At the same time, the decryption share s_S can be simulated given e' , its plaintext (which is s_b) and the share of private key sk_C of the chooser. All proofs at this stage are also simulated.

This completes the simulation for the malicious chooser. The transcript is consistent and statistically indistinguishable from the chooser's view when interacting with the honest sender.

Case 2- The sender is corrupted. We next prove the security for the case that the sender is corrupted. The simulator has only sender's private key share sk_S and all public information as described above. From this information, the simulator

constructs a view for the sender which is statistically close to the one when interacting with the honest chooser.

The simulator proceeds as follows:

1. The simulator waits until the sender produces the encryption e' and the decryption share for e' . The simulator checks all the proofs as if the honest chooser would check in the real protocol execution. If all proofs are passed, the simulator goes on, otherwise it aborts.
2. Now, simulator prepares e'' as $e' \cdot E(0)$ and outputs it along with a simulated proof of knowledge. Also, it simulates \hat{s}_C calling the simulator to the decryption process on inputs e''/e' , plaintext 0 and the sender's secret key share sk_S .

This completes the simulation for the malicious sender. The transcript is consistent and statistically indistinguishable from the sender's view when interacting with the honest receiver. \square

4 Complexity analysis and comparison

Our protocol involves only a constant computational, communication and round complexities. When studied in similar frameworks, our protocols are as efficient as the COT protocol by Garay *et al.* [GM04] which is the most efficient one up to now. We stress that the protocol of Garay *et al.* works in a stronger model, since they are interested in the UC framework. We, instead, will adapt their protocol to our framework to be able to carry out a comparison.

In the following, we present the precise description for the complexity of our protocol. For a concrete result we use (2,2)-threshold ElGamal cryptosystem by considering offline computations. In the protocol by Garay *et al.* they need Ω -protocols for the proofs of knowledge. For simplicity, we trimmed them down to the simpler Σ -protocols. This is done to make a reasonable comparison.

COT protocol by Garay et al. Let's roughly sketch the protocol idea. The CRS consists of the pair (g, h) , where nobody knows the discrete log x of h to the base g , i.e. $h = g^x$. The protocol uses Pedersen commitments, and so, let $E_0 = g^{r_0} h^{s_0}$ and $E_1 = g^{r_1} h^{s_1}$ denote the commitment to sender's inputs s_0 and s_1 . Also, $E = g^r h^b$ is the commitment to chooser's input b . The protocol has the following two main steps:

1. The sender "re-encrypts" E_0 and E_1 under the 'keys' E and E/h respectively. Denote E'_0 and E'_1 the resulting encryptions. Note that E_b will be re-encrypted with the key g^r . It also proves that this is done correctly.
2. The chooser can only "decrypt" the message in E'_b as it knows the secret exponent r , recovering s_b . On the other hand, the chooser cannot decrypt E'_{1-b} unless the discrete-log of h to the base g is known. To finish, the chooser has to recommit to the received value s_b and prove that this is the case.

See [GM04] for more details. In the first step, for the reencryption, 4 exponentiations are computed by the sender (2 of them can be off-line). The proofs at that step cost 16 exponentiations (8 of them can be off-line). As for the second step, the chooser needs only 1 on-line exponentiation to retrieve the chosen value. To finish, the chooser computes a fresh commitment which costs 1 off-line exponentiation. The proof of knowledge at the end costs 8 exponentiations (4 can be off-line). In total, there are 15 on-line and 15 off-line exponentiations.

VOT by Jarecki and Shmatikov. We now just sketch the VOT protocol in [JS07]. The input commitments are encryptions under a homomorphic public key cryptosystem (the public key is part of the CRS). The chooser first sends a new public key together with the encryption of its selection bit under this new cryptosystem, proving that this is done correctly. Later, the sender encrypts its inputs under this new public key, combining them with the encryption for the selection bit. Finally, the chooser can decrypt both ciphertexts, but only one of them contains the selected value, and the other one is random.

To convert it into COT, the chooser must recommit to its received value, producing a proof that the value encrypted is consistent with previous commitments. This protocol results in 3 rounds.

This scheme virtually works for any homomorphic encryption. When instantiated to additively homomorphic ElGamal, for the sake of our comparison, the protocol is slightly less efficient than that of [GM04], around 17 on-line and 16 off-line exponentiations (mainly due to the generation of the new cryptosystem, the recommitment of the selection bit and the respective proofs of knowledge). Meanwhile, for the VOT protocol, the cost is 13 and 10 on-line and off-line exponentiations, resp.

Our COT protocol. Now we present the computational cost for our protocol in the case of (2,2)-threshold ElGamal. In Table 1 we study the computational complexity of the building blocks used in our protocol. For the private-multiplier gate, we include the cost of producing the output plus the Σ -proof for relation R_{pm} . In the case of the private threshold decryption, we include the costs for generation of the decryption shares and one Σ -proof for R_{tdc} . And finally, we consider the recommitment at the last step. Concretely, in the case of e'' , chooser has to encrypt to the received value plus the Σ -proof for the knowledge of the randomness used in that encryption. This suffices as if chooser passes this proof and e'/e'' decrypts to 0, it implies it knows the plaintext in e'' . We divide the complexities analysis into on-line and off-line computations.

To get the total number of exponentiations, we note that our protocol requires one private-multiplier gate at the first step (to produce e'), two private threshold decryptions (for decrypting e' and e''/e') and one encryption at the last step (to generate e''). Therefore, we have in total 12 on-line and 12 off-line exponentiations.

Observe that the way of proving that the fresh commitment is correct in our protocol is different (yet equally efficient as the proof in [GM04]). The protocol in [GM04] needs 9 exponentiations to recommit and prove. Our needs

	Online	Offline
Private-multiplier gate	3	5
Private threshold decryption	4	2
Recommitment	1	3

Table 1. Number of exponentiations of building blocks used in our COT protocol for (2,2)-threshold ElGamal setting

9 exponentiations as well: produce e'' and one threshold decryption. technique we use is a little bit different from the general zero knowledge proofs.

If we restrict ourselves to a VOT protocol, removing the recommitment step, we can see our protocol is really much more efficient than the current protocols in the state-of-the-art. It certainly requires 7 on-line and 7 off-line exponentiations (against 11 and 10 resp. for Garay *et al.*'s protocol) and also in only one round of interaction. Ours easily generalizes to any (2,2)-threshold homomorphic cryptosystem at a cost of a distributed key generation protocol at the beginning.

5 Concluding Remarks

As we mentioned before there is a generic attack that can be produced when oblivious transfer is used as a building block for higher level protocols implementations. Kiraz and Schoenmakers [KS06] present that there are several protocols for secure two-party computation using Yao's garbled circuit in the presence of malicious adversaries [Pin03,MNPS04,MF06] which have a security issue with the use of standard OT, and COT is presented as a direct solution. Generally, the problem arises due to the fact that there is no connection between the intermediate outputs in the protocol to the ones that are input for the OT protocols. We note that COT protocols (or any other combination of OT and commitments) may be therefore better to use within larger protocols assuming the correctness of the values inside the commitments. This correctness is controlled by the surrounding protocol and not by the COT protocol.

Moreover, there are a number of protocols for standard OT over bit strings which are profitable for many applications, and therefore, we stress that our COT protocol may also result in efficient implementations since it works for bit strings. Also we stress that once OT is used as a subprotocol in the semi-honest model, a COT protocol might be a good candidate to extend the higher-level protocol to the malicious case. Of course other solutions may be applicable though, but that would imply, in most of the cases, a redesign of the protocol being considered.

Finally, we highlight that our setting is quite different from the previous OT protocols. We use a (2,2)-threshold setting in our protocol and of course, one might easily extend it to (2, n)-threshold cryptosystem. In particular it might be interesting the case $n = 3$ since still the adversary consists of only one party. The setting to adopt might clearly depend on the applications.

As further work, it could be interesting to present a protocol for committed oblivious transfer for bit strings in the universal composable framework and in the non-erasure model.

Acknowledgements. We thank Stas Jarecki and anonymous referees for valuable comments on a previous version of this paper. The work has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The third author is supported by the research program Sentinels (<http://www.sentinelis.nl>). Sentinels is being financed by Technology Foundation STW, the Dutch Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

References

- [Can00] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2000.
- [CC00] C. Cachin and J. Camenisch. Optimistic fair secure computation. In *Advances in Cryptology—Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 93–111. Springer-Verlag, 2000.
- [CD97] R. Cramer and I. Damgård. Linear zero-knowledge – a note on efficient zero-knowledge proofs and arguments. In *ACM Symposium on Theory of Computing – STOC 1997*, pages 436–445. ACM Press, 1997.
- [CDN01] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300. Springer-Verlag, 2001.
- [CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—Crypto 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1994.
- [CNs07] J. Camenisch, G. Neven, and a. shelat. Simulatable adaptive oblivious transfer. In *Advances in Cryptology—Eurocrypt 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 573–590. Springer-Verlag, 2007.
- [Cré87] C. Crépeau. Equivalence between two flavours of oblivious transfers. In *Advances in Cryptology—Crypto 1987*, *Lecture Notes in Computer Science*, pages 350–354. Springer-Verlag, 1987.
- [Cré90] C. Crépeau. Verifiable disclosure of secrets and applications. In *Advances in Cryptology—Eurocrypt 1990*, volume 434 of *Lecture Notes in Computer Science*, pages 181–191. Springer-Verlag, 1990.
- [CvdGT95] C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. In *Advances in Cryptology—Crypto 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 110–123. Springer-Verlag, 1995.
- [DJ01] I. Damgård and M. Jurik. A generalization, a simplification and some applications of Paillier’s probabilistic public-key system. In *Public Key Cryptography—PKC 2001*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer-Verlag, 2001.

- [DJ03] I. Damgård and M. Jurik. A length-flexible threshold cryptosystem with applications. In *ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364. Springer-Verlag, 2003.
- [DN03] I. Damgård and J. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology—Crypto 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 247–264. Springer-Verlag, 2003.
- [EGL85] S. Even, O. Goldreich, and A. Lempel. Randomized protocol for signing contracts. In *Communications of the ACM*28, pages 637–647, 1985.
- [GMY04] J. Garay, P. MacKenzie, and K. Yang. Efficient and universally composable committed oblivious transfer and applications. In *Theory of Cryptography Conference – TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 297–316. Springer-Verlag, 2004.
- [JS07] S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. In *Advances in Cryptology—Eurocrypt 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 97–114. Springer-Verlag, 2007.
- [KS06] M. Kiraz and B. Schoenmakers. A protocol issue for the malicious case of Yao’s garbled circuit construction. In *27th Symposium on Information Theory in the Benelux*, pages 283–290, 2006.
- [Lip03] H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology—Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 416–433. Springer-Verlag, 2003.
- [MF06] P. Mohassel and M. Franklin. Efficiency tradeoffs for malicious two-party computation. In *Public Key Cryptography—PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 458–473. Springer-Verlag, 2006.
- [MNPS04] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – a secure two-party computation system. In *USENIX Security*, pages 287–302, 2004.
- [NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *12th annual ACM-SIAM symposium on Discrete algorithms—SODA ’01*, pages 448–457. ACM Press, 2001.
- [Pin03] B. Pinkas. Fair secure twoparty computation. In *Advances in Cryptology—Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 87–105. Springer-Verlag, 2003.
- [Rab81] M. Rabin. How to exchange secrets by oblivious transfer. *Technical Report TR-81, Harvard Aiken Computation Laboratory*, 1981.
- [ST04] B. Schoenmakers and P. Tuyls. Practical two-party computation based on the conditional gate. In *Advances in Cryptology—Asiacrypt 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 119–136. Springer-Verlag, 2004.
- [Tze02] W. Tzeng. Efficient 1-out-of- n oblivious transfer schemes. In *Public Key Cryptography—PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 159–171. Springer-Verlag, 2002.