Robuster Combiners for Oblivious Transfer

Remo Meier, Bartosz Przydatek, and Jürg Wullschleger

Department of Computer Science, ETH Zurich
8092 Zurich, Switzerland
remmeier@tik.ee.ethz.ch, {przydatek, wjuerg}@inf.ethz.ch

Abstract. A $(k; n)$-*robust combiner* for a primitive $\mathcal{F}$ takes as input n candidate implementations of $\mathcal{F}$ and constructs an implementation of $\mathcal{F}$, which is secure assuming that at least k of the input candidates are secure. Such constructions provide robustness against insecure implementations and wrong assumptions underlying the candidate schemes. In a recent work Harnik *et al.* (Eurocrypt 2005) have proposed a $(2; 3)$-robust combiner for oblivious transfer (OT), and have shown that $(1; 2)$-robust OT-combiners of a certain type are impossible.

In this paper we propose new, generalized notions of combiners for two-party primitives, which capture the fact that in many two-party protocols the security of one of the parties is unconditional, or is based on an assumption independent of the assumption underlying the security of the other party. This fine-grained approach results in OT-combiners *strictly stronger* than the constructions known before. In particular, we propose an OT-combiner which guarantees secure OT even when only one candidate is secure for both parties, and every remaining candidate is flawed for one of the parties. Furthermore, we present an efficient *uniform* OT-combiner, i.e., a single combiner which is secure *simultaneously* for a wide range of candidates' failures. Finally, our definition allows for a very simple impossibility result, which shows that the proposed OT-combiners achieve optimal robustness.

Keywords: robust combiners, oblivious transfer, weak oblivious transfer.

1 Introduction

Many cryptographic schemes are based on unproven assumptions about the difficulty of some computational problems. While there exist assumptions whose validity is supported by decades of research (e.g., factoring or discrete logarithm), many new assumptions offering new possibilities are being proposed in literature, and it is unclear how to decide which assumptions are trustworthy. Therefore, given multiple implementations of some cryptographic primitive, each based on different assumptions, it is often difficult to decide which implementation is the most secure one.

Robust combiners offer a method for coping with such difficulties: they take as input multiple candidate schemes based on various assumptions, and construct a scheme whose security is guaranteed if at least *some* candidates are secure. That

S.P. Vadhan (Ed.): TCC 2007, LNCS 4392, pp. 404–418, 2007.

is, the resulting scheme is secure as long as sufficiently many of the assumptions underlying the input candidates are valid. This provides tolerance against wrong assumptions since even a breakthrough algorithm for breaking one (or some) of the assumptions does not necessarily make the combined scheme insecure.

Actually, the concept of robust combiners is not new, and many constructions of this type have been used in various cryptographic schemes to improve the security guarantees, e.g., cascading of block ciphers. However, a rigorous study of robust combiners was initiated only recently [Her05, HKN+05]. More formally, a $(k; n)$-*robust* $\mathcal{F}$-*combiner* is a construction which takes as input n implementations of a primitive $\mathcal{F}$, and yields an implementation of $\mathcal{F}$ which is guaranteed to be secure as long as at least k input implementations are secure. Robust combiners for some primitives, like one-way functions or pseudorandom generators, are rather simple, while for others, e.g., for oblivious transfer (OT), the construction of combiners seems considerably harder. In particular, Harnik *et al.* [HKN+05] show that there exists no "transparent black-box" $(1; 2)$-robust OT-combiner. In the same paper they propose also a very simple and efficient $(2; 3)$-robust OT-combiner.

Contributions. We propose stronger and more general definitions of robust combiners for two-party primitives, which enable a more fine-grained approach to the design of combiners. In particular, the new definitions capture scenarios where in the candidate implementations the security of Alice is based on an assumption different from the assumption underlying Bob's security, or where the security of one party is unconditional. This finer distinction can then be exploited in constructions of combiners.

For this new definition we propose OT-combiners yielding secure OT when the total number of candidates' failures on either side is strictly smaller than the number of candidates. In particular, we propose an OT-combiner which guarantees secure OT even when only one candidate is secure for both parties, and every remaining candidate is insecure for one of the parties. Moreover, we propose also an efficient *uniform* OT-combiner, i.e., a single combiner which is secure *simultaneously* for a wide range of candidates' failures.

We show also that the proposed combiners are optimal in terms of achieved robustness. Specifically, we prove the impossibility of black-box OT-combiners achieving better robustness, and also we show that any (possibly even non-black-box) OT-combiner achieving better robustness would in fact implement OT from scratch. This is in contrast to the impossibility proof from [HKN+05] where only the existence of *transparent* black-box combiners was excluded. However, our impossibility results are not directly comparable with the previous one: on one hand our results are stronger, since they are not limited to the *transparent black-box* combiners, but on the other hand they are weaker, since they exclude a primitive which is stronger then the one considered in [HKN+05] (cf. Section 3).

Finally, since our definition is stronger than the previous definition, all constructions satisfy also the latter, and as a corollary we obtain also tight bounds for the previous definition.

Related work. As mentioned above, there are numerous implicit uses and constructions of combiners in the literature (e.g., [AB81, EG85, MM93, DK05, HL05]), but a more rigorous study of robust combiners was initiated only recently, by Herzberg [Her05] and by Harnik *et al.* [HKN+05], who formalized the notion of combiners, and have shown constructions of combiners for various primitives. Moreover, Harnik *et al.* [HKN+05] have shown also that not all primitives are easy to combine, by proving that there is no *transparent black-box* $(1;2)$-robust OT-combiner. Boneh and Boyen [BB06] studied the efficiency of combiners for collision resistant hash functions. In [MP06] robust combiners for private information retrieval were proposed, and also *cross-primitive* combiners have been studied. Such combiners can be viewed as generalized reductions between primitives, and their study, in addition to be of practical value, offers insights into relations between cryptographic primitives.

The problem of strengthening imperfect oblivious transfer, which is closely related to OT-combiners (see below), was first considered by [CK88], and has been studied in many subsequent works (e.g.,[BCW03, Cac98, DFSS06]). In particular Damgård, Kilian and Salvail [DKS99] defined the notion of *weak* oblivious transfer (WOT) and provided algorithms for strengthening it. The use of techniques for strengthening WOT in the construction of combiners has been suggested by Harnik *et al.* [HKN+05] as an alternative way of obtaining a $(2;3)$-robust OT-combiner.

Organization. In the next section we review the primitives used in the rest of the paper, and present generalized definitions of robust combiners for two-party primitives. Then in Section 3 we propose combiners for oblivious transfer tolerating an insecure minority, and prove that they achieve optimal robustness. In Section 4 we exploit the symmetry of oblivious transfer to obtain *uniform* OT-combiners with optimal robustness. Finally, in Section 5 we conclude and discuss some open problems.

2 Preliminaries and Definitions

2.1 Primitives

We review shortly the primitives relevant in this work. For more formal definitions we refer to the literature. The parties participating in the protocols and the adversary are assumed to be probabilistic polynomial time Turing machines, (PPTMs).

Oblivious transfer[1] (OT) is a protocol between a sender holding two bits b_0 and b_1, and a receiver holding a choice-bit c. The protocol allows the receiver to get bit b_c so that the sender does not learn any information about receiver's choice c, and the receiver does not learn any information about bit b_{1-c}.

[1] The version of oblivious transfer described here and used in this paper is more precisely denoted as *1-out-of-2 bit-OT* [EGL85]. There are several other versions of OT, e.g., *Rabin's OT*, *1-out-of-n bit-OT*, or *1-out-of-n string-OT*, but all are known to be equivalent [Rab81, Cré87, CK88].

Weak oblivious transfer. $((p,q)$-WOT) is an oblivious transfer with relaxed privacy guarantees for the participants [DKS99]: with probability at most p a cheating sender will learn which bit the receiver chose to receive, and with probability q a cheating receiver will learn both of the sender's input bits.

Secret sharing. [Bla79, Sha79] allows a party to distribute a secret among a group of parties, by providing each party with a *share*, such that only authorized subsets of parties can collectively reconstruct the secret from their shares. We say that a sharing among n parties is a ℓ-out-of-n secret sharing, if any ℓ correct shares are sufficient to reconstruct the secret, but any subset of less than ℓ shares gives no information about the secret. A simple method for ℓ-out-of-n secret sharing was proposed by Shamir [Sha79]: a party P having a secret value $s \in \mathbb{F}_q$ where $q > n$, picks a random polynomial $f(x)$ over $\mathbb{F}_q$, such that $f(0) = s$ and the degree of $f(x)$ is (at most) $\ell - 1$. A share for party P_i is then computed as $s_i := f(z_i)$, where $z_1, \ldots, z_n$ are fixed, publicly known, distinct non-zero values from $\mathbb{F}_q$. Since the degree of $f(x)$ is at most $\ell - 1$, any ℓ shares are sufficient to reconstruct $f(x)$ and compute $s = f(0)$ (via Lagrange interpolation). On the other hand, any $\ell - 1$ or fewer shares give no information about s, since they can be consistently completed to yield a sharing of any arbitrary $\overline{s} \in F[q]$, and the number of possible completions is the same for every $\overline{s}$.

Bit Commitment. (BC) is a two-phase protocol between a sender, holding an input bit b, and a receiver, who has no input. In the *commit* phase the sender commits to bit b without revealing it, by sending to the receiver a *commitment to* b, i.e., an "encrypted" representation e of b. Later, in the *decommit* phase, the sender gives to the receiver a decommitment string d, allowing the receiver to "open" e and obtain b. In addition to correctness, a bit commitment scheme must satisfy two properties: *hiding*, i.e., the receiver does not learn the bit b before the decommit phase, and *binding*, i.e., the sender cannot come up with decommitment strings d, d' which lead to opening the commitment as different bits.

2.2 Robust Combiners

In this section we recall definitions of robust combiners, and present some generalizations, which allow for a more fine-grained approach to the design of combiners. These generalizations are motivated by the fact, that in many implementations of cryptographic primitives various security properties are based on different, often independent, computational assumptions, or even hold unconditionally, and cannot be broken. Thus when designing combiners, whose main goal is to protect against wrong assumptions, it can be worthwhile to consider these security guarantees explicitly, as it can potentially lead to more efficient practical constructions (cf. [HKN$^+$05, MP06]). Moreover, the proposed generalizations lead to combiners which are strictly stronger than the constructions known before, and also allow for easier impossibility proofs.

Definition 1 ($(k;n)$-robust $\mathcal{F}$-combiner [HKN+05]). *Let $\mathcal{F}$ be a cryptographic primitive. A $(k;n)$-robust $\mathcal{F}$-combiner is a PPTM which gets n candidate schemes implementing $\mathcal{F}$ as inputs and implements $\mathcal{F}$ while satisfying the following two properties:*

1. *If at least k candidates securely implement $\mathcal{F}$, then the combiner securely implements $\mathcal{F}$.*
2. *The running time of the combiner is polynomial in the security parameter κ, in n, and in the lengths of the inputs to $\mathcal{F}$.*[2]

If the primitive for which one wishes to construct a combiner is a two-party primitive between Alice and Bob (like for example OT or bit commitment), we can make a finer characterization of the security required from the candidates. That is, we can distinguish cases when in the candidate implementations the security of Alice is based on an assumption different from the assumption underlying Bob's security, or when the security of one party is unconditional. For such candidates breaking one assumption does not necessarily imply a total loss of security (for both parties) and this property can be exploited for the construction of combiners.

Definition 2 ($(\alpha,\beta;n)$-robust $\mathcal{F}$-combiner). *Let $\mathcal{F}$ be a cryptographic primitive for two parties Alice and Bob. A $(\alpha,\beta;n)$-robust $\mathcal{F}$-combiner is a PPTM which gets n candidate schemes implementing $\mathcal{F}$ as inputs and implements $\mathcal{F}$ while satisfying the following two properties:*

1. *If at least α candidates implement $\mathcal{F}$ securely for Alice, and at least β candidates implement $\mathcal{F}$ securely for Bob, then the combiner securely implements $\mathcal{F}$.*
2. *The running time of the combiner is polynomial in the security parameter κ, in n, and in the lengths of the inputs to $\mathcal{F}$.*

Note that a $(k;n)$-robust combiner is a special case of a $(k,k;n)$-robust combiner, but they are not equivalent. For example, a $(2,2;3)$-robust combiner tolerates input candidates $\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3$, where one C_1 is secure for Alice only, C_2 is secure for Bob only, and C_3 is secure for both parties, while a $(2;3)$-robust combiner can fail for such candidates. In other words, the notion of a $(k,k;n)$-robust combiner (and hence of a $(\alpha,\beta;n)$-robust combiner), is strictly stronger then that of a $(k;n)$-robust combiner, and it provides better security guarantees.

Another difference between $(k;n)$- and $(\alpha,\beta;n)$-robust combiners is that for the new definition it is possible to have "non-uniform" constructions with explicit dependence on α and β. This motivates an even stronger notion of *uniform* combiners. For example, even if there exists a $(\alpha,\beta;n)$-robust combiner for every $\alpha,\beta \geq 0$ satisfying $\alpha+\beta \geq \delta$, where δ is some threshold, it might be the case that the combiner makes explicit use of the values α and β, and thus works differently for every particular pair values (α,β). In such a scenario more desirable would

[2] Here an implicit assumption is made, that the candidates themselves run in polynomial time.

be a *uniform* construction, i.e., a single combiner that is secure under the sole assumption that $\alpha + \beta \geq \delta$. In particular, a uniform combiner does not obtain the values of α and β as parameters.

Definition 3 ($\{\delta; n\}$-robust uniform $\mathcal{F}$-combiner). *Let $\mathcal{F}$ be a two-party primitive. We say that an $\mathcal{F}$-combiner is a* $\{\delta; n\}$-robust uniform $\mathcal{F}$-combiner *if it is a $(\alpha, \beta; n)$-robust $\mathcal{F}$-combiner,* simultaneously *for all α and β satisfying $\alpha + \beta \geq \delta$.*

Note that the parameter δ is a bound on *the sum* of the number of candidates secure for Alice and the number of candidates secure for Bob, hence given n candidates δ is from the range $0 \ldots 2n$. As an example consider a $\{4; 3\}$-robust *uniform* combiner. Such a combiner is a (regular) $(2; 3)$-robust combiner, but at the same time it is also a $(3, 1; 3)$-robust combiner, i.e., it tolerates input candidates $\mathsf{C}_1, \mathsf{C}_2, \mathsf{C}_3$, where one C_i is secure for both parties, and the remaining two candidates are secure for Alice only. It is not hard to see that not every $(k; n)$-robust combiner is automatically also a $\{\delta; n\}$-robust *uniform* combiner with $\delta = 2k$. In particular, the $(2; 3)$-robust OT-combiner from [HKN+05] breaks on inputs of the type described above for $(3, 1; 3)$-robust combiner.

For completeness, we recall three more definitions from [HKN+05]. Note that these definitions extend naturally to the generalized combiners from Definitions 2 and 3.

Definition 4 (Black-box combiner [HKN+05]). *A $(1; 2)$-robust combiner is called a* black-box combiner *if the following two conditions hold:*

Black-Box Implementation: *The combiner is an oracle PPTM given access to the candidates via oracle calls to their implementation function.*

Black-Box Proof: *For every candidate there exists an oracle PPTM R^A (with access to A) such that if adversary A breaks the combiner, then R^A breaks the candidate.*[3]

Definition 5 (Transparent black-box combiner [HKN+05]). *A* transparent black-box combiner *is a black-box combiner for an interactive primitive where every call to a candidate's next message function is followed by this message being sent to the other party.*

Definition 6 (Third-party black-box combiner [HKN+05]). *A* third-party black-box combiner *is a black-box combiner where the input candidates behave like trusted third parties. The candidates give no transcript to the players but rather take their inputs and return outputs.*

Since the primary goal of robust combiners is to protect against wrong assumptions, in our constructions we require that the candidates input to a combiner provide the desired functionality and the underlying assumptions can affect only

[3] For $(k; n)$-robust combiners there are at least $n - k + 1$ candidates that can be broken in this manner.

the security properties (e.g. secrecy). This approach is justified by the fact that in cryptographic schemes the security is usually based on some assumptions, while the functionality properties are straightforward and hold unconditionally. Moreover, in some cases a possible way of dealing with unknown implementations of primitives is to test them for the desired functionality, hence, even if the candidate input primitives are given as black-boxes, one can test them before applying a combiner (cf. Section 3.1 in [HKN+05]).

3 OT-Combiner with Secure Majority

The impossibility result for transparent black-box $(1;2)$-robust OT-combiners [HKN+05] implies directly the impossibility of transparent black-box $(n;2n)$-robust OT-combiners, as from their existence would follow the existence of transparent black-box $(1;2)$-robust OT-combiners. Similarly, it implies also the impossibility of transparent black-box $(\alpha,\beta;n)$-robust combiners for $\alpha+\beta\leq n$. However, since $(k,k;n)$-robust combiners are stronger than $(k;n)$-robust combiners, we can show very simple impossibility results, which essentially exclude $(\alpha,\beta;n)$-robust OT-combiners of any type[4], that would work for $\alpha+\beta\leq n$: in Lemma 1 we prove that there are no *black-box* OT-combiners with such robustness, and in Lemma 2 we show that constructing an OT-combiner of *any* type (for $\alpha+\beta\leq n$) is at least as hard as constructing an OT protocol without any assumptions.

As mentioned previously, these results are not directly comparable with the impossibility result from [HKN+05]: on one hand our results are stronger, since they go beyond *transparent black-box* combiners, but on the other hand they are weaker, since they exclude a primitive which is stronger then the one considered in [HKN+05].

Lemma 1. *There does not exist a black-box $(\alpha,\beta;n)$-robust OT-combiner for $\alpha+\beta\leq n$.*

Proof. Assume that such a combiner exists, for some values α, β, and n such that $\alpha+\beta\leq n$. Let OT_1 be the trivial instance of OT where the sender sends both values to the receiver, and let OT_2 be the trivial instance where the receiver sends his choice bit to the sender, who sends the receiver the value of his choice. Observe that OT_1 is information-theoretically secure for the receiver, and OT_2 is information-theoretically secure for the sender.

Consider calling the combiner with input consisting of β instances of OT_1 and $n-\beta\geq\alpha$ instances of OT_2, and let $\overline{\mathsf{OT}}$ denote the resulting OT protocol. By assumption, $\overline{\mathsf{OT}}$ is secure for both parties. Since it is impossible to construct an OT protocol information-theoretically secure for *both* the sender and the receiver, there exists an adversary A (possibly inefficient), which breaks the protocol $\overline{\mathsf{OT}}$. By the definition of a black-box combiner, it follows that given oracle access to A one can break $2n-\alpha-\beta+1>n$ "sides" of the candidates. However, since

[4] I.e., not only *transparent black-box* combiners.

one side of each candidate is information-theoretically secure, we can break at most n sides. A contradiction. □

Lemma 2. *Any $(\alpha, \beta; n)$-robust OT-combiner for $\alpha + \beta \leq n$ implies the existence of OT.*

Proof. Assume that such a combiner exists, for some values α, β, and n such that $\alpha + \beta \leq n$. Let OT_1 and OT_2 be trivial instances of OT, as in the proof of Lemma 1. Calling the combiner using β instances of OT_1 and $n-\beta \geq \alpha$ instances of OT_2 as input yields a secure OT protocol without any assumption. □

We will now show that the bound of Lemmas 1 and 2 is tight, by presenting constructions of $(\alpha, \beta; n)$-robust OT-combiners for any α, β, and n, if $\alpha+\beta > n$. First we describe a combiner, which is very simple but not fully satisfactory, as it is not efficient.[5]

Lemma 3. *For every $\alpha \geq 0$, $\beta \geq 0$ and $\alpha + \beta > n$ there exists an* inefficient *third-party black-box $(\alpha, \beta; n)$-robust OT-combiner.*

Proof. The combiner is a straightforward generalization of the $(2; 3)$-robust OT-combiner from [HKN+05], which is based on two "special-purpose" combiners, combiner **R** for protecting the receiver, and combiner **S** for protecting the sender.[6]

The $(\alpha, \beta; n)$-robust combiner works in two phases: in the first phase subsets of the candidates of size α are combined using the combiner **R**, resulting in $n' = \binom{n}{\alpha}$ OT schemes. Each of resulting instance is secure for the receiver and at least one is secure for both parties. In the second phase the n' OTs are combined using the combiner **S** to yield a final scheme protecting both the sender and the receiver. □

The combiner presented in the above proof is perfect in the sense that it does not introduce any additional error. However, it is inefficient in n, since the value of n', i.e., the number of OTs resulting from the first phase, would be superpolynomial in n. Lemma 5 presents a combiner that is not perfect, but efficient in n and other parameters, as required in Definition 1. In the construction we use a third-party combiner for bit commitment, which is an adaptation of a secret-sharing-based BC-combiner, due to Herzberg [Her05], to our generalized definition. As this may be of independent interest, we describe it separately.

Lemma 4. *For every $n \geq 2$ and for every $\alpha, \beta > 0$ satisfying $\alpha + \beta > n$ there exists a third-party black-box $(\alpha, \beta; n)$-robust BC-combiner.*

Proof. We describe a string commitment that lets a sender commit to a value $s \in \{0, 1\}^m$, for an arbitrary m such that $2^m > n$, using n candidate implementations of bit-commitment from which at least α are hiding and at least β are binding.

[5] Due to its inefficiency, this is strictly speaking not a robust combiner (cf. Def. 1). In a slight abuse of terminology, we call it an *inefficient* combiner.

[6] For completeness, we recall these special-purpose combiners in the appendix.

The sender computes[7] an ℓ-out-of-n Shamir's secret sharing of s over $\mathbb{F}_{2^m}$, for $\ell := n-\alpha+1$, resulting in shares $s_1, \ldots, s_n$. Then the sender uses the n instances of bit-commitment to commit to the share s_i bit-by-bit, for each $i \in [n]$. In the opening phase the sender opens the commitments to all the shares, and the receiver reconstructs the secret s.

To see that this commitment is hiding, notice that at least α shares are guaranteed to be hidden from the receiver, since at least α candidate bit-commitments are hiding. Therefore before the opening phase the receiver sees at most $n-\alpha < \ell$ shares, which give no information about the secret. On the other hand, since at least β candidates of the bit-commitments are binding, the sender is indeed committed to at least β shares. Since $\alpha + \beta > n$, i.e., $\beta > n - \alpha$, the sharing polynomial, which has degree at most $n - \alpha$, is uniquely determined by these β shares, and so the commitment to s is also binding. □

Lemma 5. *For every $n \geq 2$ and for every $\alpha, \beta > 0$ satisfying $\alpha + \beta > n$ there exists a third-party black-box $(\alpha, \beta; n)$-robust OT-combiner.*

Proof. First assume that the sender and the receiver have a common random string r at their disposal. Later we describe how this additional assumption can be dropped.

Using r, the combiner works as follows: it simulates (p, q)-WOT with $p + q \leq 1 - 1/n$ by picking each time an input OT-candidate uniformly at random. This is possible, since we are having n candidates, α of which are secure for the sender and β are secure for the receiver, with $\alpha+\beta \geq n+1$. By picking one candidate at random we obtain a probability $p \leq (n-\beta)/n$ that the sender learns the receiver's choice, and a probability $q \leq (n - \alpha)/n$ that the receiver learns both bits input by the sender, hence $p + q \leq ((n - \alpha) + (n - \beta))/n = (2n - \alpha - \beta)/n \leq 1 - 1/n$, as required. Given such a (p, q)-WOT, use the (efficient) amplification algorithm of Damgård *et al.* [DKS99] to obtain a secure OT.

To complete the argument, we have to show how the sender and the receiver can generate a common random string r. It is well-known that OT implies bit-commitment [Cré87], and bit-commitment implies coin-toss [Blu82].[8] Therefore, we can convert our n candidate implementation of OT into n candidate implementations of bit-commitment, and then use the bit-commitment-combiner of Lemma 4 to obtain a secure implementation of bit-commitment, provided that $\alpha + \beta > n$. This implementation can then be used to implement coin-toss, i.e., the parties can generate a common random string r using the input candidates only, without additional assumptions. Finally, it is easy to verify that all the described protocols use the candidates in a third-party black-box manner, and that the combiner is efficient. □

From Lemmas 1 and 5, we get immediately the following theorem about $(\alpha, \beta; n)$-robust OT-combiner.

[7] In this computation we view bit-strings from $\{0, 1\}^m$ as elements of $\mathbb{F}_{2^m}$.

[8] For completeness, we describe both protocols in the appendix.

Theorem 1. *There exists a black-box $(\alpha, \beta; n)$-robust OT-combiner if and only if $\alpha + \beta > n$ holds. The construction is third-party black-box and efficient.*

Furthermore, the impossibility result of [HKN+05] together with Lemma 5 yield the following corollary about $(k; n)$-robust OT-combiner.

Corollary 1. *There exists a transparent black-box $(k; n)$-robust OT-combiner if and only if $2k > n$. The construction is third-party black-box and efficient.*

4 OT-Combiners Based on the Symmetry of OT

A closer look at the combiners from the proofs of Lemmas 3 and 5 shows that these are "non-uniform" combiners (cf. Sect. 2.2). Namely, the proofs show that for given $\alpha, \beta > 0$ with $\alpha + \beta > n$ there exists a $(\alpha, \beta; n)$-robust combiner, i.e., the actions of the combiner are different for different values α, β. (For Lemma 3 it is explicit in the construction, and for Lemma 5 it is due to the fact that the amplification algorithm from [DKS99] makes explicit use of parameters p, q.) More desireable would be an *uniform* construction, which would have a switched order of quantifiers, i.e., we would like a single combiner that is secure for every $\alpha, \beta > 0$ with $\alpha + \beta > n$, and would therefore be *strictly stronger* than any of the special combiners. In this section we show how to construct such a combiner by exploiting the *symmetry* of OT, i.e., the fact that given OT with sender Alice and receiver Bob, we can perfectly *logically* reverse it to obtain OT with receiver Alice and sender Bob. That OT can be reversed has first been discovered independently in [CS91, OVY93]. A simpler and more efficient protocol has been proposed in [WW06].

Our construction is based on a simple trick, which is somehow non-standard, yet plausible in most scenarios: we require that the parties can *swap their roles* when executing candidate protocols, i.e., any input candidate OT_i can be executed in such a way that the sender (of the main OT-protocol) plays the role of the receiver in OT_i, and the receiver plays the role of the sender in OT_i. Moreover, we require that we have at our disposal *multiple copies of each candidate implementation* (in particular, our protocols use the candidates both in the original setting as well in the swapped configuration). For example, if the input candidates are given as software packages, these requirements are not a problem, as it means only calling different functions, but if a candidate is given as a pair of physical devices implementing the primitive, the swapping operation can be problematic, as it may require a real physical swap of the corresponding devices. However, it is difficult to come up with a primitive that cannot be swapped or duplicated *in principle*. Such a primitive would need to make use of some kind of a physical phenomenon, only available to one of the parties, but not to the other.

We use the swapping of the roles in OT — which can be viewed as a "physical" reversal — together with a *logical* reversal of OT [CS91, OVY93, WW06] to obtain an OT in the original direction (from the original sender to the original receiver), but *with swapped security properties*. More precisely, let swap be this

SENDER'S INPUT: two bits b_0, b_1
RECEIVERS'S INPUT: choice bit c
INPUT OT PROTOCOLS: OT_1, OT_2

Note: Auxiliary combiners $\mathbf{R}$ and $\mathbf{S}$ [CK88,HKN+05] are described in the appendix.

1. Parties apply swap to obtain $\mathsf{OT}_1^* = \mathsf{swap}(\mathsf{OT}_1)$ and $\mathsf{OT}_2^* = \mathsf{swap}(\mathsf{OT}_2)$.
2. Parties define $\mathsf{OT}' = \mathbf{R}(\mathsf{OT}_1, \mathsf{OT}_2)$ and $\mathsf{OT}'' = \mathbf{R}(\mathsf{OT}_1^*, \mathsf{OT}_2^*)$.
3. Parties invoke $\mathbf{S}(\mathsf{OT}', \mathsf{OT}'')(b_0, b_1; c)$.

Fig. 1. A $\{3;2\}$-robust *uniform* OT-combiner

two-step process, i.e., physical swap followed by logical reversal, and consider an implementation OT and its swapped-and-reversed version, $\mathsf{OT}^* = \mathsf{swap}(\mathsf{OT})$. If OT is a correct OT-protocol, then so is OT^*. Moreover, if in OT the security of the sender is based on assumption $\mathcal{A}$, and the security of the receiver is based on assumption $\mathcal{B}$, then in OT^* we have the opposite situation: the security of the sender is based on assumption $\mathcal{B}$, and the security of the receiver is based on assumption $\mathcal{A}$. In particular, if OT is an implementation unconditionally secure for the sender, then OT^* is an implementation unconditionally secure for the receiver.

As a first application of this swapping trick we show a $\{3;2\}$-robust *uniform* OT-combiner, i.e., a combiner which is simultaneously $(\alpha, \beta; 2)$-robust for any α, β satisfying $\alpha + \beta \geq 3$. Recall that if it is known in advance that the security of one party is guaranteed (e.g. $\alpha = 2$), then the corresponding combiner is very simple [HKN+05]. However, the combiner for the case $\alpha = 2$ is quite different from the combiner for the case $\beta = 2$, hence these simple combiners are not uniform.

The idea behind our uniform combiner is to use *both*, the two candidate OT_1, OT_2, and their swapped counterparts $\mathsf{OT}_1^* = \mathsf{swap}(\mathsf{OT}_1)$ and $\mathsf{OT}_2^* = \mathsf{swap}(\mathsf{OT}_2)$. Since $\alpha + \beta \geq 3$, at least two of OT_1, OT_2, OT_1^*, OT_2^* are secure for both parties, at most one is insecure for the sender, and at most one is insecure for the receiver. This is sufficient to implement a secure OT. The construction makes use of the two "special-purpose" OT-combiners we have used previously in the proof of Lemma 3, i.e., combiner $\mathbf{S}$ for protecting the sender, and combiner $\mathbf{R}$ for protecting the receiver (cf. Appendix). Figure 1 presents the entire construction in more detail, and the following theorem summarizes its properties.

Theorem 2. *There exists a third-party black-box* $\{3;2\}$*-robust uniform OT-combiner using the* swap*-operation.*

Proof. (sketch) Consider the protocol in Figure 1. Let $\overline{\mathsf{OT}}$ denote the resulting OT protocol. $\overline{\mathsf{OT}}$ has to satisfy correctness, privacy for the sender, and privacy for the receiver. Correctness is trivially given due to the correctness of the candidates OT_1, OT_2, the symmetric schemes OT_1^*, OT_2^*, and the combiners $\mathbf{R}$ and

S. Given the symmetry of OT, if the privacy of one party is compromised for one candidate, then the privacy of the other party is compromised for the corresponding swapped candidate. Combining OT_1, OT_2, respectively OT_1^*, OT_2^*, with **R** ensures that the receiver's privacy is protected in both OT' and OT'', and the sender's privacy in at least one of them. Hence $\mathbf{S}(\mathsf{OT}', \mathsf{OT}'')$ protects the sender from a possible security break of one of the input canditates. Finally, is easy to verify that this is a third-party black-box combiner. □

The next lemma gives a general construction to obtain a uniform combiner from a non-uniform one. This construction makes use of the swap-operation, and can be used for combiners of any symmetric two-party primitive.

Lemma 6. *If there exists a $(k, k; 2n)$-robust OT-combiner, then there exists a $\{k; n\}$-robust uniform OT-combiner using the* swap*-operation.*

Proof. (sketch) The (k, n)-robust uniform OT-combiner works as follows: given n candidate instances of OT, satisfying $\alpha+\beta \geq k$, we duplicate all instances, and apply the swap-operation to the duplicates. In this way we obtain $2n$ candidate instances, where at least k of them are secure for the sender, and at least k are secure for the receiver. Now we can apply the $(k, k; 2n)$-robust OT-combiner to these $2n$ instances, and get a secure implementation of OT. □

Lemma 6 together with Theorem 1 give us the following theorem.

Theorem 3. *For any $n \geq 2$ and $\delta > n$, there exists a third-party black-box $\{\delta; n\}$-robust uniform OT-combiner using the* swap*-operation.*

Although the presented uniform OT-combiner works with all OT protocols proposed in the literature, it naturally raises the question whether the role-swapping technique can be dropped. Sommer [Som06] has recently pointed out that for transparent black-box OT-combiners the use of the candidates in the swapped direction is in fact necessary. More precisely, he observed that the impossibility proof of Harnik *et al.* [HKN+05] can be adapted to exclude transparent black-box $\{3; 2\}$-robust uniform OT-combiners using the candidates in the prescribed direction only.

5 Conclusions and Open Problems

We proposed stronger definitions of robust combiners for two-party protocols, which yield robuster, more general combiners for oblivious transfer. The observation that a partially broken candidate implementation can still provide security for one of the parties leads to OT-combiners strictly stronger than the constructions known previously. Furthermore, we have shown that for symmetric two-party primitives even stronger combiners are possible if the parties can swap their roles in the candidate protocols.

As we mentioned above, there is currently a trade-off between the *perfect* security and the efficiency of a combiner: we do not know whether there exists an

efficient (in the number of candidates) perfect uniform OT-combiner. Moreover, it would be interesting to find other settings, in which the introduced swapping-trick could be useful.

Acknowledgments. We would like to thank Thomas Holenstein and Christian Sommer for interesting discussions, and anonymous referees for useful comments. Jürg Wullschleger was supported by a grant from the Swiss National Science Foundation (SNF).

References

[AB81] Charles A. Asmuth and George R. Blakely. An effcient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. *Computers and Mathematics with Applications*, 7:447–450, 1981.

[BB06] Dan Boneh and Xavier Boyen. On the impossibility of efficiently combining collision resistant hash functions. In *Proc. CRYPTO '06*, pages 570–583, August 2006.

[BCW03] Gilles Brassard, Claude Crépeau, and Stefan Wolf. Oblivious transfers and privacy amplification. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 16(4):219–237, 2003.

[Bla79] George R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, pages 313–317. American Federation of Information Processing Societies, 1979.

[Blu82] Manuel Blum. Coin flipping by telephone - a protocol for solving impossible problems. In *COMPCON, Proc. Twenty-Fourth IEEE Computer Society International Conference*, pages 133–137, 1982.

[Cac98] Christian Cachin. On the foundations of oblivious transfer. In *Proc. EUROCRYPT'98*, pages 361–374, May 1998.

[CK88] Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *Proc. IEEE FOCS '88*, pages 42–52, 1988.

[Cré87] Claude Crépeau. Equivalence between two flavours of oblivious transfers. In *Proc. CRYPTO '87*, pages 350–354, 1987.

[CS91] Claude Crépeau and Miklós Sántha. On the reversibility of oblivious transfer. In *Proc. EUROCRYPT '91*, volume 547, pages 106–113, 1991.

[DFSS06] Ivan Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Oblivious transfer and linear functions. In *Proc. CRYPTO '06*, pages 427–444, August 2006.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In *Proc. TCC '05*, pages 188–209, 2005.

[DKS99] Ivan Damgård, Joe Kilian, and Louis Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Proc. EUROCRYPT '99*, pages 56–73, 1999.

[EG85] Shimon Even and Oded Goldreich. On the power of cascade ciphers. *ACM Trans. Comput. Syst.*, 3(2):108–116, 1985.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[Her05] Amir Herzberg. On tolerant cryptographic constructions. In *CT-RSA*, pages 172–190, 2005. full version on Cryptology ePrint Archive, eprint.iacr.org/2002/135.

[HKN+05] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Proc. EUROCRYPT '05*, pages 96–113, 2005.

[HL05] Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In *Proc. TCC '05*, pages 264–282, 2005.

[MM93] Ueli Maurer and James L. Massey. Cascade ciphers: The importance of being first. *Journal of Cryptology*, 6(1):55–61, 1993. preliminary version in Proc. IEEE Symposium on Information Theory, 1990.

[MP06] Remo Meier and Bartosz Przydatek. On robust combiners for private information retrieval and other primitives. In *Proc. CRYPTO '06*, pages 555–569, August 2006.

[OVY93] Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Fair games against an all-powerful adversary. In *Advances in Computational Complexity Theory*, volume 13 of *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 155–169. AMS, 1993.

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer., 1981. Tech. Memo TR-81, Aiken Computation Laboratory, available at eprint.iacr.org/2005/187.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Som06] Christian Sommer. Robust combiners for cryptographic protocols, 2006. Master Thesis, Computer Science Department, ETH Zurich.

[WW06] Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In *Proc. EUROCRYPT '06*, pages 222–232, 2006.

Appendix

For completeness, we recall some constructions used in the proposed combiners. First we describe the "special-purpose" combiners **R** and **S** from [CK88, HKN+05]. Combiner **R** takes n OT candidates, and guarantees security of the receiver if at least one of the candidates is secure for the receiver:

$\mathbf{R}(\mathsf{OT}_1, \ldots, \mathsf{OT}_n)(b_0, b_1; c)$

1. The sender picks random bits $r_1^0, r_2^0, \ldots, r_n^0$, such that $b_0 = r_1^0 \oplus r_2^0 \oplus \cdots \oplus r_n^0$, and sets $r_i^1 := r_i^0 \oplus b_0 \oplus b_1$, for every $i = 1 \ldots n$.
2. The receiver picks random bits $c_1, c_2, \ldots, c_n$ such that $c = c_1 \oplus c_2 \oplus \cdots \oplus c_n$.
3. For every $i = 1 \ldots n$ parties run $\mathsf{OT}_i(r_i^0, r_i^1; c_i)$.
   From i-th execution the receiver obtains output $r_i^{c_i}$.
4. The receiver outputs b_c computed as the XOR of his outputs from all executions, i.e.
$$b_c = r_1^{c_1} \oplus r_2^{c_2} \oplus \cdots \oplus r_n^{c_n} \ .$$

Combiner **S** takes n OT candidtates, and guarantees security of the sender if at least one of the candidates is secure for the sender:

$\mathbf{S}(\mathsf{OT}_1, \ldots, \mathsf{OT}_n)(b_0, b_1; c)$

1. The sender picks random bits $r_1^0, r_2^0, \ldots, r_n^0$, and $r_1^1, r_2^1, \ldots, r_n^1$, such that
$$b_0 = r_1^0 \oplus r_2^0 \oplus \cdots \oplus r_n^0 \quad \text{and} \quad b_1 = r_1^1 \oplus r_2^1 \oplus \cdots \oplus r_n^1 \ .$$
2. For every $i = 1 \ldots n$ parties run $\mathsf{OT}_i(r_i^0, r_i^1; c)$.
From i-th execution the receiver obtains output r_i^c.
3. The receiver outputs b_c computed as the XOR of his outputs from all executions, i.e.
$$b_c = r_1^c \oplus r_2^c \oplus \cdots \oplus r_n^c \ .$$

The following protocol generates a random bit-string using bit-commitment. Let $m > 0$.

Coin-toss

1. The sender picks a random $s' \in \{0,1\}^m$ and commits to it.
2. The receiver picks a random $s'' \in \{0,1\}^m$ and sends it to the sender.
3. The sender opens the commitment to s', and both parties output $s = s' \oplus s''$.

The following protocol implements bit-commitment using OT. Let $\kappa > 0$ be a security parameter.

Commit(v)

1. The sender picks random $r \in \{0,1\}^\kappa$, and the receiver a $c \in \{0,1\}^\kappa$.
2. The sender inputs $x_0 = r_i$ and $x_1 = r_i \oplus v$ and the receiver c_i to the i-th instance of OT.
3. The receiver obtains y_i from the i-th instance of OT.

Open

1. The sender sends v and r to the receiver.
2. The receiver verifies whether for all i we have $y_i = r_i \oplus c_i v$.