Error-Tolerant Combiners for Oblivious Primitives

Bartosz Przydatek[1] and Jürg Wullschleger[2]

[1] Google Switzerland, (Zurich, Switzerland)
przydatek@google.com

[2] University of Bristol (Bristol, United Kingdom)
j.wullschleger@bristol.ac.uk

Abstract. A *robust combiner* is a construction that combines several implementations of a primitive based on different assumptions, and yields an implementation guaranteed to be secure if at least *some* assumptions (i.e. sufficiently many but not necessarily all) are valid.
In this paper we generalize this concept by introducing *error-tolerant* combiners, which in addition to protection against insecure implementations provide tolerance to functionality failures: an error-tolerant combiner guarantees a secure and correct implementation of the output primitive even if some of the candidates are insecure or faulty. We present simple constructions of error-tolerant robust combiners for oblivious linear function evaluation. The proposed combiners are also interesting in the regular (not error-tolerant) case, as the construction is much more efficient than the combiners known for oblivious transfer.

1 Introduction

The security of many cryptographic schemes is based on unproven assumptions about difficulty of some computational problems, like factoring integer numbers or computing discrete logarithms. Even though some standard assumptions are supported by years of extensive study and attacks, there is still no guarantee of their validity. Moreover, for many newer assumptions even such supporting evidence is missing, and so in general it is unclear how to decide which assumptions are trustworthy. Therefore, when given several implementations of some cryptographic primitive, each based on a different assumption, it is difficult to decide which implementation is the most secure one.

Robust combiners offer a method of coping with such difficulties: they take as input several candidate schemes based on different assumptions, and construct a scheme whose security is guaranteed if at least *some* candidates are secure. Such an approach 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. The concept of robust combiners is actually not so new, but a more formal treatment of robust combiners was initiated quite recently [14, 13]. Combiners for some primitives, like one-way functions or pseudorandom generators, are rather simple, while for others, e.g., for oblivious transfer, the construction of combiners seems considerably harder [13].

Most constructions of robust combiners proposed so far assume the correct functionality of the candidate implementations and focus on protecting security against wrong assumptions only.[3] Such an approach is justified by the fact that often it is possible to test or to verify the correctness of the candidates prior to their use in the combiner [13]. However, even though in principle the correctness tests and verification can be performed efficiently, in practice they require considerable effort and expert knowledge. Moreover, such a testing-based approach provides no protection when the errors of the candidates are controlled by an adversary or are caused by sporadic failures after the initial tests, e.g. due to aging hardware or external noise.

Contributions. We propose a systematic approach to coping with erroneous candidates input to a combiner. That is, we introduce *error-tolerant* combiners, which in addition to protection against insecure assumptions and implementations provide tolerance to functionality errors of the candidates. In other words, an error-tolerant robust combiner guarantees a secure and correct implementation of the output primitive even if some of the candidates are insecure or faulty. To exemplify the proposed notion of error-tolerant robust combiners we focus on two-party oblivious primitives. We present constructions of error-tolerant combiners for oblivious linear function evaluation (OLFE). The primitive of OLFE can be viewed as a generalization of oblivious transfer (OT), and so is complete for arbitrary secure distributed computations [16, 11, 28].

The proposed combiners are optimal in terms of candidates' use (they use every candidate implementation only once), and in the honest-but-curious case work as soon as the input is non-trivial. For OT, no such reduction is known [7]. Additionally, the presented OLFE-combiner is also interesting in the regular (not error-tolerant) case. In particular, the construction is much more efficient than the combiners known for oblivious transfer.

Related work. As mentioned above, there are numerous implicit uses and constructions of combiners in the literature (e.g., [1, 9, 17, 8, 15]), but a more rigorous study of robust combiners was initiated only recently, by Herzberg [14] and by Harnik *et al.* [13], who have formalized the notion of combiners, and have shown constructions of combiners for various primitives. In particular Harnik *et al.* [13] have proposed a combiner for key agreement, which tolerates some failures of the candidates (cf. Sect. 2), and also have shown that not all primitives are easy to combine. In [18] robust combiners for private information retrieval (PIR) were proposed, and also *cross-primitive* combiners have been studied, which can be viewed as generalized reductions between primitives. This generalization lead to a partial separation of PIR from OT. In [19] generalized definitions of combiners for two-party primitives have been proposed, leading to constructions strictly stronger than the ones known before, and allowing for easier impossibility proofs.

[3] A notable exception is a combiner for key agreement due to Harnik *et al.* [13], which can handle some failures of the candidates (cf. discussion in Sect. 2).

OLFE has been studied in [26], where its symmetry has been shown. Rivest [24] has shown that with one run of OLFE a very simple commitment scheme can be implemented. In a recent work [12], Harnik *et al.* improve the efficiency of previous OT-combiners, and show that it is possible to construct them with a *constant rate*. While this is a great improvement for OT-combiners, it is only *asymptotically* efficient, and even then by a (probably big) constant factor less efficient than our OLFE-combiner, which is not only much simpler, but also uses every candidate exactly *once*, which is certainly optimal.

2 Preliminaries and Definitions

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[4] (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}.

Oblivious Linear Function Evaluation (OLFE) over a finite field $\mathbb{F}$ is a natural generalization of oblivious transfer for domains larger than one bit [26], and is a special case of *oblivious polynomial evaluation* [20]. In OLFE over $\mathbb{F}$ the sender's input is a linear function $f(x) = a_1x + a_0$, where $a_0, a_1, x \in \mathbb{F}$, and the receiver's input is an argument $c \in \mathbb{F}$. The goal of OLFE is that the receiver learns the value of sender's function at the argument of his choice, i.e. he learns $y = f(c)$ (and nothing else), and the sender learns nothing. Oblivious transfer is indeed a special case of OLFE: the output bit b_c of OT with inputs (b_0, b_1) and c respectively, can be interpreted as an evaluation of the linear function $f(c) = a_1 \cdot c + a_0$ over $\mathbb{F}_2$, since

$$b_c = \underbrace{(b_0 + b_1)}_{\equiv a_1} \cdot c \; + \; \underbrace{b_0}_{\equiv a_0} \; .$$

Many protocols based on OT can be generalized to OLFE, and thereby increasing their efficiency. For more applications, see [20].

Secret sharing [3, 25] 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 k-out-of-n secret sharing, if any k correct shares are sufficient to reconstruct the secret, but any subset of less than k shares gives

[4] The version of oblivious transfer described here and used in this paper is more precisely denoted as *1-out-of-2 bit-OT* [10]. 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 [23, 4, 5].

no information about the secret. A simple method for k-out-of-n secret sharing was proposed by Shamir [25]: a party P having a secret $s \in \mathbb{F}_q$, 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) $k-1$. A share for party P_i is 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 $k-1$, any k shares are sufficient to reconstruct $f(x)$ and compute $s = f(0)$ (via Lagrange interpolation). On the other hand, any $k-1$ or fewer shares give no information about s, since they can be completed consistently to yield a sharing of any arbitrary $\overline{s} \in F[q]$, and the number of possible completions is the same for every $\overline{s}$.

Robust combiners. In this section we recall definitions of robust combiners, and present generalizations for combiners of two-party primitives, which capture constructions that can tolerate even incorrect candidates. We say that a candidate implements a primitive *correctly*, if it produces the correct output when both players are honest, and we say that a candidate is *secure* for Alice (Bob), if the action of a dishonest Bob (Alice) can be simulated in an ideal setting (for more details cf. [22]).

Definition 1 **($(k;n)$-robust $\mathcal{F}$-combiner [13]).** *Let $\mathcal{F}$ be a cryptographic primitive. A $(k;n)$-*robust $\mathcal{F}$-combiner *is a PPTM which gets n candidates implementing $\mathcal{F}$ as inputs and implements $\mathcal{F}$ while satisfying the following 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 $\mathcal{F}$.*

To capture the error-tolerance of a combiner we introduce a parameter γ, which denotes the number of candidates that are assumed to provide correct functionality. The following definition of error-tolerant combiners is based on a generalization of Definition 1, proposed recently in [19].

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

1. *If at least γ candidates implement $\mathcal{F}$* correctly*, 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 $\mathcal{F}$.*

Combiners which are $(\alpha, \beta, n; n)$-robust, that is constructions working only when all candidates provide correct functionality, are called in short just $(\alpha, \beta; n)$-*robust.* Note that any $(k, k; n)$-robust combiner is always also a $(k; n)$-robust combiner, but a $(k; n)$-robust combiner may *not* be a $(k, k; n)$-robust combiner.

Motivated by an observation that the constructions of $(\alpha, \beta; n)$-robust combiners can be "non-uniform", with explicit dependence on α, β, a stronger notion of *uniform* combiners was introduced in [19]. Intuitively, an $\{\delta; n\}$-robust *uniform* combiner is a single construction that is *simultaneously* a $(\alpha, \beta; n)$-robust combiner for all $\alpha, \beta \in \{0, \ldots, n\}$ satisfying $\alpha + \beta \geq \delta$. In particular, a uniform combiner doesn't obtain the values of α, β as parameters. The next definition generalizes the notion of uniform combiners to the error-tolerant setting.

Definition 3 ($\{\delta, \gamma; n\}$-robust uniform $\mathcal{F}$-combiner). *Let $\mathcal{F}$ be a two-party primitive. We say that an $\mathcal{F}$-combiner is a* $\{\delta, \gamma; n\}$-robust uniform $\mathcal{F}$-combiner *if it is a $(\alpha, \beta, \gamma; n)$-robust $\mathcal{F}$-combiner,* simultaneously *for all $\alpha, \beta \in \{0, \ldots, n\}$ satisfying $\alpha + \beta \geq \delta$.*

Uniform combiners which are $\{\delta, n; n\}$-robust, that is constructions working only when all candidates provide functionality, correspond to the original notion $\{\delta; n\}$-robust uniform combiners. 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: it is a (regular) $(2; 3)$-robust combiner, but at the same time it is also a $(3, 1; 3)$-robust combiner and a $(1, 3; 3)$-robust combiner.

Error tolerance of robust combiners. So far research on robust combiners focused on protection against wrong computational assumptions, and the proposed constructions usually required that the candidates input to a combiner provide the desired functionality. In general, this approach is justified by the fact that in cryptographic schemes usually the security is based on some assumptions, while the functionality properties are straightforward and hold unconditionally. Moreover, Harnik *et al.* [13] suggested that sometimes a possible way of dealing with unknown implementations of primitives is to test them for the desired functionality before combining them.

However, in many realistic scenarios this functionality assumption is not always justified. For example, if the candidate implementations are given as black-boxes, their failures might be time-dependent, e.g. caused by some transient external factors, or even controlled by an adversary, due to some malicious "features" (malware). In such a case, the testing could be successful, but the actual run within a combiner would produce wrong results. Moreover, even if the candidate implementations are given as programs or software packages, it is sometimes unreasonable to assume that a user will be able to check their functionality, as such a check requires substantial effort and expert knowledge.

A notable exception of the above functionality assumption is a robust combiner for key agreement (KA) proposed by Harnik *et al.* [13]. This combiners works by constructing a relaxed KA (where agreement is achieved with relatively high probability), and then by using it together with an error correcting code to increase the probability of agreement. However, this construction is also partially based on testing (to construct a relaxed KA), and is tolerant only against stochastic errors. In particular, the construction fails when the errors are caused by a malicious adversary which has ability of introducing arbitrary errors in a single candidate. Such an adversary can easily force incorrect final results, by causing errors at selected locations in the transmitted codeword, which lead to wrong error correction.

We suggest a more systematic approach to error tolerance. In particular, the proposed definitions of error-tolerant combiners require that constructions guarantee correctness and security of the resulting implementation even when some of the input candidates are under full control of an adversary.

3 Robust combiners for OLFE

In this section, we propose a number of robust combiners for *oblivious linear function evaluation* (OLFE). In particular, we present a $(\alpha, \beta; n)$-robust OLFE-combiner works for any $\alpha + \beta > n$. The proposed construction achieves optimal robustness and is much more efficient than the most efficient OT-combiners with the same robustness [19]: it uses any candidate instance only *once*, which is optimal, and therefore might be preferable in some scenarios. Moreover, we exploit the symmetry of OLFE to construct efficient *uniform* OLFE-combiners. Subsequently we present error-tolerant combiners for OLFE, which work even if some of the input candidates are incorrect and do not guarantee the OLFE-functionality. These combiners are optimal both terms of candidates' use, and in the achieved bound on α and β. For OT, no such reduction is known [7, 27].

3.1 OLFE-combiner

We start by presenting a robust OLFE-combiner, which does not provide error-tolerance, but is of interest for at least two reasons. First, it constitutes a basis for error-tolerant constructions presented later. Second, it is very efficient, and can be advantageous in scenarios where otherwise an OT-combiner would be used. For example, if we have a protocol for secure function evaluation based on OLFE, a protocol for OLFE based on OT, and three implementations of OT from which we assume two to be correct, it will be more efficient to use the OLFE-combiner we present in this section than the OT-combiner: the construction would require only half as many calls to each OT-instance. Furthermore, the construction is perfect (its error probability is equal zero).

Recall that OLFE is a primitive defined over some finite field $\mathbb{F}_q$, where the sender's input is a linear function $f(x) = a_1x + a_0$, with $a_0, a_1, x \in \mathbb{F}_q$, and the receiver's input is an argument $c \in \mathbb{F}_q$. The receiver learns the value of the function on his input, $y = f(c)$, and the sender learns nothing.

In our OLFE-combiner we use Shamir secret sharing scheme [25] for both players at the same time to protect the privacy of the inputs. Given inputs $f(x) := a_1x + a_0$ resp. $c \in \mathbb{F}_q$, the sender and the receiver proceed as follows. The sender picks two random polynomials $A_0(z)$ of degree $n - 1$ and $A_1(z)$ of degree $n - \alpha$, such that $A_0(0) = a_0$ and $A_1(0) = a_1$. The receiver picks a random polynomial $C(z)$ of degree $n - \beta$, such that $C(0) = c$. Then the parties evaluate

Protocol `OLFE-rc`

SENDER'S INPUT: linear function $f(x) := a_1x + a_0$, with $a_0, a_1 \in \mathbb{F}_q$
RECEIVER'S INPUT: evaluation point $c \in \mathbb{F}_q$
INPUT OLFE PROTOCOLS: $\mathsf{OLFE}_1, \ldots, \mathsf{OLFE}_n$
parameters: $n < q; \alpha$, β; distinct non-zero constants $z_1, \ldots, z_n \in \mathbb{F}_q$

1. Sender picks two random polynomials:
   $A_0(z)$ of degree $n-1$ such that $A_0(0) = a_0$, and
   $A_1(z)$ of degree $n-\alpha$ such that $A_1(0) = a_1$.
2. Receiver picks a random polynomial $C(z)$ of deg. $n-\beta$, s.t. $C(0) = c$.
3. $\forall i \in [n]$ the parties run OLFE_i, with sender holding input $f_i(x) = A_1(z_i)x + A_0(z_i)$, and receiver holding input $c_i = C(z_i)$.
4. Receiver uses the values $\{(z_1, f_1(c_1), \ldots, (z_n, f_n(c_n))\}$ to interpolate a polynomial $\tilde{h}(z)$ of degree $n-1$, and outputs $y = \tilde{h}(0)$.

Fig. 1: `OLFE-rc`: A (α,β;n)-robust OLFE-combiner for $\alpha + \beta > n$.

locally these polynomials for n distinct non-zero values $z_1, \ldots, z_n \in \mathbb{F}_q$, and use the resulting values as input to the instances of OLFE.

More precisely, we can view the two polynomials of the sender, $A_0(z)$ and $A_1(z)$ as parts of a two-dimensional polynomial $F(x, z)$ of degree 1 in x and degree $n-1$ in z, $F(x, z) := A_1(z) \cdot x + A_0(z)$, which satisfies $F(x, 0) = f(x)$. Note that for any constant $z_i \in \mathbb{F}_q$, the polynomial $f_i(x) := F(x, z_i) = A_1(z_i)x + A_0(z_i)$ is just a linear function. Furthermore we define a polynomial $h(z) := F(C(z), z) = A_1(z) \cdot C(z) + A_0(z)$, which clearly satisfies $h(0) = f(c)$, i.e., $h(0)$ is the value the receiver should obtain. Hence the goal is now to allow the receiver to learn $h(0)$. To achieve this, the parties run OLFE candidates, through which the receiver obtains sufficiently many points on $h(z)$ to enable its interpolation and the computation of $h(0)$. To evaluate through OLFE_i the polynomial $h(z)$ at any particular value $z_i \in \mathbb{F}_q$, the sender's input is $f_i(x) := F(x, z_i)$, and the receiver's input is $C(z_i)$.

Intuitively, the privacy of the users in this construction is protected by the degrees of the polynomials we use. Since we have to be prepared that in worst case only α of the n input OLFE-protocols are secure for the sender, and only β are secure for the receiver, the polynomials $A_0(z)$ and $A_1(z)$ must have degree at least $n-\alpha$, and the degree of $C(z)$ must be at least $n-\beta$. On the other hand, note that $h(z)$ is of degree $\max\{\deg(A_0), \deg(A_1)+\deg(C)\} = \max\{n-1, 2n-\alpha-\beta\}$. Since we're using n evaluation points, this degree must be at most $n-1$, otherwise interpolation is not possible. This implies, that $\alpha + \beta > n$ must be satisfied. Indeed, this construction works for any α, β with $\alpha + \beta > n$, which is optimal. Figure 1 presents the combiner in full detail, and its analysis is given below.

Lemma 1. *Protocol* `OLFE-rc` *is correct if* $\alpha + \beta > n$.

Proof. By construction, we have $f(x) = F(x, 0)$ and $c = C(0)$. Hence $f(c) = F(C(0), 0) = h(0)$. Further, we have $y_i := f_i(c_i) = f(g(z_i), z_i) = h(z_i)$. Since h has degree $\max\{n-1, 2n-\alpha-\beta\} < n$, the interpolated of y_i's will result in a polynomial $\tilde{h}(z)$ identical to $h(z)$. Hence, we have $y = \tilde{h}(0) = h(0) = f(c)$. $\square$

Lemma 2. *Protocol* `OLFE-rc` *is secure against a malicious sender if at least β input instances of OLFE are secure against a malicious sender.*

Proof. *(sketch)* Note that the values c_i form a $(n-\beta+1)$-out-of-n secret sharing of c, hence a malicious sender that sees at most $(n-\beta)$ values of c_i (of his choice) does not get any information about c. $\square$

Lemma 3. *Protocol* `OLFE-rc` *is secure against a malicious receiver if at least α input instances of OLFE are secure against a malicious receiver.*

A proof of this lemma is given in [22]. From the above lemmas, and from lower bounds on robustness of OT-combiners [19] we obtain the following theorem:

Theorem 1. *For every $n > 1$ and α, β satisfying $\alpha + \beta > n$ there exists an efficient third-party black-box $(\alpha, \beta; n)$-robust combiner for OLFE over $\mathbb{F}_q$ with $q > n$. The combiner is perfect, achieves optimal robustness, and uses only one run of each candidate instance of OLFE, which is also optimal.*

3.2 Uniform OLFE-combiner based on symmetry of OLFE

A two-party primitive is *symmetric* if it can be *logically* reversed, i.e. if any implementation of a primitive $\mathcal{F}$ between Alice and Bob it can be logically transformed into an implementation of $\mathcal{F}$ with reversed roles of Alice and Bob. Such a logical reversal differs from *physical* reversal, in which the parties just swap their roles when executing the corresponding protocol. In a recent work [19] it was shown, that the symmetry of oblivious transfer [6, 21, 26] can be used to construct efficient *universal* OT-combiners, which are based on an observation that a physical reversal of a symmetric primitive followed by a logical reversal (so-called swap-operation) yields an implementation with swapped security properties. Since OLFE is also symmetric [26], we can use the same trick as in [19] to obtain the following theorem.

Theorem 2. *For any $n > 1$ and any $\delta > n$ there exists a third-party black-box $\{\delta; n\}$-robust uniform combiner for OLFE over $\mathbb{F}_q$ with $q > 2n$, using the* swap*-operation. The combiner is perfect and uses only two runs of each candidate instance of OLFE.*

3.3 Error-tolerant OLFE-combiners

The OLFE-combiners described so far are very efficient, but break if any of the candidates is incorrect, i.e. if a candidate provides incorrect output to the receiver. In this section we present two very efficient constructions of error-tolerant OLFE combiners, which are both robust against *insecure* candidates and tolerate *erroneous* candidates. However, the proposed constructions differ in the error-tolerance and robustness bounds, and also in the guaranteed level of security. The first construction achieves better error-tolerance and robustness, but guarantees security against an honest-but-curious receiver only. The second achieves security against malicious parties, at a cost of lower error-tolerance and robustness. Both combiners are based on the combiner `OLFE-rc` from Sect. 3.1.

OLFE-combiner with honest-but-curious receiver. We modify the construction `OLFE-rc` to enable error correction while still preserving privacy of the participants. To allow for error correction (using for example Berlekamp-Welch algorithm [2]), we introduce additional redundancy in the information obtained by the receiver, by decreasing the degree of the polynomial $h(z)$. In particular, the degree of the polynomial $A_0(z)$ (which shares the coefficient a_0) is decreased by 2ε, where ε denotes the number of erroneous candidates tolerated by the combiner. This degree reduction of polynomial $A_0(z)$ in `OLFE-rc`, together with error-correction added in Step 4 (before interpolation of a polynomial $\tilde{h}(z)$ of degree $n-1-2\varepsilon$), are already all modifications we need to obtain an error-tolerant combiner. Let `OLFE-rc-hr` denote the resulting construction. Since the degree of $h(z)$ is given by $\max\{\deg(A_0), \deg(A_1)+\deg(C)\} = \max\{n-1-2\varepsilon, 2n-\alpha-\beta\}$, to ensure correction of up to ε errors it must hold that $n-1-2\varepsilon \geq 2n-\alpha-\beta$. Using $\varepsilon = n-\gamma$, this implies that successful error correction is possible if $\alpha+\beta+2\gamma > 3n$, i.e. we obtain the following lemma.

Lemma 4. *Protocol* `OLFE-rc-hr` *is correct if* $\alpha+\beta+2\gamma > 3n$.

Since the modifications introduced to construct `OLFE-rc-hr` do not affect the information that can possibly leak (due to insecure candidates) from the receiver to the sender, the security of the receiver remains unchanged.

Lemma 5. *Protocol* `OLFE-rc-hr` *is secure against a malicious sender if at least* β *input instances of OLFE are secure against a malicious sender.*

To complete the security analysis of `OLFE-rc-hr` it remains to argue the security of the sender.

Lemma 6. *Protocol* `OLFE-rc-hr` *is secure against an honest-but-curious receiver if at least* α *input instances of OLFE are secure against a malicious receiver, and if* $\alpha+\beta+2\gamma > 3n$ *holds.*

A proof of this lemma is given in [22]. From the above analysis we obtain the following theorem:

Theorem 3. *Let* $n > 1$ *and* α, β, γ *be such that* $\alpha+\beta+2\gamma > 3n$. *There exists an efficient perfect* $(\alpha, \beta, \gamma; n)$*-combiner for OLFE over* $\mathbb{F}_q$ *with* $q > n$, *secure against an honest-but-curious receiver. The combiner uses only one run of each candidate OLFE protocol.*

OLFE-combiner secure against malicious parties. It is not hard to see that combiner `OLFE-rc-hr` does not guarantee privacy of the sender when the receiver is malicious. For example,[5] when $n = 4, \alpha = 3, \beta = 4$, and $\gamma = 3$, the degree of the corresponding $h(z)$ is equal 1, i.e. only two points are needed for interpolation of $h(z)$. A malicious receiver can choose the evaluation points c_i arbitrarily (instead of a constant representing his polynomial $C(z)$ of degree $n-\beta = 0$), and so interpolate two various polynomials $h'(z)$ and $h''(z)$ of degree 1, corresponding to two values on the senders input function $f(x) = a_1 x + a_0$.

An additional simple modification is sufficient to avoid the above problem: to ensure the privacy of the sender we increase the degree of the sharing of the coefficient a_1 by 2ε. That is, the new error-tolerant OLFE-combiner secure also against malicious receivers, called `OLFE-rc-et` in the following, works as the combiner `OLFE-rc-hr`, but the degree of the polynomial $A_1(z)$ is $n - \alpha + 2\varepsilon$ rather than only $n-\alpha$. Since now the degree degree of $h(z)$ is given by $\max\{n-1-2\varepsilon, 2n-\alpha-\beta+2\varepsilon\}$, to ensure correction of up to ε errors the following condition must then hold

$$n - 1 - 2\varepsilon \geq 2n - \alpha - \beta + 2\varepsilon \ ,$$

and we obtain the following lemma.

Lemma 7. *Protocol* `OLFE-rc-et` *is correct if $\alpha + \beta + 4\gamma > 5n$.*

As previously, the modifications introduced to construct `OLFE-rc-et` do not affect the information that can possibly leak to the sender the security of the receiver remains unchanged.

Lemma 8. *Protocol* `OLFE-rc-et` *is secure against a malicious sender if at least β input instances of OLFE are secure against a malicious sender.*

Finally, we have to argue the security of the sender.

Lemma 9. *Protocol* `OLFE-rc-et` *is secure against a malicious receiver if at least α input instances of OLFE are secure against a malicious receiver, and if $\alpha + \beta + 4\gamma > 5n$ holds.*

A proof of this lemma is given in [22], and we obtain the following theorem:

Theorem 4. *Let $n > 1$ and α, β, γ be such that $\alpha + \beta + 4\gamma > 5n$. There exists an efficient perfect $(\alpha, \beta, \gamma; n)$-combiner for OLFE over $\mathbb{F}_q$ with $q > n$, secure against malicious parties. The combiner uses only one run of each candidate OLFE protocol.*

Uniform error-tolerant OLFE-combiners. As with the combiner `OLFE-rc`, we can exploit the symmetry of OLFE and use the `swap`-operation (cf. Sect. 3.2) to obtain *uniform* error-tolerant OLFE-combiners based on `OLFE-rc-hr` and `OLFE-rc-et`. For details see [22].

[5] Note, that these values satisfy the condition $\alpha + \beta + 2\gamma > 3n$.

4 Conclusions

Robust combiners are a useful tool for dealing with implementations of cryptographic primitives based on various computational assumptions. In this paper we have studied robust combiners for oblivious primitives. In particular, we have introduced error-tolerant combiners, which offer protection not only against insecure but also against erroneous candidate implementations. We have presented a number of robust combiners for OLFE, both regular and error-tolerant. The proposed constructions differ in the achieved efficiency and robustness/error-tolerance, and offer a trade-off between these two measures.

While the presented constructions of error-tolerant combiners are optimally efficient, they are not optimal in every respect. For example, there is a gap in the robustness achieved by the proposed error-tolerant OLFE-combiners secure against honest-but-curious and malicious parties. Of particular interest are also regular OLFE-combiners, as they are both optimally efficient and optimally robust. Moreover, we believe that OLFE is also interesting as a stand-alone primitive, and not only as a special case of oblivious polynomial evaluation (OPE). In particular, OLFE is symmetric, which allows for construction of uniform OLFE-combiners.

References

1. C. Asmuth and G. 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.
2. E. R. Berlekamp and L. R. Welch. Error correction for algebraic block codes, 1986. U.S. Patent 4 633 470.
3. G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, pages 313–317. American Federation of Information Processing Societies, 1979.
4. C. Crépeau. Equivalence between two flavours of oblivious transfers. In *Proc. CRYPTO '87*, pages 350–354, 1987.
5. C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *Proc. IEEE FOCS '88*, pages 42–52, 1988.
6. C. Crépeau and M. Sántha. On the reversibility of oblivious transfer. In *Advances in Cryptology—EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1991.
7. I. Damgård, J. Kilian, and L. Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Proc. EUROCRYPT '99*, pages 56–73, 1999.
8. Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *Proc. TCC '05*, pages 188–209, 2005.
9. S. Even and O. Goldreich. On the power of cascade ciphers. *ACM Trans. Comput. Syst.*, 3(2):108–116, 1985.
10. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

11. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game — a completeness theorem for protocols with honest majority. In *Proc. 19th ACM STOC*, pages 218–229, 1987.
12. D. Harnik, Y. Ishai, E. Kushilevitz, and J. B. Nielsen. OT-combiners via secure computation. In *Proc. TCC '08*, 2008.
13. D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In *Proc. EUROCRYPT '05*, pages 96–113, 2005.
14. A. Herzberg. On tolerant cryptographic constructions. In *CT-RSA*, pages 172–190, 2005. full version on Cryptology ePrint Archive, eprint.iacr.org/2002/135.
15. S. Hohenberger and A. Lysyanskaya. How to securely outsource cryptographic computations. In *Proc. TCC '05*, pages 264–282, 2005.
16. J. Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th ACM STOC*, pages 20–31, 1988.
17. U. Maurer and J. 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.
18. R. Meier and B. Przydatek. On robust combiners for private information retrieval and other primitives. In *Proc. CRYPTO '06*, pages 555–569, Aug. 2006.
19. R. Meier, B. Przydatek, and J. Wullschleger. Robuster combiners for oblivious transfer. In *Proc. TCC '07*, pages 404–418, Feb. 2007.
20. M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006.
21. R. Ostrovsky, R. Venkatesan, and M. 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.
22. B. Przydatek and J. Wullschleger. Error-tolerant combiners for oblivious primitives, 2008. full version of this paper, Cryptology ePrint Archive, eprint.iacr.org.
23. M. O. Rabin. How to exchange secrets by oblivious transfer., 1981. Tech. Memo TR-81, Aiken Computation Laboratory, available at eprint.iacr.org/2005/187.
24. R. L. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Unpublished manuscript, 1999.
25. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
26. S. Wolf and J. Wullschleger. Oblivious transfer is symmetric. In *Proc. EUROCRYPT '06*, pages 222–232, 2006.
27. J. Wullschleger. Oblivious-transfer amplification. In *Proc. EUROCRYPT '07*, 2007. Full version on arxiv.org/abs/cs.CR/0608076.
28. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *Proc. 27th IEEE FOCS*, pages 162–167, 1986.