Efficiently from Semi-honest to Malicious OT via OLFE

Jürg Wullschleger
University of Bristol, UK
j.wullschleger@bristol.ac.uk

September 1, 2009

Abstract

A combiner securely implements a functionality out of a set implementations of another functionality from which some may be insecure. We present two efficient combiners for oblivious linear function evaluation (OLFE). The first is a constant-rate OLFE combiner in the semi-honest model, the second combiner implements Rabin string oblivious transfer (RabinOT) from OLFE in the malicious model.

As an application, we show a very efficient reductions in the malicious model of RabinOT over strings to one-out-of-two oblivious transfer over bits (OT) that is only secure in the semi-honest model. For string of size $\ell = \omega(k^2)$, our reductions uses only $4\ell + o(\ell)$ instances of OT, while previous results required $\Omega(\ell k^2)$. Our new reduction leads to an efficiency improvement for general multi-party computation (MPC) based on semi-honest OT, and makes it almost as efficient as MPC based on malicious OT.

All reductions are unconditionally secure, black-box, universally composable and secure against adaptive adversaries.

1 Introduction

Secure two-party computation (introduced in [Yao82]) allows two parties to execute a computation in a secure way, without the need of a trusted third party. While for most computations this is impossible to achieve with unconditional security for both parties, any two-party computation can be implemented in a secure way if a functionality called *oblivious transfer* [Wie83, Rab81, EGL85] is available [Kil88] (see also [GMW87, GV88]). More efficient implementations have been presented in [Cré90, CvdGT95, IPS08].

In [EGL85] it has been shown how to implement one-out-of-two oblivious transfer (OT) from any (enhanced) trap-door permutation secure against a semi-honest adversary, i.e., an adversary that follows the protocol, but tries to learn additional information during the execution of the protocol. To make that protocol (or any other implementation of semi-honest OT) secure against a malicious adversary, the compiler from [GMW87] can be used. As shown in [CLOS02], this also works against adaptive adversaries. The problem if this approach is that the compiler uses the semi-honest protocol in a non-black-box way, and is therefore very inefficient. A more efficient protocol that only uses black-box access to the semi-honest OTs has been presented in [IKLP06, Hai08]. In [CDSMW09], a similar reduction is shown to be constant round and secure against adaptive adversaries in the universal composability model if the players have access to an ideal bit-commitment. To implement a single malicious OT over bits, these reductions require $\Theta(k^2)$ instances of the semi-honest OTs over bits. To implement malicious OT or RabinOT (another variant of oblivious transfer) over strings of size ℓ, at least $\Omega(\ell k^2)$ instances are needed.

Only recently, the first efficient *direct* implementations of OT have been presented in [PVW08, GWZ09] that give full security against static and adaptive adversaries, respectively.

A *combiner* (formally introduced in [Her05]), is a protocol that takes some candidate implementations of a functionality and outputs an implementation that is secure if enough of the candidates are secure. Implementations (and impossibilities) of OT-combiners have been shown in [HKN+05]. Some generalizations can be found in [MPW07]. In [HIKN08], a more efficient, *constant rate* combiner was presented, where the number of OTs implemented are a constant fraction of the number of calls to the candidates, while a constant fraction of the candidates are allowed to be faulty. While the semi-honest protocol in [HIKN08] is unconditionally secure, the protocol in the malicious model requires additional computational assumptions. A constant-rate OT-combiner that is unconditionally secure in the malicious model has finally been presented in [IPS08]. *Oblivious linear function evaluation* (OLFE) is a generalization of OT over bigger finite fields, and seems to be a functionality that is particularly well-suited for combiners. As shown in [PW08], a simple OLFE-combiner can be obtained using Shamir secret sharings [Sha79]. The resulting combiner has also the advantage that every instance of OLFE is only used once, a property that none of the OT-combiner has.

1.1 Contribution

First, we show that the OLFE-combiner from [PW08] can be generalized to a constant-rate combiner that is secure in the semi-honest model. As in [PW08], every candidate implementation is used black-box and only once. From n instances out of which s remain secure, we are able to construct

$$m = \frac{2s - n + 1}{2}$$

secure instances of OLFE. Since there cannot exist a perfect combiner for $m > 2s - n$, this is close to optimal.

Then, we present an OLFE-to-RabinOT-combiner that is secure in the malicious model. From n instances of OLFE out of which s remain secure, it implements a RabinOT over strings of length

$$\ell = \frac{2s - n + 1}{4} \cdot \log q - 2k \ ,$$

where q is the size of the field and k is the security parameter. Our protocol uses a simple privacy-amplification step to ensure that a malicious receiver does not get too much information. To show security against an adaptive adversary, we use a special (but standard) implementation of a 2-universal hash function that has a certain invertibility property[1]. Note that using the results from [DFSS06, Wul07], OT over strings could be implemented in a similar way. We have chosen RabinOT here because it is used in [IPS08], which we consider one of the main applications of our reduction.

Finally, using a protocol presented in [IPS09] that implements in the semi-honest model OLFE from OT, we show that our OLFE-to-RabinOT-combiner can be used to improve the efficiency of the black-box reduction of malicious RabinOT to semi-honest OTs. If the string length ℓ is big enough, i.e., if $\ell = \omega(\max(k^2, \log^2 1/p))$, our reductions uses only

$$4\ell + o(\ell)$$

instances of OT, which is by a factor of $\Omega(k^2)$ more efficient than [IKLP06, Hai08, CDSMW09]. Note that if ℓ is big enough and $p < 1/4$, our reduction even beats a straight-forward[2] implementation of RabinOT based on *ideal* OT (that is secure against a malicious adversary), which would require at least $2\ell \log 1/p$ instances of OT.

RabinOT over strings secure against malicious adversaries is a key ingredient of the protocols presented in [IPS08]. Our results imply that it is possible to base these protocols on semi-honest OTs instead of malicious OT at almost no loss in efficiency.

2 Preliminaries

For a random variable X over $\mathcal{X}$, we denote its distribution by P_X. For a given distribution P_{XY}, we write for the marginal distribution $P_X(x) := \sum_{y \in \mathcal{Y}} P_{XY}(x, y)$ and, if $P_Y(y) \neq 0$, $P_{X|Y}(x \mid y) := P_{XY}(x, y)/P_Y(y)$ for the conditional distribution. For any $n > 0$, let $[n] := [1, \ldots, n]$.

The *statistical distance* of two distributions P_X and $P_{X'}$ over a domain $\mathcal{X}$ is defined as

$$\delta(P_X, P_{X'}) := \frac{1}{2} \sum_{x \in \mathcal{X}} \left| P_X(x) - P_{X'}(x) \right| = \max_f \left| \Pr[f(X) = 1] - \Pr[f(X') = 1] \right| \ .$$

Let X and Y be distributed according to P_{XY}. We say that X is ε-close to uniform with respect to Y, if $\delta(P_{XY}, P_U P_Y) \leq \varepsilon$, where P_U is the uniform distribution over $\mathcal{X}$.

[1] Our proof-technique could probably also be used to show that many other protocols that use a similar privacy-amplification step are also secure against adaptive adversaries.

[2] We first implement string-OT from bit-OT using [BCW03], convert it into a RabinOT with transmission probability 1/2, and xor them $\log 1/p$ times.

2.1 Security

We use the *universal composability* security definitions from [Can01] (see also [PW01]). For simplicity, we will assume that communication channels are authenticated and always deliver all messages, and that the communication is synchronous. (The protocols and the proofs could easily be adapted to cover asynchronous and unreliable communication as well.) Furthermore, we will omit session id's and player id's, and assume that they are given additionally to any message sent, and are verified by the receiver.

A protocol π securely implements a functionality $\mathcal{F}$, if for any adversary $\mathcal{A}$, there exists an efficient simulator $\mathcal{S}$ (that runs $\mathcal{A}$ as a black-box), such that the input/output behavior of $\mathcal{S}$ interacting with $\mathcal{F}$ is *indistinguishable* to the real execution of the protocol, where $\mathcal{A}$ interacts with π. $(\mathcal{S}, \mathcal{F})$ and $(\mathcal{A}, \pi)$ are indistinguishable if for all environments $\mathcal{Z}$ that interact with the system and output a single bit denoted by $\text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ and $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, resp., we have

$$\left| \Pr[\text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = 1] - \Pr[\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} = 1] \right| \leq \varepsilon \ ,$$

for $\varepsilon \leq 2^{-k}$ where k is a security parameter. We may also write $\text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \approx \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$, and $\varepsilon = 0$, $\text{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. As there is no restriction on how $\mathcal{A}$ and $\mathcal{Z}$ communicate, $\mathcal{S}$ normally lets $\mathcal{A}$ freely talk to $\mathcal{Z}$ and simulates only the communication of π with $\mathcal{A}$.

We distinguish between the *semi-honest* and the *malicious* model. In the semi-honest model, the adversary $\mathcal{A}$ follows the protocol, but tries to find out additional information about the uncorrupted player's input/output. Here, the simulator is required to use the same input as the honest player. In the malicious model, $\mathcal{A}$ may behave arbitrarily, and the simulator can freely choose the input of the corrupted players.

The adversary can corrupt players *adaptively*, which means that the adversary can decide to corrupt a player *during* the execution of the protocol in which case he receives the whole internal view of that player.

As shown in [Can01], this security definition implies that protocols can be *composed*, which means that if a protocol π securely implements $\mathcal{F}$, then in any $\mathcal{F}$-hybrid protocol $\rho^{\mathcal{F}}$ (i.e, a protocol that has access to $\mathcal{F}$ as a black-box), $\mathcal{F}$ can be replaced by π without affecting the security of protocol ρ.

2.2 Ideal Functionalities

We will now present the functionalities we will be using in this work.

- *One-out-of-two Oblivious transfer* (OT) over string of size ℓ: The sender has inputs $(x_0, x_1) \in \{0,1\}^\ell \times \{0,1\}^\ell$, and the receiver input $c \in \{0,1\}$. After sending these values to OT, the receiver gets $y := x_c$.
- *Oblivious Linear Function Evaluation* (OLFE) over a finite field $GF(q)$: The sender chooses $a, b \in GF(q)$, and the receiver chooses $c \in GF(q)$. After sending these values to OLFE, the receiver get $d := a + b \cdot c$.
- *Rabin oblivious Transfer* ($\mathsf{RabinOT}$) over string of size ℓ and with transmission probability p: The sender has an input $x \in \{0,1\}^\ell$. After the sender has sent x to $\mathsf{RabinOT}$, the receiver get a value y, where

$$y := \begin{cases} x & \text{with probability } p \\ \Delta & \text{with probability } 1-p \ . \end{cases}$$

Sometimes, we will omit the values ℓ, p and q but they should be clear from the context. Note that OLFE over $GF(2)$ is equivalent to OT over strings of size 1, as we can write

$$y = x_c = x_0 + (x_0 \oplus x_1) \cdot c \ .$$

We will also need the following two variations of OLFE:

- OLFE^m is the functionality that runs m *parallel* instances of OLFE, which means that the receiver only gets his output once the inputs of all instances have been received.
- ROLFE is the same functionality as OLFE, with the only difference that the honest players cannot choose their input, it is chosen by the functionality uniformly at random. On the other hand, the adversary may freely choose the inputs of corrupted players.

The protocol where both players choose random inputs and send them to OLFE implements ROLFE. Furthermore, as shown in the appendix, ROLFE can be converted input OLFE using the Protocol Precomp-OLFE, which can also be used to transform OLFE^m into m instances of OLFE.

2.3 Adaptive Security for Combiners

A (s, n)-combiner[3] is a protocol that takes n instances of a functionality $\mathcal{F}$ as input, and outputs one instance of a functionality $\mathcal{G}$, which is secure if at least s instances are secure. Instead of formally defining a combiner, we define the functionality $\mathcal{F}^{(s,n)}$ that models the *input* of a combiner. A (s, n)-combiner is then a $\mathcal{F}^{(s,n)}$-hybrid protocol that securely implements $\mathcal{G}$.

- $\mathcal{F}^{(s,n)}$ is equal to n instances of $\mathcal{F}$, out of which s instances are secure. That means that during the execution of the n instances of $\mathcal{F}$, the adversary may corrupt up to $n-s$ instances of $\mathcal{F}$, one by one. The choice which instances to corrupt next may depend on what he knows so far. For every corrupted instance, he may listen to all the interaction of $\mathcal{F}$ with all the players, i.e., he receives all the inputs and outputs of all players. (In deterministic primitives such as OT or OLFE, it suffices to learn the inputs.)

Note that the adversary is allowed to corrupt instances even without corrupting players. If the two players execute n protocols that implement $\mathcal{F}$ out of which s are secure, and the adversary knows which of them are secure, then they implement $\mathcal{F}^{(s,n)}$: Using the composition theorem, we can replace all the secure instances with ideal $\mathcal{F}$'s, and let the adversary corrupt all the remaining instances at the beginning of the protocol. He gets the input and output of the players for these instances, and hence can simulate the entire view of the protocol himself. In our definition, every implementation of $\mathcal{F}$ may only be executed once. To cover the situation where the implementations of $\mathcal{F}$ can be executed several times, $\mathcal{F}$ must be replace by a functionality $\mathcal{F}^*$ that executes $\mathcal{F}$ several times.

2.4 Privacy Amplification against Adaptive Adversaries

A function $h : \mathcal{X} \times \mathcal{R} \to \mathcal{Y}$ is called a 2-*universal hash function* [CW79], if for all $x \neq x' \in \mathcal{X}$, we have

$$\Pr[h(x, R) = h(x', R)] \leq \frac{1}{|\mathcal{Y}|} \ , \tag{2.1}$$

[3] In [HKN+05], combiners were defined in a more general way. Our type was called a *third-party black-box* combiner.

if R is chosen uniform over $\mathcal{R}$. The *min-entropy* of a random variable X distributed according to P_X is defined as

$$H_{\min}(X) := -\log \max_x P_X(x) \ .$$

To implement privacy amplification — to convert a partially secure key into a secure key — we will need the *leftover hash lemma*. It shows that a two-universal hash function can be used to extract almost uniform randomness.

Lemma 1 (Leftover hash lemma [BBR88, ILL89])**.** *Let X be a random variable over $\mathcal{X}$ and let $\ell > 0$. Let $h : \mathcal{X} \times \mathcal{R} \to \{0,1\}^\ell$ be a 2-universal hash function. If $\ell \leq H_{\min}(X) - 2\log 1/\varepsilon$ and R is uniform over $\mathcal{R}$, then $h(X,R)$ is ε-close to uniform with respect to R.*

To achieve security against adaptive adversaries, we need a hash function h that is not only 2-universal, but has an additional property. We will therefore fix h to the following (standard) implementation of $h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^\ell$: To calculate $h(x,r)$, we first encode x and r in $GF(2^n)$ — choosing an arbitrary basis of $GF(2)$ in $GF(2^n)$ — and multiply them, which gives a number in $GF(2^n)$. Then, we map it back to a string in $\{0,1\}^n$, and output the least ℓ bits of this string. It is easy to see that this function satisfies condition (2.1) of a 2-universal hash function: Let $x, x' \in GF(2^n)$, $x \neq x'$, and let $d := x - x'$. For any fixed $d \neq 0$, the function $f(r) := d \cdot r$ is a permutation on $GF(2^n)$, which implies that there are exactly $2^{n-\ell}$ values $r \in GF(2^n)$ where the least ℓ bits of $f(r)$ are 0. Hence, for any $x \neq x'$, there are exactly $2^{n-\ell}$ values r where $h(x',r) = h(x,r)$, and therefore,

$$\Pr[h(x',R) = h(x,R)] = \frac{1}{2^\ell} \ ,$$

if R is chosen uniformly from $\{0,1\}^n$. Note that if X and R are chosen uniformly from $\{0,1\}^n$, then also $h(X,R)$ is uniform, which is a nice property that not all 2-universal hash functions have.

Let us now choose X uniformly over a subset $\mathcal{S} \subset \mathcal{X}$ of size at least $2^{\ell+2k}$, and R uniformly and independently over $\mathcal{R}$. Since $H_{\min}(X) \geq \ell + 2k$, Lemma 1 tells us that $Y := h(X,R)$ is 2^{-k}-close to uniform, given R. Furthermore, $P_{X|R=r,Y=y}$, the distribution of X conditioned on values r and y, is uniform among all values $x \in \mathcal{S}$ that satisfy $y = h(x,r)$. Hence, if instead we choose $\bar{Y}$ and R uniformly at random, and then choose $\bar{X}$ uniformly from $\mathcal{S}$ such that $\bar{Y} = h(\bar{X},R)$, we end up with random variables $(R,\bar{X},\bar{Y})$ which have

$$\delta(P_{R\bar{X}\bar{Y}}, P_{RXY}) \leq 2^{-k} \ ,$$

since $\delta(P_{RY}, P_{R\bar{Y}}) \leq 2^{-k}$ and $P_{\bar{X}|R\bar{Y}} = P_{X|RY}$.[4]

Whether given $\bar{y}$ and r, it is possible to *efficiently* choose a value $\bar{X}$ uniformly from a set $\mathcal{S}$ such that $\bar{y} = h(x,r)$ depends on $\mathcal{S}$ and on the function h. In our case, $\mathcal{S}$ will be defined by some linear equations that must be satisfied. But also the condition $\bar{y} = h(x,r)$ are just linear equations for our function h, so we only need to solve a system of linear equations and output a random solution of these equations. This can be done efficiently.

[4]Note that with small probability it is possible that we have values r and y such that for no x we haver $y = h(x,r)$. In this case, we can choose $\bar{X}$ arbitrarily.

3 An OLFE Combiner in the Semi-Honest Model

In [PW08], a simple combiner for OLFE based on Shamir secret sharings [Sha79] has been presented. The following protocol generalizes that protocol using ideas from [FY92] to share many secrets in a single secret sharing. The protocol uses one instance of $\mathsf{OLFE}^{(s,n)}$ as input, and implements one instance of OLFE^m. We limit the adversary to semi-honest behavior.

Let $s > m > 0$ and $n := 2s - 2m + 1$. Let $q > n + m$ and $z_1, \ldots, z_n, r_1, \ldots r_m \in GF(q)$ be distinct constants.

Protocol **OLFE-Combiner**

sender's input: $\forall i \in [m]$: $(a_i, b_i) \in GF(q) \times GF(q)$.

receiver's input: $\forall i \in [m]$: $c_i \in GF(q)$.

receiver's output: $\forall i \in [m]$: $d_i \in GF(q)$.

1. The sender picks two random polynomials:

   $A(z)$ of degree $n-1$ such that $\forall i \in [m]$: $A(r_i) = a_i$, and
   $B(z)$ of degree $n - s + m - 1$ such that $\forall i \in [m]$: $B(r_i) = b_i$.

2. The receiver picks a random polynomial $C(z)$ of degree $n - s + m - 1$, such that $\forall i \in [m]$: $C(r_i) = c_i$.

3. $\forall i \in [n]$: The parties run OLFE_i, with sender holding input $(\alpha_i, \beta_i) := (A(z_i), B(z_i))$, and receiver holding input $\gamma_i := C(z_i)$. The receiver receives $\forall i \in [n]$ values δ_i.

4. Receiver uses the n values δ_i to interpolate a polynomial $D(z)$ of degree $n-1$ such that $\forall i \in [n] : D(z_i) = \delta_i$, and outputs $\forall i \in [m] : d_i = D(r_i)$.

Lemma 2. *We have $D(z) = A(z) + B(z) \cdot C(z)$.*

Proof. The polynomial $A(z) + B(z) \cdot C(z)$ has degree $\max(n-1, 2(n-s+m-1)) = n-1$. Hence, the interpolation of n points gives the receiver $D(z) = A(z) + B(z) \cdot C(z)$. □

Lemma 3. *Given the inputs (a_i, b_i, c_i) and the output $d_i = a_i + b_i c_i$ for $i \in [m]$, the same distribution of the polynomials $A(z)$, $B(z)$, $C(z)$ and $D(z)$ as in Protocol OLFE-Combiner can be obtained, if $B(z)$ and $C(z)$ are chosen the same way, i.e. uniformly at random such that $B(r_i) = b_i$ and $C(r_i) = c_i$ and both have degree $n - s + m - 1$, $D(z)$ is chosen uniformly at random such that $D(r_i) = d_i$, and $A(z) := D(z) - B(z)C(z)$.*

Proof. The polynomial $A(z)$ is chosen uniformly at random of degree $n-1$ such that $A(r_i) = a_i$ for $i \in [m]$. Therefore, for a fixed set of $n - m$ points z_i, the values $\alpha_i = A(z_i)$ are uniform and independent of the inputs. Since $\delta_i = \alpha_i + \beta_i \gamma_i$, also the $n-m$ values δ_i are uniform and independent of the inputs. Since $D(z)$ has also degree $n-1$ and m values are fixed by $D(r_i) = d_i$, it follows that $D(z)$ is a uniform polynomial of degree $n-1$ with $D(r_i) = d_i$. The statement follows now together with Lemma 2. □

Theorem 1. *Protocol OLFE-Combiner securely implements OLFE^m in the semi-honest model, using one instance of $\mathsf{OLFE}^{(s,n)}$.*

Proof. In the following, let π be the protocol that runs in the $\mathsf{OLFE}^{(s,n)}$-hybrid model. $\mathcal{A}$ denotes the adversary in that model, and $\mathcal{Z}$ the environment. During the execution of π, $\mathcal{A}$ may corrupt $n-s$ instances of OLFE, for which he receives the inputs $(\alpha_i, \beta_i, \gamma_i)$. At some point, he may also corrupt the sender or the receiver, in which case he gets his internal state.

The simulator $\mathcal{S}$ does the following:

- If $\mathcal{A}$ corrupts the sender, $\mathcal{S}$ also corrupts the sender and gets the inputs (a_i, b_i). He chooses the polynomials $B(z)$ uniformly at random of degree $n-s+m-1$, such that for all the input values, we have $B(r_i) = b_i$ and for all instances of OLFE that have been corrupted so far, we have $B(z_i) = \beta_i$. If the receiver has been corrupted already, it chooses $A(z) := D(z) - B(z) \cdot C(z)$. Otherwise, it also chooses $A(z)$ uniformly at random of degree $n-1$ such that $A(r_i) = a_i$ and $A(z_i) = \alpha_i$ for all the corrupted instances. From $A(z)$ and $B(z)$, $\mathcal{S}$ reconstructs the sender's view and sends it to $\mathcal{A}$.

- If $\mathcal{A}$ corrupts the receiver, $\mathcal{S}$ also corrupts the receiver and gets the inputs c_i and the outputs d_i. He chooses the polynomials $C(z)$ of degree $n-s+m-1$ uniformly at random such that for all the inputs $C(r_i) = c_i$ and for all instances of OLFE that have been corrupted so far $C(z_i) = \gamma_i$. If the sender has been corrupted already, he chooses $D(z) = A(z) + B(z) \cdot C(z)$. Otherwise, he chooses a random polynomial $D(z)$ of degree $n-1$ where $D(z_i) = \alpha_i + \beta_i \cdot \gamma_i$ for all the corrupted instances of OLFE and for all the inputs $D(r_i) = d_i$. From $C(z)$ and $D(z)$, $\mathcal{S}$ reconstructs the receiver's view and sends it to $\mathcal{A}$.

- Whenever $\mathcal{A}$ corrupts an instance of OLFE, $\mathcal{S}$ sends $\mathcal{A}$ values $(\alpha_i, \beta_i, \gamma_i)$. If one or both of the players are corrupted, then these values are calculated from the polynomials, otherwise they are chosen uniformly at random.

From Lemma 2 follows that

$$\forall i \in [m] : d_i = D(r_i) = A(r_i) + B(r_i) \cdot C(r_i) = a_i + b_i \cdot c_i \ .$$

So the protocol outputs the correct values. We now distinguish the following three cases.

- If no player gets corrupted: All $n-s$ values $(\alpha_i, \beta_i, \gamma_i)$ of the corrupted instances of OLFE are uniform and independent of the m inputs values (a_i, b_i, c_i), since the degrees of $A(z)$, $B(z)$ and $C(z)$ are at least $n-s+m-1$. Hence, $\text{EXEC}_{\mathsf{OLFE}^m,\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

- If the sender gets corrupted first: Since $C(z)$ has a degree of $n-s+m-1$, the $n-s$ values γ_i of the corrupted instances are uniform and independent of the inputs c_i. It is easy to check that $\text{EXEC}_{\mathsf{OLFE}^m,\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

- If the receiver gets corrupted first: Since $B(z)$ has a degree of $n-s+m-1$, the $n-s$ values β_i of the corrupted instances are uniform and independent of the inputs b_i. With Lemma 3, it is easy to check that $\text{EXEC}_{\mathsf{OLFE}^m,\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

□

3.1 Impossibility

Protocol OLFE-Combiner is perfect, i.e., it does not have any error. For such reductions it is close to optimal, as there cannot exist a perfect OLFE combiner that is unconditionally secure for

$$m > 2s - n = n - 2(n - s) \ ,$$

i.e., where m is twice as big as in our protocol. If there were such a combiner, we could use it on $n - 2(n - s)$ secure instances, $n - s$ trivial instances that are secure for the sender and $n - s$ trivial instances that are secure for the receiver. The resulting combiner would implement $m > n - 2(n - s)$ secure instances, and therefore would give us a way to *extend* OLFE in an unconditionally secure way. Since OLFE can be build from OT without any errors in the semi-honest model (for example, using the protocol from [GV88]), and OT can be implemented from OLFE by restricting the input to 0 and 1, this would lead to a contradiction of the impossibility of extending OT shown in [Bea96].

4 A OLFE-to-RabinOT Combiner in the Malicious Model

Unfortunately, the OLFE-combiner from Section 3 is only secure in the semi-honest model. In the malicious model the reduction is not fully secure, but it does give some security against malicious adversaries: If the adversary corrupts the sender and $n - s$ instances, he does not get to know any information about the receiver's input $(c_1, \dots c_m)$, as the degree of $C(z)$ is $n - s + m - 1$. To shows a similar statement for the receiver, we must require that the input values a_i are chosen uniformly at random, which is equivalent to say that $A(z)$ must be chosen uniformly at random. Then, the values δ_i are uniform and independent of the polynomial $B(z)$. Since the degree of $B(z)$ is $n - s + m - 1$, the adversary does not get to know any information about $(b_1, \dots b_m)$. We will now show this is sufficient to implement a combiner that implements $\mathsf{RabinOT}$ from $\mathsf{OLFE}^{(s,n)}$ in the malicious model. We will first apply Protocol OLFE-combiner from Section 3, and use the resulting (partially secure) OLFE^m to implement $\mathsf{RabinOT}$. Of course, our security proof will need to give a proof against the combined, $\mathsf{OLFE}^{(s,n)}$-hybrid protocol, as Protocol OLFE-combiner is not fully secure.

Let $s > m > 0$, $n := 2s - 2m + 1$, and let $\ell := m/2 \cdot \log q - 2k$, where k is the security parameter. We require that $q > \max(n + m, \bar{p})$, where $1/\bar{p} = p$ is the probability that the receiver gets the senders input. We also require that $\bar{p} \in \mathbb{N}$, that there exists a $\hat{q}$ such that $q = 2^{\hat{q}}$, and that ℓ is a multiple of $\hat{q}$. Let $h : \{0,1\}^{\hat{q}m} \times \mathcal{R} \to \{0,1\}^{\ell}$ be the 2-universal hash function defined in Section 2.4. Note that h can also be interpreted as a randomized function that takes m elements of $GF(q)$ as input and outputs $m' := \ell/\hat{q}$ elements of $GF(q)$.

Protocol **OLFE-to-RabinOT-Combiner**

sender's input: $x \in \{0,1\}^{\ell}$.

receiver's output: $y \in \{0,1\}^{\ell} \cup \Delta$.

1. The sender chooses a random value $e \in_R [\bar{p}]$. $\forall i \in [m]$: He chooses the values $(a_i, b_i) \in_R GF(q)^2$ uniformly at random. The receiver chooses $c \in_R [\bar{p}]$ uniformly at random.
2. The sender and the receiver execute Protocol OLFE-Combiner, where for the ith instance of OLFE, $i \in [m]$, they use (a_i, b_i) and $c_i := c$ as input. The receiver gets d_i as output.

3. Let $w := (a_1 + b_1 \cdot e, \ldots, a_m + b_m \cdot e)$. The sender chooses $r \in \mathcal{R}$ at random and sends (e, r, u) to the receiver where $u := x \oplus h(w, r)$.

4. The receiver gets (e, r, u). If $e \notin [\bar{p}]$, he aborts. Otherwise, he outputs $y := u \oplus h((d_1, \ldots, d_m), r)$ if $c = e$, and $y := \Delta$ otherwise.

Theorem 2. *Protocol OLFE-to-RabinOT-Combiner uses one instance of* $\mathsf{OLFE}^{(s,n)}$ *and implements one instance of* RabinOT *that is secure against a malicious, adaptive adversary.*

The statement of Theorem 2 for the static cases follows from the following Lemmas 4 – 6. The adaptive cases are shown in Lemmas 8 – 12 in the appendix. In the following, let π be the protocol that runs in the $\mathsf{OLFE}^{(s,n)}$-hybrid model. $\mathcal{A}$ denotes the adversary in that model, and $\mathcal{Z}$ the environment. Our simulator $\mathcal{S}$ will only talk to the ideal RabinOT and $\mathcal{A}$. Note that if both players are corrupted from the beginning of the protocol, the simulation is easy.

Lemma 4. *The protocol is secure if the adversary never corrupts any player.*

Proof. The probability that $c = e$ is $p = 1/\bar{p}$. If $c = e$, it follows from Lemma 2 that $d_i = a_i + b_i \cdot c$, and hence the output is

$$y = u \oplus h((d_1, \ldots, d_m), r) = x \oplus h(w, r) \oplus h(d, r) = x \ .$$

If $c \neq e$, the output is $z = \Delta$.

All values $(\alpha_i, \beta_i, \gamma_i)$ sent by the honest players to the corrupted instances of OLFE are independent of the values (a_i, b_i, c) and (e, r, u) chosen by the players during the protocol. Since the values a_i and b_i are chosen uniformly, also the value w is uniform, and hence (for our choice of h, see Section 2.4) also the value $h(w, r)$ is uniform. It follows that the values u is uniform and independent from the inputs of the sender and the values $(\alpha_i, \beta_i, \gamma_i)$ of the corrupted instances.

Therefore the simulator just needs to choose the values $(\alpha_i, \beta_i, \gamma_i)$ at random whenever an instance gets corrupted, and choose the values (e, r, u) uniform at random. We get $\text{EXEC}_{\mathsf{RabinOT},\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. □

Lemma 5. *The protocol is secure if the adversary corrupts the sender at the beginning of the protocol.*

Proof. The adversary corrupts the sender at the beginning of the protocol. Furthermore, he corrupts at most $n - s$ instance of OLFE. For each of them, $\mathcal{A}$ receives the input γ_i from the receiver.

The simulator $\mathcal{S}$ does the following: For every instance of OLFE that $\mathcal{A}$ corrupts, he sends him a random value γ_i. Either $\mathcal{A}$ aborts, in which case also $\mathcal{S}$ aborts. Otherwise, $\mathcal{A}$ outputs n values (α_i, β_i) and (e, r, v). If $e \notin [\bar{p}]$, $\mathcal{S}$ aborts. Otherwise, he chooses $c := e$ and a random polynomial $C(z)$ such that $C(r_i) = c$ for $i \in [m]$ and $C(z_i) = \gamma_i$ for all instances that have been corrupted so far. Thereafter, $\mathcal{S}$ chooses $\gamma_i := C(z_i)$ if additional instances get corrupted. $\mathcal{S}$ then calculates $\delta_i := \alpha_i + \beta_i C(z_i)$. Then he calculates y from the values δ_i as the honest receiver would, and sends y to RabinOT.

Since the degree of $C(z)$ is $n - s + m - 1$, the $n - s$ values γ_i are uniform and independent of c. In the protocol, the receiver chooses $c = e$ with probability p, and receives the value y which has the same distribution as the value y in the simulation. It follows that $\text{EXEC}_{\mathsf{RabinOT},\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. □

Lemma 6. *The protocol is secure if the adversary corrupts the receiver at the beginning of the protocol.*

Proof. The adversary corrupts the receiver at the beginning, and at most $n - s$ instance of OLFE, from which he receivers the values (α_i, β_i).

Since the degree of $B(z)$ is $n - s + m - 1$, the $n - s$ input values β_i of the corrupted instances are random and independent of the values b_i. Furthermore, since $A(z)$ is a random polynomial of degree $n-1$ chosen independent of the values b_i and β_i, all values δ_i (since they are one-time-padded by the values α_i) are random and independent of the values β_i and b_i. Therefore, all $n - s$ values (α_i, β_i) of the corrupted instances and the s values δ_i of the uncorrupted instances are uniform and independent of the m values b_i.

Let v be the string that contains values γ_i and δ_i the adversary has sent and received to/from the uncorrupted instances of OLFE, and the values (α_i, β_i) of the corrupted instances.

Let $w_{j,i} := a_i + j \cdot b_i$, for $i \in [m]$ and $j \in [\bar{p}]$. The values γ_i determine which values of $w_{j,i}$ the adversary can calculate from v, and which are uniform given v. (One of the two must hold, it is not possible that the adversary get only partial knowledge over $w_{j,i}$.) Since for every $i \in [m]$ and any pair $j, j' \in [\bar{p}]$ with $j \neq j'$ we have $(w_{1,i} - w_{0,i})/(j - j') = b_i$, either $w_{j,i}$ or $w_{j',i}$ must be uniform given v. Therefore, the exists a value $c \in [\bar{p}]$ (that can efficiently be calculated from the values γ_i), such that for all $j \neq c$ and $j \in [\bar{p}]$, at least $m/2$ values $w_{j,i}$ are uniform given v, which implies that

$$H_{\min}((W_{j,1}, \ldots, W_{j,m}) \mid V = v) \geq m/2 \cdot \log q \; .$$

Therefore, if $e \neq c$, it follows from Lemma 1 that $h(W, R)$ is 2^{-k}-close to uniform with respect to (R, V).

The simulator $\mathcal{S}$ does the following: He simulates the honest sender in the steps 1 and 2. Whenever the adversary corrupts an instance of OLFE, he sends him the corresponding values (α_i, β_i). (Therefore, the simulator sends at most $n - s$ values (α_i, β_i) to $\mathcal{A}$, one for each corrupted instance. For each uncorrupted instance, $\mathcal{S}$ receives the value γ_i from $\mathcal{A}$ and sends the value δ_i.) From the values γ_i, $\mathcal{S}$ calculates the index c as described above. $\mathcal{S}$ receives y from RabinOT. If $y = \Delta$, $\mathcal{S}$ chooses $e \in_R \{1, \ldots, c-1, c+1, \ldots, \bar{p}\}$ and u at random. If $y \neq \Delta$, $\mathcal{S}$ chooses $e := c$ and $u := h((a_1 + c \cdot b_1, \ldots, a_m + c \cdot b_m), r) \oplus y$. $\mathcal{S}$ sends (e, r, u) to $\mathcal{A}$, where r is chosen uniformly at random. From the arguments above follows that $\text{EXEC}_{\mathsf{RabinOT},\mathcal{S},\mathcal{Z}} \approx \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. □

5 Malicious RabinOT from Semi-Honest OT

In [IPS08], a protocol for general multi-party computation based on OT has been presented that is unconditionally secure against a malicious adversary. Opposed to previous such results (for example [Kil88, Cré90, CvdGT95]), it only requires the OTs used in the protocol to be secure against a semi-honest adversary. To prevent a malicious adversary from cheating, it creates a so-called watch list at the beginning of the protocol, that allows the players to verify the behavior of the other players, without breaking the security of the protocol. To initialize the watch list, the protocol additionally needs some instances of RabinOT that are secure against a malicious adversary.

In order to implement this protocol based only on OTs secure against a semi-honest adversary, we need to implement RabinOT from such OTs. In this section, we give a new, more efficient way to do this. Our reduction is black-box and uses a simple cut-and-choose approach similar to [IKLP06, CDSMW09], but together with the combiner from Section 4. Our protocol runs in the $\mathcal{F}_{\text{com}}$-hybrid model, which means that the players have access to an ideal commitment functionality.

The following Protocol Commit-Cut-And-Choose converts n instances of semi-honest OLFE secure in the semi-honest model into one instance of $\mathsf{ROLFE}^{((1-\lambda)n,(1-2\lambda)n)}$ — $(1 - \lambda)n$ instances

of randomized OLFE, out of which at least $(1-2\lambda)n$ are secure — that is secure in the malicious model. Let $\lambda, n > 0$.

Protocol **Commit-Cut-And-Choose**

input: none.

output: the output of $(1-\lambda)n$ instances of ROLFE.

1. The sender commits n times to random strings $s'_1, \ldots, s'_n$, the receiver to random strings $r'_1, \ldots r'_n$.
2. The sender chooses random strings $s''_1, \ldots, s''_n$ and sends them to the receiver. The receiver chooses random strings $r''_1, \ldots r''_n$ and sends them to the sender.
3. They execute n times a protocol that implements OLFE in the semi-honest model, with random inputs. The randomness for the sender comes from $s_i := s'_i \oplus r''_i$ an the randomness for the receiver comes form $r_i := r'_i \oplus s''_i$.
4. Using commitments, the two players implement a coin-toss and randomly choose a set of λn instances. They open all the commitments for these instances.
5. They check if the protocols have been executed correctly. If not, they abort. Otherwise, they output the random input and the output of the remaining $(1-\lambda)n$ instances of OLFE.

Theorem 3. *Protocol Commit-Cut-And-Choose implements* $\mathsf{ROLFE}^{((1-\lambda)n,(1-2\lambda)n)}$ *in the malicious model, using* n *instances of* OLFE *that are secure in the semi-honest model, with an error of at most* $e^{-\lambda^2 n}$.

To prove Theorem 3, we need the following lemma.

Lemma 7. *If one of the players in Protocol Commit-Cut-And-Choose does not follow the protocol of at least* λn *instances of* OLFE*, the probability that the other player does not abort is at most* $e^{-\lambda^2 n}$.

Proof. Let $t \leq \lambda n$ be the number of instances where the player does not follow the protocol, and let p be the probability that he is not caught. We have $p = p_0 \cdot p_2 \cdots p_{t-1}$, where p_i is the probability that the ith instance is not in the set of size λn, given that all the previous instances are also not in that set. Since

$$p_i = \frac{n - i - \lambda n}{n} \leq \frac{n - \lambda n}{n} = 1 - \lambda \leq e^{-\lambda} ,$$

it follows that $p \leq (1-\lambda)^t \leq e^{-\lambda t} \leq e^{-\lambda^2 n}$. □

Proof Sketch of Theorem 3. If the adversary corrupts one of the players before Step 4 and does not follow the protocol of at least λn instances of OLFE, it follows from Lemma 7 that the other player will abort with probability $1 - e^{-\lambda^2 n}$. So the simulator can simply abort the protocol. If $\mathcal{A}$ corrupts less than λn instance, the simulator can corrupt the corresponding instances in $\mathsf{ROLFE}^{((1-\lambda)n,(1-2\lambda)n)}$. He can simulate the corrupted instances because he now knows their inputs, and the uncorrupted because they are secure. If the second player is corrupted at the end as well, its view can be simulated because the values of the corrupted instances are already known, and the uncorrupted instances are adaptively secure.

If the adversary corrupts both players before step 4, the simulation is easy as there is no input. If the adversary does not corrupt any of the players before step 4, all instances of OLFE are secure, and hence any later corruption can easily be simulated. □

Using the combiners from Section 4, we can now implement RabinOT, based on semi-honest implementations of OLFE. As before, ℓ is the length of the message and p is the probability that RabinOT transmits the message.

Corollary 1. *Let $n, k, \hat{q} > 0$, $q = 2^{\hat{q}}$ and $q > \max(2n, 1/p)$. There exists a protocol that uses n instances of* OLFE *over $GF(q)$ and implements a* RabinOT *over strings of length*

$$\ell \geq (n/4 - \sqrt{kn}) \cdot \hat{q} - 2k$$

with an error of at most 2^{-k}, secure against a malicious, adaptive adversary.

If we prefer to have a protocol based on OT instead of OLFE, we can use a protocol presented in [IPS09] that implements a product-share functionality $\mathcal{F}_{\text{pdt-shr}}$ from OT in the semi-honest model. It is easy to see that $\mathcal{F}_{\text{pdt-shr}}$ is equivalent to OLFE over the same field. (The only difference is that instead of choosing the share z^A at random, the sender receives it as input.) Furthermore, in our setting that does not rely on black-box rings, it is easy to show that the protocol is in fact adaptively secure.

Proposition 1 ([IPS09])**.** *There exists a protocol that implements* OLFE *over $GF(q)$ from $\log q + 2k'$ instances of* OT *with an error of $2^{-k'}$, secure against a semi-honest, adaptive adversary.*

We get the following corollary, which implies that if $\ell = \omega(\max(k^2, \log^2 1/p))$, our reductions uses only $4\ell + o(\ell)$ instances of OT.

Corollary 2. *There exists a protocol that implements a* RabinOT *over strings of length*

$$\ell \geq n/4 - 2(kn)^{\frac{2}{3}} - \sqrt[3]{n^2/k} \cdot \log n - \sqrt[3]{kn} \cdot \log 1/p - 2k$$

with an error of at most 2^{-k} that is secure against a malicious, adaptive adversary, using n instances of OT *that are secure against a semi-honest, adaptive adversary.*

Proofs of Corollary 1 and 2 can be found in the appendix.

Acknowledgment

I thank Yuval Ishai for many helpful comments. I was supported by the U.K. EPSRC, grant EP/E04297X/1.

References

[BBCS92] C. H. Bennett, G. Brassard, C. Crépeau, and H. Skubiszewska. Practical quantum oblivious transfer. In *Advances in Cryptology — CRYPTO '9c*, volume 576 of *Lecture Notes in Computer Science*, pages 351–366. Springer, 1992.

[BBR88] C. H. Bennett, G. Brassard, and J.-M. Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.

[BCW03] G. Brassard, C. Crépeau, and S. Wolf. Oblivious transfers and privacy amplification. *Journal of Cryptology*, 16(4):219–237, 2003.

[Bea95] D. Beaver. Precomputing oblivious transfer. In *Advances in Cryptology — EUROCRYPT '95*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer-Verlag, 1995.

[Bea96] D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 479–488. ACM Press, 1996.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42th Annual IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pages 136–145, 2001. Updated Version at http://eprint.iacr.org/2000/067.

[CDSMW09] S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Simple, black-box constructions of adaptively secure protocols. In *Theory of Cryptography Conference — TCC '09*. Springer-Verlag, 2009.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 494–503. ACM Press, 2002. Full version available at http://eprint.iacr.org/2002/140.

[Cré90] C. Crépeau. Verifiable disclosure of secrets and applications. In *Advances in Cryptology — EUROCRYPT 1989*, 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 '95*, Lecture Notes in Computer Science, pages 110–123. Springer-Verlag, 1995.

[CW79] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[DFSS06] I. Damgård, S. Fehr, L. Salvail, and C. Schaffner. Oblivious transfer and linear functions. In *Advances in Cryptology — CRYPTO '06*, volume 4117 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FY92] M. Franklin and M. Yung. Communication complexity of secure computation. STOC 92, 1992.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '87)*, pages 218–229. ACM Press, 1987.

[GV88] O. Goldreich and R. Vainish. How to solve any protocol problem - an efficiency improvement. In *Advances in Cryptology — CRYPTO '87*, Lecture Notes in Computer Science, pages 73–86. Springer-Verlag, 1988.

[GWZ09] J. Garay, D. Wichs, and H.-S. Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *Advances in Cryptology — CRYPTO '09*, 2009.

[Hai08] I. Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *Theory of Cryptography Conference — TCC '08*, pages 412–426, 2008.

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

[HIKN08] D. Harnik, Y. Ishai, E. Kushilevitz, and J. B. Nielsen. OT-combiners via secure computation. In *Theory of Cryptography Conference — TCC '08*, 2008.

[HKN+05] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In *Advances in Cryptology — EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113, 2005.

[IKLP06] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. Black-box constructions for secure computation. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 99–108. ACM, 2006.

[ILL89] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pages 12–24. ACM Press, 1989.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer — efficiently. In *Advances in Cryptology — CRYPTO '08*, pages 572–591. Springer-Verlag, 2008.

[IPS09] Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In *Theory of Cryptography Conference — TCC '09*. Springer-Verlag, 2009.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC '88)*, pages 20–31. ACM Press, 1988.

[MPW07] R. Meier, B. Przydatek, and J. Wullschleger. Robuster combiners for oblivious transfer. In *Theory of Cryptography Conference — TCC '07*. Springer-Verlag, 2007.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology — CRYPTO '08*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer-Verlag, 2008.

[PW01] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy (SP '01)*, page 184, 2001. Also available at http://eprint.iacr.org/2000/066.

[PW08] B. Przydatek and J. Wullschleger. Error-tolerant combiners for oblivious primitives. In *ICALP 2008*, Lecture Notes in Computer Science, 2008.

[Rab81] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

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

[Wie83] S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.

[Wul07] J. Wullschleger. Oblivious-transfer amplification. In *Advances in Cryptology — EUROCRYPT '07*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[Yao82] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS '82)*, pages 160–164, 1982.

A Appendix

A.1 Precomputing OLFE

Using a protocol from [BBCS92] it has been shown in [Bea95] that OT can be *precomputed*. The following protocol is a straight-forward generalization of that protocol to OLFE.

Protocol **Precomp-OLFE**

sender's input: $(a, b) \in GF(q) \times GF(q)$.

receiver's input: $c \in GF(q)$.

receiver's output: $d \in GF(q)$.

1. The sender and the receiver execute OLFE with random inputs $(\bar{a}, \bar{b})$ and $\bar{c}$. The receiver gets value $\bar{d} := \bar{a} + \bar{b} \cdot \bar{c}$. (Or they just execute one instance of ROLFE.)
2. The receiver sends $e := \bar{c} - c$ to the sender.
3. The sender sends $f := a + \bar{a} + \bar{b}e$ and $g := b + \bar{b}$ to the receiver.
4. The receiver outputs $d := f + g \cdot c - \bar{d}$.

It is easy to see that Protocol Precomp-OLFE is correct: We have

$$f + g \cdot c = a + \bar{a} + \bar{b}(\bar{c} - c) + (b + \bar{b})c = \bar{a} + \bar{b}\bar{c} + a + bc = \bar{d} + d \; .$$

We do not give a security proof here. Intuitively, the protocol is secure for the receiver since e does not have any information about c, as it is one-time padded with $\bar{c}$. Also, g does not give any information about b, because it is one-time padded with $\bar{b}$. f does not depend on b, and hence the receiver never gets to know b.

A.2 Proof of Theorem 2, Adaptive Cases

We will now show the adaptive cases of Thereom 2, which proves Protocol OLFE-to-RabinOT-Combiner. Note that if the sender or the receiver is corrupted during the protocol, but before before u is sent (which is the first value that depends on the input), the simulation can be done in the same way as in the case where the player is corrupted from the beginning. Therefore, it only remains to show that the protocol is secure when one of the players gets corrupted at the end of the protocol, or both players get corrupted.

Lemma 8. *The protocol is secure if the adversary corrupts the sender at the end of the protocol.*

Proof. The uncorrupted sender and receiver complete the protocol, while the adversary $\mathcal{A}$ gets to see values (e, r, u), as well as up to $n - s$ inputs $(\alpha_i, \beta_i, \gamma_i)$ from the instances of OLFE that he corrupts. Then $\mathcal{A}$ corrupts the sender, and gets to see all the internal values of the sender, including the input x. Finally, he may again corrupt some instances of OLFE, for which he gets to see $(\alpha_i, \beta_i, \gamma_i)$.

Note that u is uniform and independent of x, and the values a_i and b_i for $i \in [m]$ are uniformly distributed among the values that satisfy

$$u \oplus x = h((a_1 + e \cdot b_1, \ldots, a_m + e \cdot b_m), r) \ .$$

At first, $\mathcal{S}$ does the same as in Lemma 4. When $\mathcal{A}$ corrupts the sender, $\mathcal{S}$ also corrupts the sender and gets the input x. Using the invertibility property of h described in Section 2.4, he chooses $(a_1, \ldots, a_m)$ and $(b_1, \ldots, b_m)$ randomly such that

$$u \oplus x = h((a_1 + e \cdot b_1, \ldots, a_m + e \cdot b_m), r)$$

holds. For all the instances that get corrupted thereafter, he again chooses $(\alpha_i, \beta_i, \gamma_i)$ uniformly at random.

Using Lemma 4, it is easy to verify that the distribution in the simulation is exactly the same as in the protocol, and hence $\text{EXEC}_{\mathsf{RabinOT},\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. □

Lemma 9. *The protocol is secure if the adversary corrupts the receiver at the end of the protocol.*

Proof. The uncorrupted sender and receiver complete the protocol, while the adversary $\mathcal{A}$ gets to see values (e, r, u), as well as up to $n - s$ inputs $(\alpha_i, \beta_i, \gamma_i)$ from the instances of OLFE that he corrupts. Then $\mathcal{A}$ corrupts the receiver, and gets to see all the internal values of the receiver, including the output y. Finally, he may again corrupt some instances of OLFE, for which he gets to see $(\alpha_i, \beta_i, \gamma_i)$.

Let us condition on the values $(\alpha_i, \beta_i, \gamma_i)$ for each corrupted instance and (e, r, u) the adversary has received before corrupting the receiver, and the value y. If $y = \Delta$, then c is uniformly distributed among $\{1, \ldots, e-1, e+1, \ldots, \bar{p}\}$, and if $y \neq \Delta$, then $c = e$. Furthermore, $C(z)$ is uniformly distributed among the polynomials of degree $n - s + m - 1$ with $C(r_i) = c$ and $C(z_i) = \gamma_i$ for the corrupted instances. $D(z)$ is uniformly distributed among the polynomials that satisfy $D(z_i) = \alpha_i + \beta_i\gamma_i$ for all the corrupted instances, and

$$y = u \oplus h((D(r_1), \ldots, D(r_m)), r) \ .$$

Hence, using the invertibility property of h described in Section 2.4, it is not hard to simulate $C(z)$ and $D(z)$, and hence the entire view of the receiver. □

It remains to show that the protocol is secure if both players get corrupted at some point. Note that any corruption of instances of OLFE after both players have been corrupted is easy to simulate, as they do not reveal any additional information to the adversary.

Lemma 10. *The protocol is secure if the adversary corrupts the sender at the beginning and the receiver at the end of the protocol.*

Proof. At first, simulator $\mathcal{S}$ does the same as in Lemma 5. Note that during the simulation he constructs a polynomial $C(z)$. Then, when $\mathcal{A}$ corrupts the receiver, $\mathcal{S}$ also corrupts the receiver and get the output y. If $y = x$, he chooses $C'(z) := C(z)$. If $y = \Delta$, he chooses $c \in_R \{1, \ldots, e-1, e+1, \ldots, \bar{p}\}$ and a new polynomial $C'(z)$ uniformly of degree $n - s + m - 1$ such that $C'(r_i) = c$ for $i \in [m]$ and $C'(z_i) = \gamma_i$. Using $C'(z)$, he can construct the whole view of the receiver. It is easy to check that $\text{EXEC}_{\mathsf{RabinOT},\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. □

Lemma 11. *The protocol is secure if the adversary corrupts the receiver at the beginning and the sender at the end of the protocol.*

Proof. At first, simulator $\mathcal{S}$ does the same as in Lemma 6. Note that during this simulation, he chooses polynomials $A(z)$ and $B(z)$ and calculates a value c, and produces outputs (α_i, β_i) and (e, r, u). Then, when $\mathcal{A}$ corrupts the sender as well, $\mathcal{S}$ corrupts the sender, and gets the input x. $\mathcal{S}$ now needs to find polynomials $A'(z)$ and $B'(z)$, such that they fit to x and the view of $\mathcal{A}$. If $y = x$, he simply takes $A'(z) := A(z)$ and $B'(z) := B(z)$. If $y = \Delta$, he chooses $A'(z)$ and $B'(z)$ uniformly among all polynomials that satisfy

$$x \oplus u = h((A'(r_1) + e \cdot B'(r_1), \ldots, A'(r_m) + e \cdot B'(r_m)), r) \ ,$$

for all the insecure instances $A'(z_i) = \alpha_i$, $B'(z_i) = \beta_i$, and for all the secure instances $A'(z_i) + \gamma_i B'(z_i) = \delta_i$. Since all these constraints are linear in the coefficients of $A'(z)$ and $B'(z)$, this can be done efficiently (see also Section 2.4). From $A'(z)$ and $B'(z)$ he simulates the sender's view and sends it to the adversary. Since in the protocol u is 2^{-k}-close to uniformly distributed, we get $\text{EXEC}_{\mathsf{RabinOT},\mathcal{S},\mathcal{Z}} \approx \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. □

Lemma 12. *The protocol is secure if both players get corrupted at the end of the protocol.*

Proof. The simulator does the same simulation as in Lemma 8 for the sender, which gives him the polynomials $A(z)$ and $B(z)$ and a value e. If $y = \Delta$, $\mathcal{S}$ chooses $c \in_R \{1, \ldots, e-1, e+1, \ldots, \bar{p}\}$ and $c := e$ otherwise. Then he chooses $C(z)$ at random with degree $n - s + m - 1$ such that $C(r_i) = c$ and $C(z_i) = \gamma_i$ for the corrupted instances. With $C(z)$, $\mathcal{S}$ can simulate the view of the receiver. We get $\text{EXEC}_{\mathsf{OT},\mathcal{S},\mathcal{Z}} = \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. □

A.3 Proof of Corollary 1 and Corollary 2

Proof of Corollary 1. Protocol Commit-Cut-And-Choose implements $\mathsf{ROLFE}^{((1-\lambda)n,(1-2\lambda)n)}$ with an error of at most $e^{-\lambda^2 n}$. Using Protocol PreCompOLFE, we can implement $\mathsf{OLFE}^{((1-\lambda)n,(1-2\lambda)n)}$, which we can then use to implement $\mathsf{RabinOT}$ using Protocol OLFE-to-RabinOT-Combiner.

We choose $\lambda := \sqrt{k/n}$, which implies that $e^{-\lambda^2 n} \leq 2^{-k}$. For $s := (1-2\lambda)n$ and $n' := (1-\lambda)n$, we get

$$m = \frac{2s - n' + 1}{2} = \frac{2n - 4\lambda n - n + \lambda n + 1}{2} = \frac{n - 3\lambda n + 1}{2} \ ,$$

and

$$\ell := \frac{m}{2}\log q - 2k = \frac{n - 3\lambda n + 1}{4}\log q - 2k \geq \frac{n}{4}\log q - \frac{3}{4}\sqrt{kn}\cdot\log q - 2k\ .$$

□

Proof of Corollary 2. By choosing $k' := k + \log n'$, the total error to implement n' instances of OLFE is $n \cdot 2^{-k'} = 2^{-k}$. In total, our protocol needs

$$n := n' \cdot (\log q + 2k + 2\log n')$$

calls to OT to implement a ℓ-string $\mathsf{RabinOT}$ with an error of $3 \cdot 2^{-k}$. We get

$$\begin{aligned}\ell &\geq (n'/4 - \sqrt{kn'})\log q - 2k \\ &= n/4 - kn'/2 - n'\log n'/2 - \sqrt{kn'}\cdot\log q - 2k\ .\end{aligned}$$

We choose $\log q := \max(\sqrt{kn'}, \log 1/p)$ — where p is the transmission probability — and get

$$\ell \geq n/4 - 2kn' - n'\log n'/2 - \sqrt{kn'}\cdot\log 1/p - 2k$$

since $\log q \leq \sqrt{kn'} + \log 1/p$. Note that $q \geq 2n'$ since $\sqrt{kn'} \geq \sqrt{n'} \geq \log n' + 1$ for $n' > 1$. Since we have $n' \leq n$ and $n \geq n'\log q \geq n'\sqrt{kn'} = \sqrt{kn'^3}$, we get $n' \leq \sqrt[3]{n^2/k}$ and $kn' \leq (kn)^{\frac{2}{3}}$. The statement follows. □