Extending Oblivious Transfers Efficiently
How to get Robustness Almost for Free

Jesper Buus Nielsen*

June 6, 2007

Abstract

At Crypto 2003 Ishai *et al.* gave a protocol which given a small number of (possibly extremely inefficient) oblivious transfers implements an essentially unbounded number of oblivious transfers for an additional overhead, per oblivious transfer, of computing and sending only two hash values. This highly efficient protocol is however only passive secure. To get active security, except with probability 2^{-m}, the protocol had to suffer an additional overhead of a factor $1 + m$. We introduce a new approach to adding robustness. For practical security parameters this approach allows to add robustness while suffering only a small constant overhead over the passive secure protocol. As an example we can generate one million oblivious transfers with security 2^{-42} with an amortized cost of just 9 hash values per oblivious transfer.

1 Introduction

The notion of oblivious transfer (OT) was found by Rabin[Rab81], and independently by Weisner [Wei83] under the name of multiplexing. In a (1-out-of-2 string) OT a sender, Alice, holds two k-bit strings $x_0, x_1 \in \{0,1\}^k$ and a receiver, Bob, holds a selection bit $c \in \{0,1\}$. The protocol lets Bob learn x_c and guarantees that Alice gets no information on c and that Bob gets no information on x_{1-c}.

Since its introduction OT has found a vast number of applications, including electronic contract signing[EGL85], mental poker[Cré86], electronic voting[NSS91], zero-knowledge proofs[BM89, KMO89, SCP95], gradual release of secrets and exchange of secrets[Cle89], fair computation [GL90] and identification[CS95, FNW96]. Oblivious transfer has in fact been proven to be complete for secure two-party and multiparty computation[GMW87, GV87, Kil88, Cré89, GL90, CGT95].

However, especially the compilation results showing that OT is complete for secure two-party and multiparty computation use a vast number of applications of the underlying OT primitive when solving real-life problems. It has therefore been a major research topic to realize OT efficiently, where [CK88, FMR96, NP05, NP00, GM00, TT01, NP01, MZV02, IKNP03, Lip03, Gar04, CT05, Lip04, CS06, Lip07] is an incomplete list of notable contributions.

One important issue which has been explored is whether OT can be based on symmetric cryptography, like one-way permutations. The motivation has been that implementations based on asymmetric cryptography tends to be expensive in terms of computation and communication. Impagliazzo and Rudich[IR89] have proven that any black-box construction of OT from one-way permutations would imply a proof for P $\neq$ NP, making it unlike that we find such a construction in the near future, and so far no non-black-box construction of OT from one-way

*Funded by the Danish Agency for Science, Technology and Innovation

permutations has been found. So, with our current knowledge of complexity theory it seems that OT requires expensive asymmetric cryptographic techniques. In the breakthrough paper [Bea96] Beaver however showed that a few OTs can be extended to a larger number of OTs using one-way functions. Beaver's protocol is non-black-box and is unfortunately too inefficient to have practical applications. Later Ishai *et al.* however gave an efficient construction which extends a few OTs to an essentially unbounded number of OTs using a hash function. Their construction is highly efficient in that each produced OT only has Alice and Bob compute and send two hash values – below we say that the amortized price per OT in such a scheme is 2. This approach to efficiently constructing a vast number of OTs parallels the hybrid schemes for public-key encryption, where an expensive asymmetric cryptosystem is used to encrypt a short secret key, allowing the encryption of the bulk data to be encrypted efficiently using a symmetric encryption scheme. Unfortunately, the efficient scheme in [IKNP03] is only passive secure. To achieve active security a cut-and-choose technique is proposed. With this technique the protocol can be made secure against an active adversary except with probability 2^{-m}. The cut-and-choose technique however raises the amortized price per OT to approximately $2(1+m)$.

In the rest of the paper we describe and analyze a new and more efficient approach to adding robustness to the protocol from [IKNP03]. This new approach is not based on cut-and-choose.

In Section 2 we show how to implement a tool, which we call the ABM box, given a few OTs. In Section 3 we then phrase the passive secure protocol from [IKNP03] in terms of an ABM box, and describe why it is secure against an actively cheating Alice and a passively cheating Bob. In Section 4 we describe why this protocol is not active secure against an actively cheating Bob and make some important observations about the strategies Bob has for cheating. In Section 5 we then show how a simple test can be used to protect Alice against all cheating strategies of Bob. This test increases the amortized cost per OT from 2 to 3. As a result of protecting Alice we will however created a situation where an actively cheating Alice can learn (very) few of the selection bits of Bob. In Section 6 we show how to fix this efficiently and analyze the efficiency of the resulting scheme. As an example we can generate one million OTs with security 2^{-42} with an amortized cost per OT of just 9. And, with an amortized cost per OT of just 12 we can generate a million OTs with security 2^{-63}.

2 The ABM Box

The basis of our protocol will be a tool, which we call the ABM box. This is just an ideal functionality between Alice and Bob. Alice has as input a κ-bit vector

$$\mathbf{a} = (a^1, \ldots, a^\kappa) \in \{0,1\}^\kappa \ .$$

For $j = 1, \ldots, \ell$, Bob has as input

$$\mathbf{b}_j = (b_j^1, \ldots, b_j^\kappa) \in \{0,1\}^\kappa \ , \quad \mathbf{m}_j = (m_j^1, \ldots, m_j^\kappa) \in \{0,1\}^\kappa \ .$$

For $j = 1, \ldots, \ell$, the output of Alice is

$$\mathbf{d}_j = \mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j \ ,$$

where $(a^1, \ldots, a^\kappa) * (b^1, \ldots, b^\kappa) = (a^1 b^1, \ldots, a^\kappa b^\kappa)$.

2.1 Implementing the ABM Box

We let κ denote the security parameter and assume that we have κ OTs of κ-bit strings at our disposal. Using a pseudo-random generator this can easily be used to implement κ OTs

of ℓ-bit strings for $\ell = \text{poly}(\kappa)$. The first crucial observation is then that the ABM box can be implemented robustly using κ OTs of ℓ-bit strings. This observation was implicitly made in [IKNP03], though the construction in [IKNP03] was not phrased in terms of an underlying ABM box. Instead the below implementation of the ABM box occurs as part of the overall construction. We find it convenient to factor out the ABM box, as this part of the protocol from [IKNP03] is already robust.

Towards implementing the ABM box, note that $\mathbf{d}_j = (a^1 b_j^1 \oplus m_j^1, \ldots, a^\kappa b_j^\kappa \oplus m_j^\kappa)$. So, if we let

$$D = \begin{pmatrix} a^1 b_1^1 \oplus m_1^1 & \cdots & a^\kappa b_1^\kappa \oplus m_1^\kappa \\ a^1 b_2^1 \oplus m_2^1 & \cdots & a^\kappa b_2^\kappa \oplus m_2^\kappa \\ \vdots & \ddots & \vdots \\ a^1 b_\ell^1 \oplus m_\ell^1 & \cdots & a^\kappa b_\ell^\kappa \oplus m_\ell^\kappa \end{pmatrix} .$$

be the matrix with $\mathbf{d}_j$ as the j'th row and let $\mathbf{d}^i$ be the i'th columns in D, then we have that

$$\mathbf{d}^i = (a^i b_1^i \oplus m_1^i, \ldots, a^i b_\ell^i \oplus m_\ell^i) = a^i (b_1^i, \ldots, b_\ell^i) \oplus (m_1^i, \ldots, m_\ell^i) .$$

So, if $a^i = 0$, then

$$\mathbf{d}^i = \mathbf{m}^i ,$$

and if $a^i = 1$, then

$$\mathbf{d}^i = \mathbf{b}^i \oplus \mathbf{m}^i ,$$

where $\mathbf{b}^i = (b_1^i, \ldots, b_\ell^i)$ and $\mathbf{m}^i = (m_1^i, \ldots, m_\ell^i)$.

So, to implement the ABM box Alice and Bob run κ OTs, from Bob to Alice. In the i'th OT Alice has selection bit a^i and Bob offers messages $(X_0^i, X_1^i) = (\mathbf{m}^i, \mathbf{b}^i \oplus \mathbf{m}^i)$. Alice takes the outputs $Y^i = X_{a^i}^i$, forms a matrix D with i'th column equal to Y^i and lets $\mathbf{d}_j$ be the j'th row of this matrix.

It is straight-forward to verify that this implementation is robust. In particular, if the underlying OTs are simulatable, then also the ABM box will be simulatable. The main idea behind the proof is that there is in fact no room to cheat: For any messages $(X_0^1, X_1^1), \ldots, (X_0^\kappa, X_1^\kappa)$ offered by Bob, define $\mathbf{m}^i = X_0^i$ and $\mathbf{b}^i = X_0^i \oplus X_1^i$. Then the columns defined by Alice will be exactly $\mathbf{d}^i = a^i \mathbf{b}^i \oplus \mathbf{m}^i$, and thus the outputs will be $\mathbf{d}_j = \mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j$, as required.

3 From ABM to Many Private OTs

In [IKNP03] it is noted that the ABM box can be used to privately implement ℓ OTs. For this purpose Alice picks $\mathbf{a}$ uniformly at random and Bob picks each $\mathbf{m}_j$ uniformly at random. Each $\mathbf{b}_j$ is picked uniformly at random between the two monochrome values. I.e., first $b_j \in_R \{0,1\}$ is picked uniformly at random, and then $\mathbf{b}_j = (b_j, \ldots, b_j)$. Now the ABM box is called and Alice learns

$$\mathbf{d}_j \leftarrow \mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j = b_j \mathbf{a} \oplus \mathbf{m}_j .$$

Then Alice computes

$$\mathbf{d}_j^{(0)} \leftarrow \mathbf{d}_j , \quad \mathbf{d}_j^{(1)} \leftarrow \mathbf{d}_j \oplus \mathbf{a} . \tag{1}$$

It is then easy to see that

$$\mathbf{d}_j^{(b_j)} = \mathbf{m}_j , \quad \mathbf{d}_j^{(1-b_j)} = \mathbf{m}_j \oplus \mathbf{a} .$$

This means that $\mathbf{d}_j^{(b_j)}$ is known by Bob and $\mathbf{d}_j^{(1-b_j)}$ is completely unknown by Bob, as $\mathbf{a}$ is uniformly random and unknown to Bob. Looking at a single index, this means that we could

implement an OT of (x_j^0, x_j^1) from Alice to Bob by having Alice send $(\mathbf{c}_j^{(0)}, \mathbf{c}_j^{(1)}) = (\mathbf{d}_j^{(0)} \oplus x_j^{(0)}, \mathbf{d}_j^{(1)} \oplus x_j^{(1)})$ to Bob. Then

$$\mathbf{c}_j^{(b_j)} = \mathbf{m}_j \oplus x_j^{(b_j)} \ , \quad \mathbf{c}_j^{(1-b_j)} = \mathbf{m}_j \oplus \mathbf{a} \oplus x_j^{(1-b_j)} \ ,$$

which allows Bob to compute $x_j^{(b_j)}$ and hides $x_j^{(1-b_j)}$ perfectly because of the one-time pad encryption with $\mathbf{a}$. This argument, of course, does not work for more than one index, as $\mathbf{a}$ is common for all ℓ rows. Since

$$\begin{aligned}
\mathbf{d}_1^{(b_1)} &= \mathbf{m}_1 \ , \quad & \mathbf{d}_1^{(1-b_1)} &= \mathbf{m}_1 \oplus \mathbf{a} \\
\mathbf{d}_2^{(b_2)} &= \mathbf{m}_2 \ , \quad & \mathbf{d}_2^{(1-b_2)} &= \mathbf{m}_2 \oplus \mathbf{a} \\
&\vdots & & \\
\mathbf{d}_\ell^{(b_\ell)} &= \mathbf{m}_\ell \ , \quad & \mathbf{d}_\ell^{(1-b_\ell)} &= \mathbf{m}_\ell \oplus \mathbf{a} \ ,
\end{aligned}$$

it is, however, clear that even though Bob has a lot of information on the values $\mathbf{d}_j^{(1-b_j)}$ (e.g. $\mathbf{d}_j^{(1-b_j)} \oplus \mathbf{d}_{j'}^{(1-b_{j'})} = \mathbf{m}_j \oplus \mathbf{m}_{j'}$) Bob does not fully know any $\mathbf{d}_j^{(1-b_j)}$-value. To move from this situation of having unknown but related values $\mathbf{d}_j^{1-b_j}$ to having unknown and unrelated values a hash function is applied. For $j = 1, \ldots, \ell$ and $c = 0, 1$, let $H_j^{(c)} : \{0,1\}^\kappa \to \{0,1\}^\kappa$ be an independent hash function and let

$$\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{d}_j^{(0)}) \ , \quad \mathbf{e}_j^{(1)} = H_j^{(1)}(\mathbf{d}_j^{(1)}) \ .$$

Then

$$\begin{aligned}
\mathbf{e}_1^{(b_1)} &= H_1^{(b_1)}(\mathbf{m}_1) \ , \quad & \mathbf{e}_1^{(1-b_1)} &= H_1^{(1-b_1)}(\mathbf{m}_1 \oplus \mathbf{a}) \\
\mathbf{e}_2^{(b_2)} &= H_2^{(b_2)}(\mathbf{m}_2) \ , \quad & \mathbf{e}_2^{(1-b_2)} &= H_2^{(1-b_2)}(\mathbf{m}_2 \oplus \mathbf{a}) \\
&\vdots & & \\
\mathbf{e}_\ell^{(b_\ell)} &= H_\ell^{(b_\ell)}(\mathbf{m}_\ell) \ , \quad & \mathbf{e}_\ell^{(1-b_\ell)} &= H_\ell^{(1-b_\ell)}(\mathbf{m}_\ell \oplus \mathbf{a}) \ .
\end{aligned}$$

Clearly $\mathbf{e}_j^{(b_j)} = H_j^{(b_j)}(\mathbf{m}_j)$ can be computed by Bob, and if the $H_j^{(c)}$ are modeled as uniformly random oracles, then since Bob does not know any $\mathbf{m}_j \oplus \mathbf{a}$ and thus cannot query any $H_j^{(1-b_j)}$ on any $\mathbf{m}_j \oplus \mathbf{a}$, it follows that the values $\mathbf{e}_j^{(1-b_j)}$ are uniformly random and independent in the view of Bob. Alice can therefore use the values $\mathbf{e}_j^{(0)}$ and $\mathbf{e}_j^{(1)}$ to mask the messages in the j'th OT.

4 The Lack of Robustness

It is clear that the protocol described above is secure for Bob against a cheating Alice. In the output $\mathbf{d}_j = \mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j$ of Alice only $\mathbf{a} * \mathbf{b}_j$ depends on the selection bit of Bob. This information is however perfectly masked using the one-time pad encryption with $\mathbf{m}_j$.

The protocol is however not secure against a cheating Bob. The analysis above namely depended on Bob choosing each $\mathbf{b}_j$ monochrome. A cheating Bob can choose $\mathbf{b}_j$ as he wants. To

1. The starting configuration is that for $j = 1, \ldots, \ell$ Alice knows

$$\mathbf{d}_j^{(0)} = \mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j \ , \quad \mathbf{d}_j^{(1)} = \mathbf{a} * \overline{\mathbf{b}}_j \oplus \mathbf{m}_j$$

   for a uniformly random $\mathbf{a}$, and Bob knows $\mathbf{b}_j = (b_j, \ldots, b_j)$ and $\mathbf{m}_j$ and nothing else.

2. When Alice learns the inputs $(\mathbf{x}_1^{(0)}, \mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_\ell^{(0)}, \mathbf{x}_\ell^{(1)})$, then for $j = 1, \ldots, \ell$ Alice computes

$$\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{d}_j^{(0)}) \ , \quad \mathbf{e}_j^{(1)} = H_j^{(1)}(\mathbf{d}_j^{(1)})$$

   and sends

$$(\mathbf{g}_j^{(0)}, \mathbf{g}_j^{(1)}) = (\mathbf{e}_j^{(0)} \oplus \mathbf{x}_j^{(0)}, \mathbf{e}_j^{(1)} \oplus \mathbf{x}_j^{(1)})$$

   to Bob.

3. For $j = 1, \ldots, \ell$, Bob computes $\mathbf{x}_j^{(b_j)} = \mathbf{g}_j^{(b_j)} \oplus H_j^{(b_j)}(\mathbf{m}_j)$.

Figure 1: **Protocol I**

analyze the effect of this, it is convenient to use the notation $\overline{\mathbf{v}} = \mathbf{v} \oplus (1, \ldots, 1)$. I.e., $\overline{\mathbf{v}}$ is the bit-vector $\mathbf{v}$ with all bits flipped. Since $\mathbf{x} * (\mathbf{y} \oplus \mathbf{z}) = \mathbf{x} * \mathbf{y} \oplus \mathbf{x} * \mathbf{z}$, it follows that

$$\mathbf{a} * \mathbf{b}_j \oplus \mathbf{a} = \mathbf{a} * \mathbf{b}_j \oplus \mathbf{a} * (1, \ldots, 1) = \mathbf{a} * (\mathbf{b}_j \oplus (1, \ldots, 1)) = \mathbf{a} * \overline{\mathbf{b}}_j \ .$$

By (1) it thus follows that

$$\mathbf{d}_j^{(0)} = \mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j \ , \quad \mathbf{d}_j^{(1)} = \mathbf{a} * \overline{\mathbf{b}}_j \oplus \mathbf{m}_j \ .$$

With this notation, Alice and Bob are running the protocol in Fig. 1, where we only list the steps after the call to the ABM box. If Bob knows the messages to be sent in some OT he will learn $\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j)$ and $\mathbf{e}_j^{(1)} = H_j^{(1)}(\mathbf{a} * \overline{\mathbf{b}}_j \oplus \mathbf{m}_j)$. This is no problem when $\mathbf{b}_j$ is monochrome, as those values are then $H_j^{(b_j)}(\mathbf{m}_j)$ and $H_j^{(1-b_j)}(\mathbf{a} \oplus \mathbf{m}_j)$. If, however, Bob e.g. picks $\mathbf{b}_j = (1, 0, 0, \ldots, 0)$, then

$$\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j) = H_j^{(0)}((a_1, 0, 0, \ldots, 0) \oplus \mathbf{m}_j) \ .$$

By checking whether $\mathbf{e}_j^{(0)} = H_j^{(0)}((0, 0, 0, \ldots, 0) \oplus \mathbf{m}_j)$ or $\mathbf{e}_j^{(0)} = H_j^{(0)}((1, 0, 0, \ldots, 0) \oplus \mathbf{m}_j)$, Bob can thus compute a_1. By using $\mathbf{b}_j = (0, 1, 0, \ldots, 0)$, Bob can learn a_2 and so forth. Thus, by pick κ of the values $\mathbf{b}_j$ polychrome, Bob can learn $\mathbf{a}$. It is clear that once Bob knows $\mathbf{a}$, Bob can learn $\mathbf{e}_j^{(0)}$ and $\mathbf{e}_j^{(1)}$ for all j, allowing Bob to cheat maximally. The protocol is therefore completely insecure against a cheating Bob.

5 Protecting Alice against a Cheating Bob

In this section we describe our main contribution, which is a simple and efficient procedure for protecting Alice against a cheating Bob with very high probability.

In [IKNP03] a cut-and-choose techniques is proposed for protecting against the above cheating Bob. Many instances, $2m$, of the protocol are run in parallel and then for half of the runs, picked at random by Alice, Bob has to show that his $\mathbf{b}_j$ all are monochrome, by showing what was his inputs to the OTs. This then shows that among the remaining m executions of the OT,

at least one was run with monochrome $\mathbf{b}_j$-values, except with probability $2^{-\Theta(m)}$. This good execution gives rise to ℓ good OTs. Since it is not known which execution is good, the final OTs are combined out of several of the produced OTs: The j'th combined OT consists of one OT from each of the m parallel executions which were not opened, and these OTs are combined using the S-combiner from [CK88].

The main problem with the above approach is that to force the error probability far down (to e.g. 2^{-50}) the value of m has to be set so large that most efficiency gains are lost in practice. And, indeed [IKNP03] only suggest their techniques for settings were a relatively high error probability can be tolerated, e.g., $\frac{1}{2}$. This might for instance be a setting where a significant loss (of money or some other utility) is associated with being detected as a cheater.

We propose a new procedure for catching a cheating Bob. The test is not cut-and-choose based and reduces the error probability to exponentially low using one simple test. Indeed, the test involves sending only ℓ hash values from Alice to Bob plus a small number of bits independent of ℓ.

5.1 Advantage equals Uncertainty

The main observation is that Bob's advantage in using polychrome $\mathbf{b}_j$-vectors equals his uncertainty about the value he should have received from the j'th OT if it had been run.

Consider an index j where Bob uses a polychrome $\mathbf{b}_j$-vector, and consider the values

$$\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j) \ , \quad \mathbf{e}_j^{(1)} = H_j^{(1)}(\mathbf{a} * \overline{\mathbf{b}}_j \oplus \mathbf{m}_j) \ .$$

Bob's being able to cheat is based on the fact that he does not know any of the values $\mathbf{e}_j^{(0)}$ or $\mathbf{e}_j^{(1)}$ with certainty.

As an example, by picking $\mathbf{b}_j = (1, 0, 0, \ldots, 0)$ he certainly does not know

$$\mathbf{e}_j^{(1)} = H_j^{(1)}(\mathbf{a} * \overline{\mathbf{b}}_j \oplus \mathbf{m}_j) = H_j^{(1)}((0, a_2, a_3, \ldots, a_\kappa) \oplus \mathbf{m}_j) \ ,$$

as it depends on $\kappa - 1$ bits of the uniformly random $\mathbf{a}$. And, he does not know

$$\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j) = H_j^{(0)}((a_1, 0, 0, \ldots, 0) \oplus \mathbf{m}_j)$$

either as it depends on a_1. A priori, his uncertainty about $\mathbf{e}_j^{(0)}$ is close to 1 bit in the sense that if we asked Bob to guess $\mathbf{e}_j^{(0)}$ his best strategy would be to try to guess $a_1' = a_1$ and then submit the guess $H_j^{(0)}((a_1', 0, 0, \ldots, 0) \oplus \mathbf{m}_j)$. He would be correct with probability 2^{-1}. On the other hand, given the value $\mathbf{e}_j^{(0)}$, Bob would be able to compute a_1 efficiently, and would of course have no information on other bits of $\mathbf{a}$. Had Bob used $\mathbf{b}_j = (1, 1, 1, 0, 0, \ldots, 0)$ he would a priori be able to guess $\mathbf{e}_j^{(0)}$ with probability 2^{-3} (he would have to guess a_1, a_2, a_3), and given $\mathbf{e}_j^{(0)}$ he could use it to compute a_1, a_2, a_3 efficiently. In both cases $\mathbf{e}_j^{(1)}$ does not help Bob to compute bits of $\mathbf{a}$ as the uncertainty about the input $\mathbf{a} * \overline{\mathbf{b}}_j \oplus \mathbf{m}_j$ is too large that Bob will ever query $H_j^{(1)}$ on this input. So, if we equate Bob's advantage in using polychrome $\mathbf{b}_j$-vectors with the number of bits about $\mathbf{a}$ he can compute efficiently given $\mathbf{e}_j^{(0)}$ and $\mathbf{e}_j^{(1)}$, we see that his advantage equals his uncertainty about the value among $\mathbf{e}_j^{(0)}$ and $\mathbf{e}_j^{(1)}$ that he can guess with highest probability. We base out test on this observation.

5.2 The Test

Concretely, we will devise an efficient test where before the OTs are run Bob must show that for each $j = 1, \ldots, \ell$ he knows either $\mathbf{e}_j^{(0)}$ or $\mathbf{e}_j^{(1)}$. We do this by having Alice send $\mathbf{f}_j = \mathbf{e}_j^{(0)} \oplus \mathbf{e}_j^{(1)}$ to Bob and then letting Bob show that he can compute $\mathbf{e}_j^{(0)}$. An honest Bob can always compute $\mathbf{e}_j^{(0)}$ from $\mathbf{e}_j^{(b_j)}$ and $\mathbf{f}_j$. Since a cheating Bob trying to create a situation where m bits can be learned efficiently has m bits of uncertainty about the value out of $\mathbf{e}_j^{(0)}$ and $\mathbf{e}_j^{(1)}$ that is easiest to guess, he will pass the test with probability about 2^{-m}.

1. The starting configuration is that for $j = 1, \ldots, \ell$ Alice knows $\mathbf{d}_j^{(0)} = \mathbf{a} * \mathbf{b}_j \oplus \mathbf{m}_j$, $\mathbf{d}_j^{(1)} = \mathbf{a} * \overline{\mathbf{b}}_j \oplus \mathbf{m}_j$ for a uniformly random $\mathbf{a}$, and Bob knows $\mathbf{b}_j = (b_j, \ldots, b_j)$ and $\mathbf{m}_j$ and nothing else.
2. For $j = 1, \ldots, \ell$ Alice computes $\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{d}_j^{(0)})$, $\mathbf{e}_j^{(1)} = H_j^{(1)}(\mathbf{d}_j^{(1)})$, $\mathbf{f}_j = \mathbf{e}_j^{(0)} \oplus \mathbf{e}_j^{(1)}$ computes $E_A = (\mathbf{e}_1^{(0)}, \ldots, \mathbf{e}_\ell^{(0)})$ and sends $F = (\mathbf{f}_1, \ldots, \mathbf{f}_\ell)$ to Bob.
3. For $j = 1, \ldots, \ell$ Bob computes $\mathbf{e}_j^{(b_j)} = H_j^{(b_j)}(m_j)$, $\mathbf{e}_j^{(1-b_j)} = \mathbf{f}_j \oplus \mathbf{e}_j^{(b_j)}$, computes $E_B = (\mathbf{e}_1^{(0)}, \ldots, \mathbf{e}_\ell^{(0)})$ and sends $h_B = H(E_B)$ to Alice.
4. If $h_B \neq H(E_A)$, then Alice terminates. Otherwise, when Alice learns the inputs $(\mathbf{x}_1^{(0)}, \mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_\ell^{(0)}, \mathbf{x}_\ell^{(1)})$, then for $j = 1, \ldots, \ell$ Alice computes $\tilde{\mathbf{e}}_j^{(0)} = \tilde{H}_j^{(0)}(\mathbf{d}_j^{(0)})$, $\tilde{\mathbf{e}}_j^{(1)} = \tilde{H}_j^{(1)}(\mathbf{d}_j^{(1)})$ and sends $(\mathbf{g}_j^{(0)}, \mathbf{g}_j^{(1)}) = (\tilde{\mathbf{e}}_j^{(0)} \oplus \mathbf{x}_j^{(0)}, \tilde{\mathbf{e}}_j^{(1)} \oplus \mathbf{x}_j^{(1)})$ to Bob.
5. For $j = 1, \ldots, \ell$, Bob computes $\mathbf{x}_j^{(b_j)} = \mathbf{g}_j^{(b_j)} \oplus \tilde{H}_j^{(b_j)}(\mathbf{m}_j)$.

Figure 2: **Protocol II**

It is clear that having sent $\mathbf{f}_j = \mathbf{e}_j^{(0)} \oplus \mathbf{e}_j^{(1)}$ to Bob, Alice cannot later use $(\mathbf{g}_j^{(0)}, \mathbf{g}_j^{(1)}) = (\mathbf{e}_j^{(0)} \oplus \mathbf{x}_j^{(0)}, \mathbf{e}_j^{(1)} \oplus \mathbf{x}_j^{(1)})$ to do an OT of $(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)})$ to Bob. To fix this, new values $\tilde{\mathbf{e}}_j^{(c)} = \tilde{H}_j^{(c)}(\mathbf{d}_j^{(b)})$ are used, where $\tilde{H}_j^{(c)}$ is a fresh, independent uniformly random oracle. For efficiency, the test that Bob knows each $\mathbf{e}_j^{(0)}$ is performed by sending the hash of these values to Alice, using yet another independent hash function H. The details are given in Fig. 2.

5.3 Concrete Security Analysis

In this section we analysis the concrete security of the proposed scheme against a cheating Bob in the random oracle model. In doing this we measure the running time of Bob in how many times he queries a random oracle. By breaking the scheme we mean that Bob should create a view of the protocol such that for some j neither $\tilde{\mathbf{e}}_j^{(0)}$ nor $\tilde{\mathbf{e}}_j^{(1)}$ is uniformly random in the view of Bob. We use $\text{Success}_B(q', q)$ to denote the probability that the Bob B creates such a view while using at most q' queries before h_B is sent and using at most q queries in total. We let $\text{Success}(q', q) = \max_B \text{Success}_B(q', q)$, and we are going to prove that

$$\text{Success}(q', q) \leq ((q'/2)^2 + 2(q' + 1) + q)2^{-\kappa} .$$

If we let $q' = 2^{t'}$ and $q = 2^t$, this means that $\text{Success}(q', q) < 2\max(2^{2t'-1}, 2^t)2^{-\kappa} = \max(2^{2t'-\kappa}, 2^{t+1-\kappa})$. As an example, if we assume that Bob can make at most 2^{50} queries to the hash function during the execution of the protocol and that Bob is able to make at most 2^{99} queries in the time span

in which we want the security to hold, then using $\kappa = 160$ OTs as seed and using a hash function with output length 160 we get that the scheme is secure, except with probability 2^{-60}.

1. The starting configuration is that for $j = 1, \ldots, \ell$ Alice knows $\mathbf{d}_j^{(0)} = \mathbf{a} * \mathbf{b}_j$ and $\mathbf{d}_j^{(1)} = \mathbf{a} * \overline{\mathbf{b}}_j$ for a uniformly random $\mathbf{a}$. The values $\mathbf{b}_j$ are chosen by Bob.
2. For $j = 1, \ldots, \ell$ Alice computes $\mathbf{e}_j^{(0)} = H_j^{(0)}(\mathbf{d}_j^{(0)})$, $\mathbf{e}_j^{(1)} = H_j^{(1)}(\mathbf{d}_j^{(1)})$, and $\mathbf{f}_j = \mathbf{e}_j^{(0)} \oplus \mathbf{e}_j^{(1)}$ and $E_A = (\mathbf{e}_1^{(0)}, \ldots, \mathbf{e}_\ell^{(0)})$ and sends $F = (\mathbf{f}_1, \ldots, \mathbf{f}_\ell)$ to Bob.
3. Bob sends a value h_B to Alice.
4. If $h_B \neq H(E_A)$, then Bob loses the game. Otherwise, for $j = 1, \ldots, \ell$, Alice sends $\tilde{\mathbf{e}}_j^{(0)} = \tilde{H}_j^{(0)}(\mathbf{d}_j^{(0)})$ and $\tilde{\mathbf{e}}_j^{(1)} = \tilde{H}_j^{(1)}(\mathbf{d}_j^{(1)})$ to Bob.
5. When Bob terminates, if for some j Bob queried $\tilde{H}_j^{(0)}$ on $\mathbf{d}_j^{(0)}$ and Bob queried $\tilde{H}_j^{(1)}$ on $\mathbf{d}_j^{(1)}$, then Bob wins the game. Otherwise, Bob loses the game.

Figure 3: **Game I**

3. Bob specifies a value E_B.
4. If $E_B \neq E_A$, then Bob loses the game. Otherwise, for $j = 1, \ldots, \ell$ Bob is given the oracles $J_j^{(0)}(Q)$ and $J_j^{(1)}(Q)$.
5. When Bob terminates, if for some j Bob queried $J_j^{(0)}$ on $\mathbf{d}_j^{(0)}$ and Bob queried $J_j^{(1)}$ on $\mathbf{d}_j^{(1)}$, then Bob wins the game. Otherwise, Bob loses the game.

Figure 4: **Game II**

3. For $j = 1, \ldots, \ell$, Bob specifies $(c_j, \mathbf{d}_j)$.
4. If for some j it does not hold that $\mathbf{d}_j = \mathbf{d}_j^{(c_j)}$, then Bob loses the game. Otherwise, for $j = 1, \ldots, \ell$ Bob is given the oracles $J_j^{(0)}(Q)$ and $J_j^{(1)}(Q)$.

Figure 5: **Game III**

2. For $j = 1, \ldots, \ell$ Bob is given two oracles $I_j^{(0)}(Q)$ and $I_j^{(1)}(Q)$. As long as either $I_j^{(0)}(Q)$ never was queried on $\mathbf{d}_j^{(0)}$ or $I_j^{(1)}(Q)$ never was queried on $\mathbf{d}_j^{(1)}$ each oracle returns 0 to all queries. Otherwise, the oracle return $\mathbf{a}$.

Figure 6: **Game IV**

To help analyze Success(q', q) we create a simple game with at least the same success probability for Bob. In this game, we think of Bob as running in Fig. 2 and to help Bob maximally we give

1. The starting configuration is that for $j = 1, \ldots, \ell$ Alice knows $\mathbf{d}_j^{(0)} = \mathbf{a} * \mathbf{b}_j$ and $\mathbf{d}_j^{(1)} = \mathbf{a} * \overline{\mathbf{b}}_j$ for a uniformly random $\mathbf{a}$. The values $\mathbf{b}_j$ are chosen by Bob.
2. For $j = 1, \ldots, \ell$, Bob specifies $(c_j, \mathbf{d}_j)$.
3. If for some j it does not hold that $\mathbf{d}_j = \mathbf{d}_j^{(c_j)}$, then the game terminates with outcome 0. Otherwise, for $j = 1, \ldots, \ell$ Bob is given the oracle $J_j^{(1-c_j)}(Q)$.
4. When Bob terminates, if for some j Bob queried $J_j^{(1-c_j)}$ on $\mathbf{d}_j^{(1-c_j)}$, then the outcome of the game is 1. Otherwise, the outcome of the game is 1.

Figure 7: **Game IV**

him all the message $(\mathbf{x}_j^{(0)}, \mathbf{x}_j^{(1)})$, which is equivalent to using $\mathbf{x}_j^{(c)} = 0^\kappa$ for $j = 1, \ldots, \ell; c = 0, 1$. It can be seen that the choice of the values $\mathbf{m}_j$ do not matter for the optimal strategy of Bob, essentially because $Q \mapsto H(Q)$ being a random oracle implies that $Q \mapsto H(Q \oplus m)$ is a random oracle, and given m Bob can simulate one oracle given the other using the same number q of queries. To make the game simpler we therefore assume that $\mathbf{m}_j = 0^\kappa$ for $j = 1, \ldots, \ell$. To make one last simplification, we note that because $\tilde{\mathbf{e}}_j^{(c)} = \tilde{H}_j^{(c)}(\mathbf{d}_j^{(c)})$ is uniformly random in the view of Bob until $\tilde{H}_j^{(c)}$ is queried on $\mathbf{d}_j^{(c)}$, we can make the winning condition be that for some j Bob queried $\tilde{H}_j^{(0)}$ on $\mathbf{d}_j^{(0)}$ and queried $\tilde{H}_j^{(1)}$ on $\mathbf{d}_j^{(1)}$. The details are given in Fig. 3, and by construction we have that

$$\text{Success}(q', q) \leq \text{Success}_I(q', q) \ ,$$

where $\text{Success}_I(q', q)$ is the maximal probability that some Bob wins Game I using q' queries before sending h_B and q queries in total.

To Game II To aid the analysis we further change the game. First, we implement the equality test between E_A and E_B ideally, by requiring Bob to specify E_B. He then looses the game if $E_B \neq E_A$. Second, we replace $\tilde{H}_j^{(c)}$ by an oracle $J_j^{(c)}(Q)$ which returns 1 on $Q = \mathbf{d}_j^{(c)}$ and returns 0 otherwise. The changes from Game I are given in Fig. 4. We have that

$$\text{Success}_I(q', q) \leq \text{Success}_{II}(q', q) + (q' + 1)2^{-\kappa} \ , \tag{2}$$

where q' in Game II is an upper bound on the number of queries that Bob makes before the test $E_A = E_B$.

We show this bound by simulating any Bob B for Game I using a Bob B' for Game II. The Bob B' simply runs B as if it was in Game I, but letting it use the oracles $H_j^{(c)}$ from Game II until the test. At the test, when B sends h_B, then B' specifies E_B as follows: If h_B is not the result of querying H on some Q, then use $E_B = 0$. Otherwise, pick the first input Q to H resulting in output $H(Q) = h_B$ and let $E_B = Q$. After the test, B' simulates to B the oracles $\tilde{H}_j^{(c)}$ and the values $\tilde{\mathbf{e}}_j^{(c)} = \tilde{H}_j^{(c)}(\mathbf{d}_j^{(c)})$ using the oracles $J_j^{(c)}$, as follows: Simply let each $\tilde{\mathbf{e}}_j^{(c)}$ be uniformly random. Then on each query Q from B to $\tilde{H}_j^{(c)}$, let $\tilde{H}_j^{(c)}(Q)$ be uniformly random if $J_j^{(c)}(Q) = 0$ and let $\tilde{H}_j^{(c)}(Q) = \tilde{\mathbf{e}}_j^{(c)}$ if $J_j^{(c)}(Q) = 1$. This gives a perfect simulation, using the same number of queries. And, if $\tilde{H}_j^{(0)}(Q)$ is queried on $\mathbf{d}_j^{(0)}$ and $\tilde{H}_j^{(1)}(Q)$ is queried on $\mathbf{d}_j^{(1)}$, then the simulation ensures that $J_j^{(0)}(Q)$ is queried on $\mathbf{d}_j^{(0)}$ and $J_j^{(1)}(Q)$ is queried on $\mathbf{d}_j^{(1)}$. So, if B would have won Game I in Step 5, then B' wins Game II in Step 5.

To analyze the simulation, we look at three disjoint events. Event 1 is that $h_A \neq h_B$. Event 2 is that $h_A = h_B$ and that B never queried H on E_A. Event 3 is that $h_A = h_B$ and that B queried H on E_A. These events are well-defined in both games and have the same probability in both games. We let p_c be the probability that event c occurs, we let s_c be the probability that B wins in Game I given that event c occurs, and we let s'_c be the probability that B' wins in Game II given that event c occurs. Clearly $s_1 = 0$ as $h_A \neq h_B$ makes B loose. So, $\text{Success}_I(q', q) = p_2 s_2 + p_3 s_3 \leq p_2 + p_3 s_3$. Clearly $\text{Success}_{II}(q', q) \geq p_3 s'_3$.

We first look at p_2. In event 2 we have that $h_A = h_B$ and that B never queried H on E_A. Since H is a uniformly random oracle, $h_A = H(E_A)$ is uniformly random in the view of B when it did not query H on E_A. Therefore $p_2 \leq 2^{-\kappa}$.

We then look at $p_3 s'_3$. In event 3 we have that $h_A = h_B$ and that B queried H on E_A, which then must have given the output $H(E_A) = h_A = h_B$. So, in this case B' uses $E_B = Q = E_A$ unless it happened that before B queried H on E_A, it queried H on some $Q \neq E_A$ for which it also holds that $H(Q) = H(E_A)$. Since H is a uniformly random oracle it, however, holds for each $Q \neq E_A$ that $H(Q) = H(E_A)$ with an independent probability of $2^{-\kappa}$. Since B made at most q' queries to H before making the query on E_A it therefore follows that the probability that $E_B \neq E_A$ in this case is at most $q'2^{-\kappa}$. And, since B' wins the game iff B wins the game when $E_B = E_A$, it follows that $s'_3 \geq s_3 - q'2^{-\kappa}$. Combining these observations, (2) follows.

To Game III We then further change the game by requiring that Bob guesses some $\mathbf{d}_j^{(c_j)}$ in the test, as opposed to guessing $\mathbf{e}_j^{(0)}$. The changes from Game II are given in Fig. 5. We have that

$$\text{Success}_{II}(q', q) \leq \text{Success}_{III}(q', q) + (q' + 1)2^{-\kappa} \ .$$

To see this, consider some Bob B for game II. We simulate it in Game III by running some Bob B'. Bob B' runs B using the oracles and inputs of Game III, except that when B specifies the $\mathbf{e}_j$ values, the value $(c_j, \mathbf{d}_j)$ is specified as follows: If $\mathbf{e}_j$ was ever output by $H_j^{(0)}$, then pick the first query Q for which $H_j^{(0)}(Q) = \mathbf{e}_j$, and use $(0, Q)$. Otherwise, if $\mathbf{e}_j \oplus \mathbf{f}$ was ever output by $H_j^{(1)}$, then pick the first query Q for which $H_j^{(1)}(Q) = \mathbf{e}_j \oplus \mathbf{f}_j$, and use $(1, Q)$. Otherwise, use $(c_j, \mathbf{d}_j) = (0, 0)$.

To analyze the simulation, let $\mathcal{E}$ be the event that for some j, B neither queried $H_j^{(0)}$ on $\mathbf{d}_j^{(0)}$ nor queried $H_j^{(1)}$ on $\mathbf{d}_j^{(1)}$, and yet $\mathbf{e}_j = \mathbf{e}_j^{(0)}$. It is easily seen that $\mathbf{e}_j^{(0)}$ is uniformly random in the view of a B which neither queried $H_j^{(0)}$ on $\mathbf{d}_j^{(0)}$ nor queried $H_j^{(1)}$ on $\mathbf{d}_j^{(1)}$. From this it follows that the probability of $\mathcal{E}$ is no more than $2^{-\kappa}$. It is therefore sufficient to show that $\text{Success}_{II}(q', q) \leq \text{Success}_{III}(q', q) + q'2^{-\kappa}$ when $\mathcal{E}$ does not occur. That $\mathcal{E}$ does not occur means that for each j it holds that either $\mathbf{e}_j \neq \mathbf{e}_j^{(0)}$ or $H_j^{(0)}$ was queried on $\mathbf{d}_j^{(0)}$ or $H_j^{(1)}$ was queried on $\mathbf{d}_j^{(1)}$. If $\mathbf{e}_j \neq \mathbf{e}_j^{(0)}$ for some j, then $\text{Success}_{II}(q) = 0$ and the bound is trivial, so assume that $\mathbf{e}_j = \mathbf{e}_j^{(0)}$ for each j and that for each j either $H_j^{(0)}$ was queried on $\mathbf{d}_j^{(0)}$ or $H_j^{(1)}$ was queried on $\mathbf{d}_j^{(1)}$. If $H_j^{(0)}$ was queried on $\mathbf{d}_j^{(0)}$, then because $H_j^{(0)}(\mathbf{d}_j^{(0)}) = \mathbf{e}_j^{(0)} = \mathbf{e}_j$, it follows that B' uses a correct $(0, Q) = (0, \mathbf{d}_j^{(0)})$, unless some previous query $Q' \neq \mathbf{d}_j^{(0)}$ resulted in output $H_j^{(0)}(Q') = H_j^{(0)}(\mathbf{d}_j^{(0)})$. Otherwise, if $H_j^{(1)}$ was queried on $\mathbf{d}_j^{(1)}$, then because $H_j^{(1)}(\mathbf{d}_j^{(1)}) = \mathbf{e}_j^{(1)} = \mathbf{e}_j^{(0)} \oplus \mathbf{f}_j = \mathbf{e}_j \oplus \mathbf{f}_j$, it follows that B' uses the correct $(1, Q) = (1, \mathbf{d}_j^{(1)})$, unless some previous query $Q' \neq \mathbf{d}_j^{(1)}$ resulted in output $H_j^{(1)}(Q') = H_j^{(1)}(\mathbf{d}_j^{(1)})$. Since all oracles are uniformly random it again follows that the guess of B' is incorrect due to a previous "collision" can be bounded by $q'2^{-\kappa}$, which proves the bound.

To Game IV We then replace the oracles $H_j^{(0)}$ and $H_j^{(1)}$ by oracles $I_j^{(0)}$ and $I_j^{(1)}$, as detailed in Fig. 6. It is easy to see that

$$\text{Success}_{III}(q', q) \leq \text{Success}_{IV}(q', q) \ ,$$

as a Bob for Game III can use the oracles to simulate the hash functions: In the initial phase it simply lets each $\mathbf{f}_j$ be a uniformly random value. On a query Q to $H_j^{(c)}$ it inputs Q to $I_j^{(c)}$. If the output is 0 it lets $H_j^{(c)}(Q)$ be a uniformly random input. If the output is $\mathbf{a}$ it uses $\mathbf{a}$ to find the input $\mathbf{d}_j^{(0)} = \mathbf{a} * \mathbf{b}_j$ to $H_j^{(0)}$ and the input $\mathbf{d}_j^{(1)} = \mathbf{a} * \overline{\mathbf{b}}_j$ to $H_j^{(1)}$ and then defines the one that was just queried to be consistent with $\mathbf{f}_j$. I.e., if e.g. $H_j^{(0)}$ was just queried on $Q = \mathbf{d}_j^{(0)}$, it sets $H_j^{(0)}(Q)$ to be $H_j^{(1)}(\mathbf{d}_j^{(1)}) \oplus \mathbf{f}_j$.

To Game V Finally, we take away the oracles $I_j^{(c)}$ and the oracles $J_j^{(c_j)}$. The details are in Fig. 7. We have that

$$\text{Success}_{IV}(q', q) \leq \text{Success}_{V}(q) + (q'/2)^2 2^{-\kappa} \ .$$

It is clear that Bob can simulate $J_j^{(c_j)}$ himself: If the test fails, Bob never has access to $J_j^{(c_j)}$ and will therefore not miss it. If the test succeeds, then $\mathbf{d}_j = \mathbf{d}_j^{(c_j)}$ and Bob can easily simulate $J_j^{(c_j)}$ by letting $J_j^{(c_j)}(Q) = 1$ iff $Q = \mathbf{d}_j$. After the test, Bob can simulate the oracles $I_j^{(c)}(Q)$ by using queries to $J_j^{(c)}(Q)$ instead. This will not increase the total number of queries made by Bob. Before the test, Bob can simulate the oracles $I_j^{(c)}(Q)$ simply by returning 0 on all queries. This simulation only fails if at some point some $I_j^{(0)}(Q)$ was queried on $\mathbf{d}_j^{(0)}$ and some $I_j^{(1)}(Q)$ was queried on $\mathbf{d}_j^{(1)}$. To estimate the probability of this, consider some j and let $q_j^{(c)}$ be the number of queries made to $I_j^{(c)}$. The simulation fails if $I_j^{(0)}(Q)$ was queried on $Q = \mathbf{d}_j^{(0)} = \mathbf{a} * \mathbf{b}_j$ and $I_j^{(1)}(Q)$ was queried on $Q = \mathbf{d}_j^{(1)} = \mathbf{a} * \overline{\mathbf{b}}_j$. Let o_j be the number of 1s in $\mathbf{b}_j$. The value $\mathbf{a} * \mathbf{b}_j$ is uniformly random among 2^{o_j} values. So, the probability that $I_j^{(0)}(Q)$ was queried on $Q = \mathbf{a} * \mathbf{b}_j$ is at most $q_j^{(0)} 2^{-o_j}$. The value $\mathbf{a} * \overline{\mathbf{b}}_j$ is uniformly random among $2^{\kappa - o_j}$ values. So, the probability that $I_j^{(1)}(Q)$ was queried on $Q = \mathbf{a} * \overline{\mathbf{b}}_j$ is at most $q_j^{(1)} 2^{o_j - \kappa}$. Since $\mathbf{a} * \mathbf{b}_j$ and $\mathbf{a} * \overline{\mathbf{b}}_j$ are independent, the probability that both oracles were queried on their special input is less than $q_j^{(0)} 2^{-o_j} q_j^{(1)} 2^{o_j - \kappa} = q_j^{(0)} q_j^{(1)} 2^{-\kappa}$. So, the total failure probability is no more than $2^{-\kappa} \sum_j q_j^{(0)} q_j^{(1)}$. Since $\sum_j (q_j^{(0)} + q_j^{(1)}) = q$ for a fixed q, this expressing is seen to be maximal when $q_j^{(0)} + q_j^{(1)} = q$ for some j. I.e., all queries are made to just two oracles $I_j^{(0)}$ and $I_j^{(1)}$. So, the error probability is bounded by $(q'/2)^2 2^{-\kappa}$.

Analyzing Game V In Game V it is now clear that the price for a good oracle $J_j^{(1-c_j)}(Q)$ is a low probability of getting it. Formally, let p' be the number of bits of $\mathbf{a}$ which Bob tries to guess using the values $\mathbf{b}_j$, c_j and $\mathbf{d}_j$. The probability that Bob passes the test is at most $2^{-p'}$. Now, after the test, since the values $\mathbf{d}_j^{(c_j)}$ depend only on p' bits of $\mathbf{a}$, each of the values $\mathbf{d}_j^{(1-c_j)}$ depends on at least $\kappa - p'$ bits of $\mathbf{a}$, which are still uniformly random in the view of Bob even when the test succeeds. Therefore, since Bob makes at most q queries, the probability that Bob ever queries some $J_j^{(1-c_j)}$ on any $\mathbf{d}_j^{(1-c_j)}$ is seen to be at most $q 2^{p' - \kappa}$. So, the probability that

Bob passes the test and then queries some $J_j^{(1-c_j)}$ on $\mathbf{d}_j^{(1-c_j)}$ is at most $2^{-p'}q2^{p'-\kappa} = q2^{-\kappa}$. This shows that

$$\text{Success}_V(q) \leq q2^{-\kappa} \ .$$

Combining the above bounds, we get the claimed result.

6 Protecting Bob against a Cheating Alice

It turns out that making the scheme secure against a cheating Bob, we have made the scheme insecure against a cheating Alice. The reason is that now a new piece of information flows from Bob to Alice, namely $h_B = H(E_B)$. When $E_B = E_A$, this clearly does not give Alice new information, but unfortunately Alice can force $E_A \neq E_B$ and use this to get information on the selection bits of Bob.

To see this, assume that instead of sending a correct $f_j = \mathbf{e}_j^{(0)} \oplus \mathbf{e}_j^{(1)}$ Alice sends $f'_j = \mathbf{e}_j^{(0)} \oplus \mathbf{e}_j^{(1)} \oplus \Delta_j$ for some non-zero vector Δ_j. If Bob has selection bit $b_j = 0$, he will use $\mathbf{e}_j = \mathbf{e}_j^{(0)}$. If Bob has selection bit $b_j = 1$, then he will use $\mathbf{e}_j = \mathbf{f}_j \oplus \mathbf{e}_j^{(1)} = \mathbf{e}_j^{(0)} \oplus \Delta_j$. In other symbols, he will use $\mathbf{e}_j = \mathbf{e}_j^{(0)} \oplus b_j\Delta_j$. This allows Alice to learn m of Bob's selection bits using 2^m queries. Illustrated for one bit, Alice would pick Δ_1 to be non-zero and then test whether $h_B = H(\mathbf{e}_1^{(0)}\mathbf{e}_2^{(0)} \cdots \mathbf{e}_\ell^{(0)})$ or $h_B = H((\mathbf{e}_1^{(0)} \oplus \Delta_1)\mathbf{e}_2^{(0)} \cdots \mathbf{e}_\ell^{(0)})$. We are going to combat this using again the principle that high advantage equals high uncertainty: If Alice uses m non-zero Δ_j, then she will have m bits of uncertainty about E_B.

We use this as follows: Instead of just having Bob send h_B, we first let Bob send a commitment to h_B to Alice. Then Alice sends h_A to Bob, and only if $h_A = h_B$ will Bob open the commitment to h_B. If $h_A \neq h_B$, then the protocol is terminated.

It can then be seen that Alice will be caught with probability close to $1-2^{-m}$ when trying to learn m selection bits. We can therefore expect a successful Alice to know very few selection bits of Bob. To get rid of these few remaining corrupted OTs we let Bob, uniformly at random, group the ℓ OTs into $B = \ell/S$ groups, each containing S OTs. Each group of S OTs is then combined into one OT using the S combiner from [CK88]. This combiner has the property that if just one of the underlying S OTs is not corrupted, then the combined OT will be secure against Alice. The combined OT is always secure against Bob.

An important property of the S combiner is that running the combined OT only involves running each underlying OT once, plus $4S$ exclusive ors of κ-bit strings. The complexity of the combined OT is therefore essentially $3S$, when measure in the number of queries a party has to make to the hash function. Also, $3S$ hash values are communicated. Since very few of the underlying OTs are corrupted, we can use a fairly small S in most practical settings, giving a highly efficient scheme.

6.1 Concrete Security Analysis

In this section we analysis the concrete security of the proposed scheme against a cheating Alice.

We measure the insecurity by the probability that the protocol succeeds with at least one OT being corrupted when Alice uses q queries. Since Alice knows the correct value of f_j, any strategy of Alice can be expressed by Alice picking error vectors $\Delta_1, \ldots, \Delta_\ell$, sending $f'_j = f_j \oplus \Delta_j$ to Bob and then sending some h_A. We let P be the number of positions j for which Δ_j is non-zero.

Again we model the hash functions as random oracles, and we assume that the security parameter of the commitment scheme is set large enough that we can model it as being implemented by an ideal functionality. As we did previously, we can start the analysis by also assuming that

Alice sends a value E_A and that the protocol stops if $E_A \neq E_B$. This again adds a term $(q+1)2^{-\kappa}$ to the final bound.

We then look at a game where Alice loses if she specifies $E_A \neq E_B$. Since $f'_j = f_j \oplus b_j\Delta_j$, this means that Alice has to guess the selection bit b_j for all non-zero Δ_j not to lose. We want to compute the probability p that Alice does not lose and that in addition at least one combined OT ends up being insecure. We can write $p = \sum_{P=0}^{\ell} p_P p(P)$, where p_P is the probability that Alice uses P values Δ_j which are non-zero and $p(P)$ is the probability that the protocol terminates without Alice being caught and with at least one OT being corrupted given that Alice uses P non-zero values. It follows that an optimal strategy for Alice is obtained by letting $p_P = 1$ for the P where $p(P)$ is maximal, in which case $p = p(P)$. Below, we therefore focus on upper bounding $p(P)$.

For a given value of P, Alice has to guess P uniformly random and independent bits not to be caught in the test, meaning that she will be caught with probability 2^{-P}. This means that we can write

$$p(P) = 2^{-P} q(P) \ , \tag{3}$$

where $q(P)$ is the probability that one of the combined OTs are corrupted given that Alice knows P selection bits.

Recall that Bob chooses, uniformly at random, which OTs are combined. To analyze this, consider then the following game. We throw, uniformly at random, $\ell = BS$ balls into B buckets each of size S, with the only restriction that exactly S balls end up in each bucket. One way to sample the distribution of balls is to iteratively pick a ball not yet put in a bucket and then put it in the lowest indexed bucket not yet holding S balls. Before we place the balls in the bucket, we color most balls green, except for P of them, which we color red. We call a bucket containing only red balls a red bucket. Clearly $q(P)$ is the probability that there is a red bucket in this game. Since the probability that a given bucket is red is the same for all buckets we can use the union bound to get that

$$q(P) \leq Br(P) \ , \tag{4}$$

where $r(P)$ is the probability that the first bucket is red. The probability that the first bucket is red is exactly the probability that the first S balls picked are red. I.e.,

$$r(P) = \prod_{s=0}^{S-1} \frac{P-s}{\ell - s} \leq \left(\frac{P}{\ell}\right)^S \ , \tag{5}$$

as $(P-s)/(\ell-s)$ is the probability that the next ball is red, given that the previous s balls were red, and $(P-s)/(\ell-s) \leq P/\ell$ when $P \leq \ell$.

Combining (3), (4) and (5) yields

$$p(P) \leq 2^{-P} B (P/\ell)^S = 2^{-P} P^S B \ell^{-S} = 2^{-P} P^S B (SB)^{-S} = 2^{-P} P^S S^{-S} B^{1-S} \ . \tag{6}$$

To maximize (6) in P we only have to maximize $2^{-P}P^S$, which is maximal when $P = S/\log(2)$. Plugging this into (6), we get

$$p(P) \leq (e^{-1}/\log(2))^S S^S S^{-S} B^{1-S} < 0.54^S B^{1-S} \ .$$

If $S = 3$, the bound tells us that no strategy of Alice can have probability better than $0.157464B^{-2}$, which means that the security grows quadratic in B. As a concrete example, if the protocol is run to produce $B = 1,000,000$ OTs, then the probability that one of the combined OTs is corrupted is less than 2^{-42}, which should be negligible in most practical applications. With $S = 3$, the amortized prize per OT is $3 \cdot 3 = 9$ queries per party, plus the transmission of 9 hash values. Below follows a tabulation of some other security levels.

S	Price	Security	$B = 10,000$	$B = 100,000$	$B = 1,000,000$	$B = 10,000,000$
2	6	$0.292B^{-1}$	2^{-15}	2^{-18}	2^{-21}	2^{-25}
3	9	$0.158B^{-2}$	2^{-29}	2^{-35}	2^{-42}	2^{-49}
4	12	$0.086B^{-3}$	2^{-43}	2^{-53}	2^{-63}	2^{-73}

This table shows that for applications of OT requiring a large number of OTs, the amortized price per OT can be made reasonably small even with a high level of security.

References

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 479–488, Philadelphia, Pennsylvania, 22–24 May 1996.

[BM89] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *Advances in Cryptology - Crypto '89*, pages 547–559, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science Volume 435.

[CGT95] C. Crépeau, J. van de Graaf, and A. Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *Advances in Cryptology - Crypto '95*, pages 110–123, Berlin, 1995. Springer-Verlag. Lecture Notes in Computer Science Volume 963.

[CK88] Claude Crépeau and Joe Kilian. Weakening security assumptions and oblivious transfer (abstract). In J. Kilian, editor, *Advances in Cryptology - Crypto '88*, pages 2–7, Berlin, 1988. Springer-Verlag. Lecture Notes in Computer Science Volume 2139.

[Cle89] R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In Gilles Brassard, editor, *Advances in Cryptology - Crypto '89*, pages 573–588, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science Volume 435.

[Cré86] Claude Crépeau. A zero-knowledge poker protocol that acheives confidentiality of the player' strategy or how to achieve an electronic poker face. In A. M. Odlyzko, editor, *Advances in Cryptology - Crypto '86*, pages 239–247, Berlin, 1986. Springer-Verlag. Lecture Notes in Computer Science Volume 263.

[Cré89] Claude Crépeau. Verifiable disclosure of secrets and applications. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology - EuroCrypt '89*, pages 150–154, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science Volume 434.

[CS95] Claude Crépeau and Louis Salvail. Oblivious verification of common string. *CWI Quarterly*, (2):97–109, 1995.

[CS06] Claude Crépeau and George Savvides. Optimal reductions between oblivious transfers using interactive hashing. In C.Dwork, editor, *Advances in Cryptology - Crypto 2006*, pages 201–521, Berlin, 2006. Springer-Verlag. Lecture Notes in Computer Science Volume 4117.

[CT05] Cheng-Kang Chu and Wen-Guey Tzeng. Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries. In Serge Vaudenay, editor, *Public Key Cryptography 2005*, pages 172–183, Berlin, 2005. Springer-Verlag. Lecture Notes in Computer Science Volume 3386.

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

[FMR96] Michael J. Fischer, Silvio Micali, and Charles Rackoff. A secure protocol for the oblivious transfer (extended abstract). *J. Cryptology*, 9(3):191–195, 1996.

[FNW96] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Commun. ACM*, 39(5):77–85, 1996.

[Gar04] Juan A. Garay. Efficient and universally composable committed oblivious transfer and applications. In Moni Naor, editor, *The First Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 297–316. Springer-Verlag, 2004.

[GL90] Shafi Goldwasser and Leonid Levin. Fair computation of general functions in presence of immoral majority. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - Crypto '90*, pages 77–93, Berlin, 1990. Springer-Verlag. Lecture Notes in Computer Science Volume 537.

[GM00] Juan A. Garay and Philip D. MacKenzie. Concurrent oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science*, pages 314–324, Redondo Beach, California, 12–14 November 2000. IEEE.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, 25–27 May 1987.

[GV87] Oded Goldreich and R. Vainish. How to solve any protocol problem - an efficiency improvement. In Carl Pomerance, editor, *Advances in Cryptology - Crypto '87*, pages 73–86, Berlin, 1987. Springer-Verlag. Lecture Notes in Computer Science Volume 293.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In D. Boneh, editor, *Advances in Cryptology - Crypto 2003*, Berlin, 2003. Springer-Verlag. Lecture Notes in Computer Science Volume 2729.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, Washington, 15–17 May 1989.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, Illinois, 2–4 May 1988.

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 474–479, Research Triangle Park, North Carolina, 30 October–1 November 1989. IEEE.

[Lip03] Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003*, pages 416–433, Berlin, 2003. Springer. Lecture Notes in Computer Science Volume 2894.

[Lip04] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. Cryptology ePrint Archive, Report 2004/063, 2004.

[Lip07] Helger Lipmaa. New communication-efficient oblivious transfer protocols based on pairings. Cryptology ePrint Archive, Report 2007/133, 2007.

[MZV02] Yi Mu, Junqi Zhang, and Vijay Varadharajan. m out of n oblivious transfer. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 395–405. Springer, 2002.

[NP00] Moni Naor and Benny Pinkas. Distributed oblivious transfer. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, pages 205–219, Berlin, 2000. Springer. Lecture Notes in Computer Science Volume 1976.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[NP05] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *J. Cryptology*, 18(1):1–35, 2005.

[NSS91] Hannu Nurmi, Arto Salomaa, and Lila Santean. Secret ballot elections in computer networks. *Computers & Security*, 10(6):553–560, 1991.

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

[SCP95] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Zero-knowledge arguments and public-key cryptography. *Inf. Comput.*, 121(1):23–40, 1995.

[TT01] Zhi-Jia Tzeng and Wen-Guey Tzeng. Practical and efficient electronic voting schemes. *J. Inf. Sci. Eng.*, 17(6):865–877, 2001.

[Wei83] S. Weisner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.