On the Reversibility of Oblivious Transfer *

Claude Crépeau †   Miklós Sántha ‡

Laboratoire de Recherche en Informatique
Université de Paris-Sud
Bâtiment 490
91405 Orsay FRANCE

Abstract

A $\binom{2}{1}$–OT_2 (one-out-of-two Bit Oblivious Transfer) is a technique by which a party $\mathcal{S}$ owning two secret bits b_0, b_1, can transfer one of them b_c to another party $\mathcal{R}$, who chooses c. This is done in a way that does not release any bias about $b_{\bar{c}}$ to $\mathcal{R}$ nor any bias about c to $\mathcal{S}$. How can one build a ${}_2\mathrm{TO}$–$\binom{2}{1}$ ($\binom{2}{1}$–OT_2 from $\mathcal{R}$ to $\mathcal{S}$) given a $\binom{2}{1}$–OT_2 (from $\mathcal{S}$ to $\mathcal{R}$)? This question is interesting because in many scenarios, one of the two parties will be much more powerful than the other.

In the current paper we answer this question and show a number of related extensions. One interesting extension of this transfer is the $\binom{2}{1}$–OT_2^k (one-out-of-two String O.T.) in which the two secrets q_0, q_1 are elements of $GF^k(2)$ instead of bits. We show that ${}^k_2\mathrm{TO}$–$\binom{2}{1}$ can be obtained at about the same cost as $\binom{2}{1}$–OT_2^k, in terms of number of calls to $\binom{2}{1}$–OT_2.

1 Introduction

A $\binom{2}{1}$–OT_2 (one-out-of-two Bit Oblivious Transfer) is a technique by which a party $\mathcal{S}$ owning two secret bits b_0, b_1, can transfer one of them b_c to another party $\mathcal{R}$, who chooses c. This is done in a way that does not release any bias about $b_{\bar{c}}$ to $\mathcal{R}$ nor any bias about c to $\mathcal{S}$. This primitive was first introduced in [EGL83] with application to contract signing protocols. A natural and interesting extension of this transfer is the $\binom{2}{1}$–OT_2^k (one-out-of-two String Oblivious Transfer, know as ANNBP in [BCR86]) in which the two secrets q_0, q_1 are elements of $GF^k(2)$ instead of bits. One can find in [CS] a reduction of $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2, i.e. an efficient two-party protocol to achieve $\binom{2}{1}$–OT_2^k based on the assumption of the existence of a protocol for $\binom{2}{1}$–OT_2. This reduction uses essentially $9k$ calls to $\binom{2}{1}$–OT_2 to perform one $\binom{2}{1}$–OT_2^k.

Assume now that we are in a scenario where one party is much more powerful in terms of computational power or simply in terms of technology than the other party. In such a setting it is likely that $\binom{2}{1}$–OT_2 can be implemented in one direction but not the other. In particular one can make a computational assumption of the "weaker" party but not of the other. This scenario was also studied by Ostrovsky, Venkatesan and Yung in [OVY91] where they independently give a reduction similar to 2.1. Also if quantum technology is used [Cre90,BC91] it might be the case that one party is limited in the equipment it can carry (especially if one participant sits on a smart card!). Therefore a fundamental question is: **"Can we reverse $\binom{2}{1}$–OT_2?"**.

*This work was performed while the authors were visiting the Universität des Saarlandes, Saarbrücken.
†Supported in part by an NSERC Postdoctorate Scholarship.
‡Supported in part by an Alexander von Humboldt Fellowship.

Let's call ${}_2\mathrm{TO}\text{–}\binom{2}{1}$ and ${}^k_2\mathrm{TO}\text{–}\binom{2}{1}$ the reversed versions of $\binom{2}{1}\text{–}\mathrm{OT}_2$ and $\binom{2}{1}\text{–}\mathrm{OT}_2^k$. As we shall see in section 2 we can achieve ${}_2\mathrm{TO}\text{–}\binom{2}{1}$ from $\binom{2}{1}\text{–}\mathrm{OT}_2$ at the cost of using $\binom{2}{1}\text{–}\mathrm{OT}_2$ many times (not necessarily constant) to perform a single ${}_2\mathrm{TO}\text{–}\binom{2}{1}$. On the other hand, we show in section 3 that if we wish to perform many ${}_2\mathrm{TO}\text{–}\binom{2}{1}$ simultaneously (to perform ${}^k_2\mathrm{TO}\text{–}\binom{2}{1}$ for instance) it is possible to reduce the marginal cost to a constant number of calls to $\binom{2}{1}\text{–}\mathrm{OT}_2$ per ${}_2\mathrm{TO}\text{–}\binom{2}{1}$.

2 Reversing $\binom{2}{1}$–OT$_2$

To start with, consider the following reduction that constitutes our first attempt to build a ${}_2\mathrm{TO}\text{–}\binom{2}{1}$ from $\binom{2}{1}\text{–}\mathrm{OT}_2$.

Reduction 2.1 (${}_2\mathbf{TO}\text{–}\binom{2}{1}(c,(b_0,b_1))$ from $\binom{2}{1}\text{–}\mathbf{OT}_2$)

1: $\mathcal{S}$ finds a random bit-matrix $C = \begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix}$ such that $C_{00} \oplus C_{01} = \bar{c}$ and $C_{10} \oplus C_{11} = c$.

2: $\mathcal{S}$ runs $\binom{2}{1}\text{–}\mathrm{OT}_2((C_{00}, C_{01}), b_0)$ and $\binom{2}{1}\text{–}\mathrm{OT}_2((C_{10}, C_{11}), b_1)$ with $\mathcal{R}$.

3: $\mathcal{R}$ computes $b \leftarrow C_{0b_0} \oplus C_{1b_1}$ and sends b to $\mathcal{S}$.

4: $\mathcal{S}$ computes $out \leftarrow C_{00} \oplus C_{10} \oplus b$ and outputs out.

Theorem 2.1 *If $\mathcal{S}$ and $\mathcal{R}$ follow honestly the reduction 2.1 then $\mathcal{S}$'s output value will be b_c.*

Proof. We make use of the following trivial Lemma:

Lemma 2.2 $\forall b, c_0, c_1\ [c_0 \oplus c_b = b \wedge (c_0 \oplus c_1)]$.

We have the following equalities

$$\begin{aligned} out &= C_{00} \oplus C_{10} \oplus b \\ &= (C_{00} \oplus C_{0b_0}) \oplus (C_{10} \oplus C_{1b_1}) \\ &= (b_0 \wedge (C_{00} \oplus C_{01})) \oplus (b_1 \wedge (C_{10} \oplus C_{11})) \qquad \text{by Lemma 2.2} \\ &= (b_0 \wedge \bar{c}) \oplus (b_1 \wedge c) \\ &= b_c \end{aligned}$$

■

Unfortunately, this reduction does not provide a full solution to our problem because it is clear that $\mathcal{S}$ can "cheat" this reduction in the sense that he can get $b_0 \oplus b_1$ by picking a matrix C such that $C_{00} \oplus C_{01} = C_{10} \oplus C_{11} = 1$. Indeed what the above reduction achieves is not really a ${}_2\mathrm{TO}\text{–}\binom{2}{1}$but something weaker that can be described in terms of a scalar product. Consider the following reduction **ralacs** that returns the scalar product $(c_0, c_1) \cdot (b_0, b_1)$ to $\mathcal{S}$ on respective inputs (c_0, c_1) and (b_0, b_1).

Reduction 2.2 (ralacs$((c_0, c_1), (b_0, b_1))$ from $\binom{2}{1}$–OT$_2$)

1: $\mathcal{S}$ finds a random bit-matrix $C = \begin{pmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{pmatrix}$ such that $C_{00} \oplus C_{01} = c_0$ and $C_{10} \oplus C_{11} = c_1$.

2: $\mathcal{S}$ runs $\binom{2}{1}$–OT$_2((C_{00}, C_{01}), b_0)$ and $\binom{2}{1}$–OT$_2((C_{10}, C_{11}), b_1)$ with $\mathcal{R}$.

3: $\mathcal{R}$ computes $b \leftarrow C_{0b_0} \oplus C_{1b_1}$ and sends b to $\mathcal{S}$.

4: $\mathcal{S}$ computes $out \leftarrow C_{00} \oplus C_{10} \oplus b$ and outputs out.

The proof of the correctness of this reduction can be obtained in a way similar to that of theorem 2.1. Notice that in fact the reduction 2.1 is nothing more than reduction 2.2 performed with arguments $((\bar{c}, c), (b_0, b_1))$. Thus we have

Theorem 2.3 *If $\mathcal{S}$ and $\mathcal{R}$ follow honestly the reduction 2.2 then $\mathcal{S}$'s output value will be $(c_0, c_1) \cdot (b_0, b_1)$.*

But this time, the reduction we get is also "private". The notion of privacy expresses the fact that all the actions that a cheating participant could take are of no advantage over being honest (in the sense that whatever a cheater gets by cheating he could get by behaving honestly using a different input). For a precise definition of this notion we refer the reader to [Cre90,CM91].

Theorem 2.4 *The reduction 2.2 is both $\mathcal{R}$-private and $\mathcal{S}$-private.*

Proof. The $\mathcal{R}$-privacy of this reduction is simple to prove since all the information $\mathcal{R}$ may get (one bit in each line of C) is purely random. The $\mathcal{S}$-privacy is due to the fact that any choice of C defines some legitimate values for c_0 and c_1 that could be used honestly in the reduction (this was not the case with reduction 2.1).
■

For sake of simplicity, we present the following reduction **scalar**, dual to **ralacs**, with exactly the same properties except that $\mathcal{R}$ gets the output from **scalar** instead of $\mathcal{S}$ in **ralacs**.

Reduction 2.3 (scalar$((b_0, b_1), (c_0, c_1))$ from $\binom{2}{1}$–OT$_2$)

1: $\mathcal{S}$ finds a random bit-matrix $B = \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}$ such that $B_{00} \oplus B_{01} = b_0$ and $B_{10} \oplus B_{11} = b_1$.

2: $\mathcal{S}$ runs $\binom{2}{1}$–OT$_2((B_{00}, B_{01}), c_0)$ and $\binom{2}{1}$–OT$_2((B_{10}, B_{11}), c_1)$ with $\mathcal{R}$.

3: $\mathcal{S}$ computes $b \leftarrow B_{00} \oplus B_{10}$ and sends b to $\mathcal{R}$.

4: $\mathcal{R}$ computes $out \leftarrow B_{0c_0} \oplus B_{1c_1} \oplus b$ and outputs out.

We study **scalar** instead of **ralacs** in the rest of the paper in order to be able to define reductions in the "forward" direction, that is with information flowing from $\mathcal{S}$ to $\mathcal{R}$. The reader should keep in mind that the constructions based on **scalar** can be achieved in the "reverse" direction by switching $\mathcal{S}$ and $\mathcal{R}$ and using **ralacs** instead.

The reader may observe that indeed **scalar** is nothing else but a specific implementation of a primitive known as $2BP$ defined in [BCR86]. In a computational model, a similar idea is implicitly used to solve the problem of computing scalar products in full generality in [GV88]. In [BCR86] it is shown that given any primitive that transfers either b_0, b_1 or any one bit of information about b_0, b_1, it is possible to construct a protocol statistically indistinguishable from $\binom{2}{1}$–OT$_2$. Since reduction 2.3 enables an adversary to get either b_0, b_1, $b_0 \oplus b_1$ or no information at all (!!), it is clear that we can apply their solution.

Their solution requires a blow up in the number of times the primitive is used. In fact, in order to get a protocol that will be exponentially close to $\binom{2}{1}$–OT$_2$ (in some parameter s) their approach requires $\Theta(s)$ calls to **scalar**.

Combining the two ideas, the final resulting reduction is

Reduction 2.4 ($\binom{2}{1}$–OT$_2((b_0, b_1), c)$ from scalar)

1: $\mathcal{S}$ chooses random bits $b_0^1, ..., b_0^s$ and $b_1^1, ..., b_1^s$
such that $\bigoplus_{i=1}^{s} b_0^i = b_0$ and $\bigoplus_{i=1}^{s} b_1^i = b_1$.

2: $\mathcal{S}$ chooses $3s$ random bits $\pi_1, ..., \pi_s$, ©$_0^1, ...,$ ©$_0^s$,©$_1^1, ...,$ ©$_1^s$.

3: $\mathbf{DO}_{i=1}^{s}$ IF $\pi_i = 0$ THEN
execute $a_0^i \leftarrow$**scalar**$((b_0^i,$ ©$_1^i), (\bar{c}, c))$ and $a_1^i \leftarrow$**scalar**$(($©$_0^i, b_1^i), (\bar{c}, c))$
ELSE
execute $a_0^i \leftarrow$**scalar**$(($©$_0^i, b_1^i), (\bar{c}, c))$ and $a_1^i \leftarrow$**scalar**$((b_0^i,$ ©$_1^i), (\bar{c}, c))$.

4: $\mathcal{S}$ reveals $\pi_1, \pi_2, ..., \pi_s$ to $\mathcal{R}$.

5: $\mathcal{R}$ computes $out \leftarrow \bigoplus_{i=1}^{s} a_{c\oplus\pi_i}^i$ and outputs out.

Theorem 2.5 *The reduction 2.4 is a correct and statistically private reduction of $\binom{2}{1}$–$OT_2$to* **scalar**.

In other words, if both parties behave honestly then the reduction 2.4 implements a $\binom{2}{1}$–OT$_2$using calls to **scalar**. In all cases, $\mathcal{S}$ will gain no information whatsoever about $\mathcal{R}$'s input, while $\mathcal{R}$ may learn information about both b_0, b_1 but only with probability 2^{-s}. The formal proof of these statements will appear in the final paper.

3 Achieving $\binom{2}{1}$–OT$_2^k$ from scalar

Assume $\mathcal{S}$ and $\mathcal{R}$ have a mean of accomplishing $\binom{2}{1}$–OT$_2$ and that they wish to perform a $\frac{k}{2}$TO–$\binom{2}{1}$ over the two k-bit strings q_0, q_1. For any string x, let x^i denote the i^{th} bit of x. For a set of indices $I = \{i_1, i_2, ..., i_m\}$, we define x^I to be the concatenation $x^{i_1}x^{i_2}...x^{i_m}$, the indices taken in increasing order.

As mentioned earlier, there is a reduction from $\binom{2}{1}$–OT$_2^k$ to $\binom{2}{1}$–OT$_2$, therefore we could apply this reduction to obtain $\binom{2}{1}$–OT$_2^k$ from **scalar**. Unfortunately, this solution will significantly increase the number of call to **scalar** necessary to implement $\binom{2}{1}$–OT$_2^k$. In [CS] an almost optimal reduction from $\binom{2}{1}$–OT$_2^k$ to $\binom{2}{1}$–OT$_2$ requires about $9k$ calls to $\binom{2}{1}$–OT$_2$. If we combine this with our reduction 2.4 we get a total expansion factor in $\Theta(k(s + \log k))$ calls to **scalar** in order to achieve a protocol exponentially close (in s) to $\binom{2}{1}$–OT$_2^k$. The purpose of this section is to design a better reduction that achieves $\binom{2}{1}$–OT$_2^k$ with only $O(s + k)$ calls to **scalar**.

3.1 Main Tool

Consider a function $f : GF^n(2) \rightarrow GF^k(2)$ with the nice property that for every input string x and every I such that $\#I < d$, seeing the bits x^I releases no information about $f(x)$. Let us be more precise about this.

Definition 3.1 A subset $I \subseteq \{1, 2, ..., n\}$ *biases* a function $f : GF^n(2) \rightarrow GF^k(2)$ if

$$\exists q_0, q_1, x \ \left[\#\{z|z^I = x^I,\ f(z) = q_0\} \neq \#\{z|z^I = x^I,\ f(z) = q_1\}\right]$$

such an x^I is said to *release information* about $f(x)$.

Definition 3.2 A (n, k, d)*-function* is a function $f : GF^n(2) \rightarrow GF^k(2)$ such that

$$\forall I \subseteq \{1, 2, ..., n\}, \#I < d \ [I \text{ does not bias } f].$$

We seek (n, k, d)*-functions* that are easily computable and for which random inverses can be easily computed. If we choose f to be a linear function $f(x) = Mx$ we get that f is an $(n, k, d) - function$ if and only if M is the generator matrix of an (n, k, d) binary linear code. The proof of this fact can be found in [BBR88,CGH*85].

Our main idea is the following. To transfer one of q_0, q_1, $\mathcal{S}$ picks at random two bit strings x_0, x_1 such that $q_0 = f(x_0)$ and $q_1 = f(x_1)$. Then using the protocol **scalar** the bits of x_0, x_1 are transferred to $\mathcal{R}$ in a way that an honest $\mathcal{R}$ will be able to get exactly x_c and compute $q_c = f(x_c)$, while a cheating $\mathcal{R}$ would get less than d bits of at least one of x_0, x_1 and therefore no information about one of q_0 or q_1.

3.2 New Reduction

First we present a new reduction from $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2 that uses the (n, k, d)-functions and from which the reduction from $\binom{2}{1}$–OT_2^k to **scalar** will be deduced. Assume that we define f from a $(n, Rn, \delta n)$ binary linear code, we do the following:

Reduction 3.1 ($\binom{2}{1}$–$\mathbf{OT}_2^{Rn}((q_0, q_1), c)$ from $\binom{2}{1}$–$\mathbf{OT}_2$)

1: $\mathcal{S}$ finds random x_0, x_1 such that $f(x_0) = q_0$ and $f(x_1) = q_1$.

2: $\mathcal{S}$ finds random y_0, y_1 such that $|\ y_0\ |=|\ y_1\ |= en$ and such that $\bigoplus_{m=(i-1)e+1}^{ie} y_0^m = x_0^i$ and $\bigoplus_{m=(i-1)e+1}^{ie} y_1^m = x_1^i$, for $1 \leq i \leq n$.

3: $\mathcal{S}$ finds a random permutation σ of $\{1, 2, ..., en\}$.

4: $\underset{i=1}{\overset{en}{\mathbf{DO}}}$ execute $a^i \leftarrow \binom{2}{1}$–$\mathrm{OT}_2((y_0^{\sigma(i)}, y_1^{\sigma(i)}), c)$.

5: $\mathcal{S}$ reveals σ to $\mathcal{R}$.

6: $\underset{i=1}{\overset{n}{\mathbf{DO}}}$ $\mathcal{R}$ computes $z^i \leftarrow \bigoplus_{m=(i-1)e+1}^{ie} a^{\sigma^{-1}(m)}$.

7: $\mathcal{R}$ computes $out \leftarrow f(z)$ and outputs out.

Theorem 3.3 *If $\mathcal{S}$ and $\mathcal{R}$ follow honestly the reduction 3.1 then $\mathcal{R}$'s output value will be q_c.*

Proof. By definition of the y_i^j's, it is clear that the value of z computed at step 6 is indeed x_c. It follows from the definition of f that the output is therefore correct.
■

On the other hand, the only significant way $\mathcal{R}$ could cheat this reduction is by using values of c that are not the same all the time at step 4. Name for $1 \leq i \leq en$, c^i the value of c used by $\mathcal{R}$ at step 4. Name c^* the less frequent value among the c^i's. Let

$$\xi_0^i = \begin{cases} 1 & \text{if } \mathcal{R} \text{ got } y_0^{(i-1)e+1}, ..., y_0^{ie} \\ 0 & \text{otherwise} \end{cases}$$

be the indicator random variable of x_0^i (with value 1 if and only if $\mathcal{R}$ can compute x_0^i) and define similarly ξ_1^i. We claim that the expected number of $x_{c^*}^i$ that $\mathcal{R}$ can compute will not exceed δn with very high probability (if e is big enough). This implies that he cannot get any information about q_{c^*}, because f is a $(n, Rn, \delta n)$-function.

Claim 3.4 *If for $\epsilon > 0$ we have $e > -(1+\epsilon)\log\delta$ then*

$$\text{Prob}\left(\sum_{i=1}^{n} \xi_{c^*}^i > \delta n\right) < \alpha^n$$

for some constant $0 < \alpha < 1$.

The proof of this claim can be obtain by extension of Chernoff's bound [Chv84].

What we seek now is to minimize the number of $\binom{2}{1}$–OT_2 used. If we start with string of length Rn, this reduction uses en calls to $\binom{2}{1}$–OT_2, thus the expansion factor to be minimized is e/R under the conditions that

- e is a positive integer,
- $e > -(1+\epsilon)\log\delta$,
- we most be able to obtain a $(n, Rn, \delta n)$ binary linear code.

It is known that a random $Rn \times (1+\epsilon)n$ binary matrix generates a $((1+\epsilon)n, Rn, \delta n)$ binary linear code such that $R \approx 1 - H(\delta)$ with probability essentially 1, where $H(x)$ is the entropy function $H(x) = x\log x + (1-x)\log(1-x)$ (consult [MS77]).

If we put all these facts together we get that the optimal value occurs around $e = 4$. This leads us to values of $\delta \approx 0.06$ and $R \approx \frac{2}{3}$. Thus a total expansion factor of about $e/R \approx 6$. This is not bad at all considering that the best know reduction from $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2 gives an expansion factor of about 5 (consult [CS]). Also, any family of codes with better parameter than those given by the Varshamov-Gilbert curve [MS77] will reduce the ratio e/R even more.

3.3 Switching to scalar

What we described in subsection 3.2 is a technique to perform $\binom{2}{1}$–OT_2^k from $\binom{2}{1}$–OT_2. The reason for this is that we can easily extend the above reduction to a reduction for $\binom{2}{1}$–OT_2^k based on **scalar** with an expansion factor twice bigger. The idea is simply to combine the reductions 3.1 and 2.4.

Reduction 3.2 ($\binom{2}{1}$–$\mathbf{OT}_2^{Rn}((q_0, q_1), c)$ from scalar)

1: $\mathcal{S}$ finds random x_0, x_1 such that $f(x_0) = q_0$ and $f(x_1) = q_1$.

2: $\mathcal{S}$ finds random y_0, y_1 such that $\mid y_0 \mid = \mid y_1 \mid = en$ and such that $\bigoplus_{m=(i-1)e+1}^{ie} y_0^m = x_0^i$ and $\bigoplus_{m=(i-1)e+1}^{ie} y_1^m = x_1^i$, for $1 \leq i \leq n$.

3: $\mathcal{S}$ finds a random permutation σ of $\{1, 2, ..., en\}$.

4: $\mathcal{S}$ chooses $3en$ random bits $\pi_1, ..., \pi_{en}, \copyright_0^1, ..., \copyright_0^{en}, \copyright_1^1, ..., \copyright_1^{en}$.

5: $\underset{i=1}{\overset{en}{\mathbf{DO}}}$ IF $\pi_i = 0$ THEN
run $a_0^i \leftarrow$**scalar**$((y_0^{\sigma(i)}, \copyright_1^i), (\bar{c}, c))$ and $a_1^i \leftarrow$**scalar**$((\copyright_0^i, y_1^{\sigma(i)}), (\bar{c}, c))$
ELSE
run $a_0^i \leftarrow$**scalar**$((\copyright_0^i, y_1^{\sigma(i)}), (\bar{c}, c))$ and $a_1^i \leftarrow$**scalar**$((y_0^{\sigma(i)}, \copyright_1^i), (\bar{c}, c))$.

6: $\mathcal{S}$ reveals σ and $\pi_1, \pi_2, ..., \pi_s$ to $\mathcal{R}$.

7: $\underset{i=1}{\overset{n}{\mathbf{DO}}}$ $\mathcal{R}$ computes $z^i \leftarrow \bigoplus_{m=(i-1)e+1}^{ie} a_{c \oplus \pi_{\sigma^{-1}(m)}}^{\sigma^{-1}(m)}$.

8: $\mathcal{R}$ computes $out \leftarrow f(z)$ and outputs out.

Without entering into too many details, this reduction is clearly correct by construction. Again in this case the only significant way $\mathcal{R}$ could "cheat" is by using different values of c at step 5. Replacing $\binom{2}{1}$–OT_2 by **scalar** *does not change* the expected number of x_0^i's and x_1^i's received by $\mathcal{R}$. The analysis is therefore the same as for reduction 3.1. The cost of reduction 3.2 in terms of the number of calls to **scalar** is twice the cost of reduction 3.1 in terms of calls to $\binom{2}{1}$–OT_2. This leaves us with a total expansion factor of about 12 for reduction 3.2.

3.4 $\frac{k}{2}$TO–$\binom{2}{1}$ from $\binom{2}{1}$–OT$_2$

Let's not forget our final goal which was to accomplish a $\frac{k}{2}$TO–$\binom{2}{1}$ with a minimum number of $\binom{2}{1}$–OT_2. The reduction 3.2, when the roles of $\mathcal{S}$ and $\mathcal{R}$ are reversed and **ralacs** is used instead of **scalar**, leads us to a solution for $\frac{k}{2}$TO–$\binom{2}{1}$ using roughly $12k$ calls to **ralacs**. Combining with reduction 2.2 we get a correct and statistically private reduction of $\frac{k}{2}$TO–$\binom{2}{1}$ to $\binom{2}{1}$–OT_2 using roughly $24k$ such calls. This compares reasonably with the result in the forward direction using $9k$ calls [CS].

If one wishes to accomplish an approximation to $\frac{k}{2}$TO–$\binom{2}{1}$ where all the probabilities involved may differ of at most 2^{-s}, then running reduction 3.2 to accomplish $\frac{l}{2}$TO–$\binom{2}{1}$, for $l \in O(s+k)$ will reduce all the probabilities below that limit. The initial strings q_0, q_1 of length k can be arbitrarily padded to length l. The total cost in terms in $\binom{2}{1}$–OT_2 will be in $O(s+k)$.

Acknowledgements

We would like to thank Gilles Brassard and Johannes Buchmann, for their help, comments, and support.

References

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

[BC91] G. Brassard and C. Crépeau. Quantum bit commitment and coin tossing protocols. In S. Vanstone, editor, *Advances in Cryptology: Proceedings of Crypto '90*, Springer-Verlag, 1991. to appear.

[BCR86] G. Brassard, C. Crépeau, and J.–M. Robert. Information theoretic reductions among disclosure problems. In 27^{th} *Symp. of Found. of Computer Sci.*, pages 168–173, IEEE, 1986.

[CGH*85] B. Chor, O. Goldreich, J. Hastad, J. Friedmann, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. In *Proceedings of the* 26*th IEEE Symposium on Foundations of Computer Science*, pages 396–407, IEEE, Portland, 1985.

[Chv84] V. Chvatal. Probabilistic methods in graph theory. *Annals of Operations Research*, 1:171–182, 1984.

[CM91] C. Crépeau and S. Micali. Secure two-party protocols. 1991. in preparation.

[Cre90] C. Crépeau. *Correct and Private Reductions among Oblivious Transfers*. PhD thesis, Department of Elec. Eng. and Computer Science, Massachusetts Institute of Technology, 1990. Supervised by Silvio Micali.

[CS] C. Crépeau and M. Sántha. Efficient reductions among oblivious transfer protocols. submitted to STOC 91.

[EGL83] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proceedings CRYPTO 82*, pages 205–210, Plenum Press, New York, 1983.

[GV88] O. Goldreich and R. Vainish. How to solve any protocol problem-an efficiency improvement (extended abstract). In C. Pomerance, editor, *Advances in Cryptology: Proceedings of Crypto '87*, pages 73–86, Springer-Verlag, 1988.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[OVY91] R. Ostrovsky, R. Venkatesan, and M. Yung. On the complexity of asymmetric games. In *Proceedings of Sequences '91*, 1991. to appear. This work was first presented at the DIMACS workshop on cryptography, October 1990.