Oblivious Transfers and Privacy Amplification

12 February 1996

Abstract

Assume $\mathcal{A}$ owns two secret k–bit strings. She is willing to disclose one of them to $\mathcal{B}$, at his choosing, provided he does not learn anything about the other string. Conversely, $\mathcal{B}$ does not want $\mathcal{A}$ to learn which secret he chose to learn. A protocol for the above task is said to implement One-out-of-two String Oblivious Transfer, denoted $\binom{2}{1}$–OT_2^k. This primitive is particularly useful in a variety of cryptographic settings. An apparently simpler task corresponds to the case $k = 1$ of two one-bit secrets: this is known as One-out-of-two Bit Oblivious Transfer, denoted $\binom{2}{1}$–OT_2. We address the question of reducing $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2.

This question is not new: it was introduced in 1986. However, most solutions until now have implicitly or explicitly depended on the notion of *self-intersecting codes*. It can be proved that this restriction makes it asymptotically impossible to implement $\binom{2}{1}$–OT_2^k with fewer than about $3.5277\,k$ instances of $\binom{2}{1}$–OT_2. The current paper introduces the idea of using *privacy amplification* as underlying technique to reduce $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2. This allows for more efficient solutions at the cost of an exponentially small probability of failure: it is sufficient to use slightly more than $2k$ instances of $\binom{2}{1}$–OT_2 in order to implement $\binom{2}{1}$–OT_2^k. Moreover, we show that privacy amplification allows for the efficient implementation of $\binom{2}{1}$–OT_2^k from generalized versions of $\binom{2}{1}$–OT_2 that would not have been suitable for the earlier techniques based on self-intersecting codes. An application of this more general reduction is given.

Key Words: Information-Theoretic Security, Oblivious Transfer, Privacy Amplification, Reduction Between Protocols.

1 Introduction

The equivalence between cryptographic primitives is a major research topic [BCR86, Cré88, CK88, Kil88, Cré89, GL89, Cré90, Kil91, CS91a, CS91b, CGT95, CS95, BCS96]. A large number of cryptographic protocols have been shown equivalent to one another.

One-out-of-two String Oblivious Transfer, denoted $\binom{2}{1}$–OT_2^k, is a primitive that originates with [Wie70] (under the name of "multiplexing"), a paper that marked the birth of quantum cryptography. According to this primitive, one party $\mathcal{A}$ owns two secret k–bit strings w_0 and w_1, and another party $\mathcal{B}$ wants to learn w_c for a secret bit c of his choice. $\mathcal{A}$ is willing to collaborate provided that $\mathcal{B}$ does not learn any information about $w_{\bar{c}}$, but $\mathcal{B}$ will not participate if $\mathcal{A}$ can obtain information about c. Independently from [Wie70] but inspired by [Rab81], a natural restriction of this primitive was introduced subsequently in [EGL83] with applications to contract signing protocols: One-out-of-two Bit Oblivious Transfer, denoted $\binom{2}{1}$–OT_2, concerns the case $k = 1$ in which w_0 and w_1 are single-bit secrets, generally called b_0 and b_1 in that case.

Techniques were introduced in [BCR86] and refined in [CS91b, BCS96] to reduce $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–OT_2: several two-party protocols were given to achieve One-out-of-two String Oblivious Transfer based on the assumption of the availability of a protocol for the simpler One-out-of-two Bit Oblivious Transfer. The fact that $\binom{2}{1}$–OT_2^k can be reduced to $\binom{2}{1}$–OT_2 is not surprising because a number of authors [GMW87, GV87, Kil88, Cré89, GL89, CGT95] have shown that $\binom{2}{1}$–OT_2 is sufficient to implement *any* two-party computation. Our interest in direct reductions is their far greater efficiency. With the exception of [CS91a], all previous direct reductions that we are aware of [BCR86, CS91b, BCS96] are based on a notion called *zigzag functions*, whose construction is reduced to finding particular types of error-correcting codes called *self-intersecting codes*. In a nutshell, this approach consists in selecting once and for all a suitable *zigzag* function f from $\{0,1\}^n$ to $\{0,1\}^k$ for $n > k$ as small as possible, so that if x_0 is a random preimage of w_0 and x_1 is a random preimage of w_1, and if $\mathcal{B}$ is given to choose via $\binom{2}{1}$–OT_2 to see the i^{th} bit of either x_0 or x_1, $1 \le i \le n$, then no information can be inferred on at least one of w_0 or w_1. This approach has led to various reductions with expansion factors β ranging from 4.8188 to 18, that is various polynomial-time constructible methods using $n = \beta k$ instances of $\binom{2}{1}$–OT_2 to perform one $\binom{2}{1}$–OT_2^k on k–bit strings. Komlós proved that this approach cannot yield an expansion factor β that is asymptotically better than 3.5277 [CL85].

The current paper exploits a new approach to this problem using *privacy amplification*, a notion first introduced in the context of key exchange protocols [BBR88]. The new approach allows for a solution requiring only slightly more than $2k$ instances of $\binom{2}{1}$–OT_2 to perform one $\binom{2}{1}$–OT_2^k, and it can be extended to a whole range of generalizations of $\binom{2}{1}$–OT_2 that could not be used with the reductions based on self-intersecting codes.

An application of the simplest of our generalizations is also considered: $\binom{2}{1}$–OT_2^k from $\mathcal{A}$ to $\mathcal{B}$ can be reduced to $\binom{2}{1}$–OT_2 in the other direction, ie from $\mathcal{B}$ to $\mathcal{A}$, by only doubling the cost of reducing to $\binom{2}{1}$–OT_2 from $\mathcal{A}$ to $\mathcal{B}$. This improves on an earlier result of [CS91a].

2 Privacy Amplification Method

Assume $\mathcal{A}$ knows a random n–bit string x about which $\mathcal{B}$ has partial information. *Privacy amplification* is a technique invented in [BBR88] and refined in [BBCM95] that allows $\mathcal{A}$ to shrink x to a shorter string w about which $\mathcal{B}$ has an arbitrarily small amount of information even if he knows the recipe used by $\mathcal{A}$ to transform x into w. Intuitively, this can be used to implement $\binom{2}{1}$–$\mathrm{OT}_2^k(w_0, w_1)(c)$ from $\binom{2}{1}$–OT_2 because $\mathcal{A}$ can offer $\mathcal{B}$ to read one of two random strings x_0 or x_1 by a simple sequence of $\binom{2}{1}$–$\mathrm{OT}_2(x_0^i, x_1^i)(c_i)$. Subsequently, $\mathcal{A}$ tells $\mathcal{B}$ how to transform x_0 into w_0 and x_1 into w_1 by way of privacy amplification. An honest $\mathcal{B}$ who accessed all the bits of x_c can reconstruct w_c from this information. But a dishonest $\tilde{\mathcal{B}}$ who tried to access some of the bits of x_0 and some of the bits of x_1 will not have enough information on at least one of them to infer any information on the corresponding w_i or even joint information on both w_0 and w_1.

An important fact about the method based on zigzag functions considered in earlier papers is that there is no way for $\mathcal{B}$ to learn information about both w_0 and w_1 even though the zigzag function is known before he gets to choose which bits of x_0 and x_1 to obtain through the $\binom{2}{1}$–OT_2 instances. In the new approach based on privacy amplification, $\mathcal{A}$ reveals the function to $\mathcal{B}$ *after* the necessary $\binom{2}{1}$–OT_2's have been performed. This allows for a protocol that is simpler, more general and more efficient, but at the cost of a vanishingly small probability of failure. A drawback of this approach is that a new function must be generated and transmitted at each run of the protocol.

The following table compares the efficiency of the earlier methods to that of privacy amplification. The column "expansion factor" gives a number β so that a $\binom{2}{1}$–OT_2^k can be achieved with βk instances of $\binom{2}{1}$–OT_2. Thus we see that the privacy amplification method is preferable provided a probability of failure can be tolerated.

Method	expansion factor	failure probability	construction time
Monte Carlo Zigzag[1]	$4.8188 + \varepsilon$	2^{-s}	$O(k^2)$
Las Vegas Zigzag[2]	$9.6377 + \varepsilon$	0	$O(k^2)$
Zigzag à la Justesen[3]	18	0	$O(k^4)$
Zigzag à la Goppa[4]	6.4103	0	$O(k^{32})$
Privacy Amplification	$2 + \varepsilon$	2^{-s}	$O(k^2)$

[1] Attributed to Cohen and Lempel in [BCS96].
[2] Attributed to Joe Kilian in [BCS96].
[3] From [BCS96].
[4] From [CZ94] based on a method of [CS91b].

2.1 The New Protocol

Let s be a security parameter chosen by $\mathcal{A}$ and $\mathcal{B}$ so that they agree to tolerate a probability 2^{-s} of failure. Let γ be a constant to be determined later, let $n = \gamma k + s$, and let $\mathbb{F}_2$ denote the field of integers modulo 2.

Privacy amplification is based on the general notion of universal classes of hash functions [CW79]. For sake of simplicity, we use a specific class of hash functions in our protocol to implement $\binom{2}{1}$–OT_2^k from $\binom{2}{1}$–OT_2: $\{h \mid h(x) = Mx, \text{ for } M \text{ a } k \times n \text{ matrix over } \mathbb{F}_2\}$. Other, more efficient classes of hash functions can be used, but it is not known if the definition of universal$_2$ classes is sufficient in general to make our protocol work.

Protocol 2.1 ($\binom{2}{1}$–$\mathbf{OT}_2^k(w_0, w_1)(c)$)

1: $\mathcal{A}$ picks two random n–bit strings x_0 and x_1.

2: $\mathop{\mathbf{DO}}_{i=1}^{n}$ $\mathcal{A}$ transfers $t^i \leftarrow \binom{2}{1}$–$\mathrm{OT}_2(x_0^i, x_1^i)(c)$ to $\mathcal{B}$.

3: $\mathcal{A}$ picks two random $k \times n$ matrices M_0 and M_1 over $\mathbb{F}_2$ and announces them to $\mathcal{B}$.

4: $\mathcal{A}$ sets $y_0 \leftarrow M_0 x_0 \oplus w_0$ and $y_1 \leftarrow M_1 x_1 \oplus w_1$ and announces y_0 and y_1 to $\mathcal{B}$.

5: $\mathcal{B}$ recovers w_c by computing $M_c t \oplus y_c$.

We postpone to section 2.3 the proof that this protocol is private [5] for an appropriate choice of γ because we shall first generalize it to permit at no extra cost the use of another primitive called *XOR Oblivious Transfer*.

2.2 XOR Oblivious Transfer

A $\binom{2}{1}$–XOT_2 is an extension of $\binom{2}{1}$–OT_2 that enables a sender $\mathcal{A}$ to transfer to a receiver $\mathcal{B}$ either one bit among b_0 and b_1 or their exclusive-or, at $\mathcal{B}$'s choice. More formally, $\mathcal{A}$ inputs b_0 and b_1 into the protocol, $\mathcal{B}$ inputs $c \in \{0, 1, \oplus\}$, and $\mathcal{B}$ learns b_c while $\mathcal{A}$ learns nothing, where for convenience we use $b_\oplus$ to denote $b_0 \oplus b_1$. As usual, this is done in an all-or-nothing fashion: $\mathcal{B}$ cannot get more information about b_0 and b_1 than b_0, b_1 or $b_\oplus$, however malicious or (computationally) powerful he is.

The $\binom{2}{1}$–XOT_2 comes naturally in a specific implementation of $\binom{2}{1}$–OT_2: in [BCR86a] a protocol for $\binom{2}{1}$–OT_2 is given under the assumption that deciding quadratic residuosity modulo a composite number is hard. In that implementation, the possibility that $\tilde{\mathcal{B}}$ obtains $b_\oplus$ arises naturally and some effort is made to prevent it. The current paper shows that this effort was unnecessary if the final goal is to implement $\binom{2}{1}$–OT_2^k rather than simply $\binom{2}{1}$–OT_2.

[5] Informally, the protocol is *private* if $\mathcal{B}$ cannot learn information on both w_0 and w_1 except perhaps with negligible probability. In addition, $\mathcal{B}$ must not be able to obtain joint information on w_0 and w_1 except for what follows from his *a priori* knowledge and his learning one of the two strings. We shall later relax the condition to allow $\mathcal{B}$ an exponentially small amount of unauthorized information. Please consult the Appendix for a formal definition of this notion.

We consider XOR oblivious transfers for two additional reasons:

- Protocol 2.1 above works equally well using $\binom{2}{1}$–XOT$_2$ instead of $\binom{2}{1}$–OT$_2$;
- $\binom{2}{1}$–XOT$_2$ can be implemented from only two instances of a reversed $\binom{2}{1}$–OT$_2$ (from $\mathcal{B}$ to $\mathcal{A}$). Therefore, $\binom{2}{1}$–OT$_2^k$ from $\mathcal{A}$ to $\mathcal{B}$ can be implemented from slightly more than $4k$ instances of $\binom{2}{1}$–OT$_2$ from $\mathcal{B}$ to $\mathcal{A}$, a three-fold improvement over [CS91a].

This first result is discussed in section 2.3, while the second is seen in section 2.4.

2.3 Privacy

Consider a variation of Protocol 2.1 in which the transfers at step 2 are performed through $\binom{2}{1}$–XOT$_2$ instead of $\binom{2}{1}$–OT$_2$. Even though this makes no difference if $\mathcal{B}$ follows the protocol honestly, it gives him additional opportunities for cheating if he so desires. Our goal is to show that whatever program $\tilde{\mathcal{B}}$ is ran by $\mathcal{B}$, he is not able to obtain information on both w_0 and w_1, except with a probability that is exponentially small in the security parameter s. From now on, think of x_0 and x_1 as column-vectors of length n, and of $m_0 = M_0 x_0$ and $m_1 = M_1 x_1$ as column-vectors of length k, all over $\mathbb{F}_2$. First, we show that immediately after Step 3 of the protocol, whatever program $\tilde{\mathcal{B}}$ is ran by $\mathcal{B}$, he will have no information about one of m_0 or m_1 and no information allowing him to connect m_0 and m_1 (such as $m_0 \oplus m_1$ for instance), except with exponentially small probability. (Formally, there will be some bit $\tilde{c}$ such that the first three steps of the protocol would give no additional information to $\tilde{\mathcal{B}}$ about the pair (m_0, m_1) than if he were simply told the value of $m_{\tilde{c}}$—see the Appendix.) We conclude the result about w_0 and w_1 at the end of the protocol from the fact that m_0 and m_1 are used as one-time pads to transfer them.

Suppose $\tilde{\mathcal{B}}$ reads the bits $x^i_{c_i}$ with $c_i \in \{0, 1, \oplus\}$ at his choosing. Let g be a non-trivial linear function of m_0 and m_1. In other words, $g(m_0, m_1) = v_0 m_0 \oplus v_1 m_1$ for some line-vectors v_0 and v_1 of length k over $\mathbb{F}_2$ such that both v_0 and v_1 are non-zero [6].

Theorem 2.1 *Consider the knowledge that $\tilde{\mathcal{B}}$ has about m_0 and m_1 immediately after Step 3 of the protocol.*

$$\text{Prob}\left(\exists \textit{ non-trivial } g \textit{ such that } \tilde{\mathcal{B}} \textit{ knows } g(m_0, m_1)\right) < 2^{-s}$$

provided $\gamma \geq 2$.

Proof. We first describe the condition under which $\tilde{\mathcal{B}}$ learns $g(m_0, m_1)$ at Step 3 of the protocol for some specific non-trivial linear function g. By definition

$$g(m_0, m_1) = v_0 m_0 \oplus v_1 m_1 = v_0 M_0 x_0 \oplus v_1 M_1 x_1 = z_0 x_0 \oplus z_1 x_1$$

[6] Note that by virtue of v_0 and m_0 being a line-vector and a column-vector, respectively, a "matrix" multiplication such as $v_0 m_0$ computes the scalar product; similarly, given that x_0 is also a column-vector, an expression such as $v_0 M_0 x_0$ makes sense: it is simply an element of $\mathbb{F}_2$. This notation is handy because $v_0 M_0 x_0$ can be thought of indifferently as either the scalar product of v_0 with $M_0 x_0$ or of $v_0 M_0$ with x_0.

where $z_0 = v_0 M_0$ and $z_1 = v_1 M_1$. Because x_0 and x_1 are random, $\tilde{\mathcal{B}}$ cannot learn anything about $g(m_0, m_1)$ at Step 3 unless if he is lucky enough that his choices c_i simultaneously follow

$$c_i = \begin{cases} 0 & \text{when } (z_0^i, z_1^i) = (1,0) \\ 1 & \text{when } (z_0^i, z_1^i) = (0,1) \\ \oplus & \text{when } (z_0^i, z_1^i) = (1,1) \end{cases}$$

in all the instances of $\binom{2}{1}$–XOT$_2$ such that z_0^i and z_1^i are not both 0. (The value of c_i is unimportant when $(z_0^i, z_1^i) = (0,0)$ since neither x_0^i nor x_1^i is required in that case to learn $g(m_0, m_1)$.)

But remember that M_0 and M_1 are picked at random and neither v_0 nor v_1 is zero. Therefore $z_0 = v_0 M_0$ and $z_1 = v_1 M_1$ are random binary strings of length n chosen independently according to the uniform distribution. In particular, z_0 and z_1 are independent of $\tilde{\mathcal{B}}$'s choices of c_i's. It follows that, for each i, the probability that either $(z_0^i, z_1^i) = (0,0)$ or $\tilde{\mathcal{B}}$ chose c_i appropriately according to the above case analysis is exactly $1/2$. Since $\tilde{\mathcal{B}}$ must be lucky for each i, $1 \le i \le n$,

$$\text{Prob}\left(\tilde{\mathcal{B}} \text{ learns } g(m_0, m_1)\right) = 2^{-n}$$

for each non-trivial linear function g, whatever choices $\tilde{\mathcal{B}}$ makes for the c_i's. Finally, given that there are less than 2^{2k} such linear functions, we conclude that

$$\text{Prob}\left(\exists \text{ non-trivial } g \text{ such that } \tilde{\mathcal{B}} \text{ learns } g(m_0, m_1)\right) < 2^{2k-n} = 2^{(2-\gamma)k-s} \le 2^{-s}$$

provided $\gamma \ge 2$. ■

Theorem 2.2 *Protocol 2.1 is private even if the transfers at step 2 are performed through* $\binom{2}{1}$*–XOT*$_2$ *instead of* $\binom{2}{1}$*–OT*$_2$.

Proof. We know from Theorem 2.1 that, except with probability at most 2^{-s}, $\mathcal{B}$ has not learned $g(m_0, m_1)$ by the end of Step 3 for any linear function g that involves both m_0 and m_1. On the other hand, it is possible for $\mathcal{B}$ to learn linear functions of m_0 or m_1 alone (by setting all the $c_i = 0$ or $c_i = 1$ as in the honest protocol). It follows that there is a $d \in \{0,1\}$ such that $\mathcal{B}$ learns no non-trivial linear function of m_d because if he could learn non-trivial linear functions $g_0(m_0)$ and $g_1(m_1)$, he would have learned $g_0(m_0) \oplus g_1(m_1)$, a non-trivial linear function of both m_0 and m_1. We can say something stronger: not only does $\mathcal{B}$ learn no non-trivial linear function of m_d, but he learns no information of any kind that involves m_d. This is true because m_0 and m_1 are purely random and the only source of information that $\mathcal{B}$ has about them (up until Step 3) is given by linear functions of m_0 and m_1. Since m_d is used by $\mathcal{A}$ at Step 4 as one-time pad to transmit w_d to $\mathcal{B}$, it follows that $\mathcal{B}$ learns no information of any kind that involves w_d. ■

2.4 Reversing Oblivious Transfer

Consider that $\mathcal{A}$ wants to send one of two words w_0 or w_1 to $\mathcal{B}$ when they only have an $\binom{2}{1}$–OT$_2$ channel running from $\mathcal{B}$ to $\mathcal{A}$. A very efficient protocol for sending one of two *bits* from $\mathcal{A}$ to $\mathcal{B}$ is given in [CS91a] provided $\mathcal{A}$ does not mind the possibility that $\mathcal{B}$ might learn the exclusive-or of her two bits. In other words it is much easier to implement $\binom{2}{1}$–XOT_2 than $\binom{2}{1}$–OT_2 from $\mathcal{A}$ to $\mathcal{B}$ given an $\binom{2}{1}$–OT_2 channel from $\mathcal{B}$ to $\mathcal{A}$. This is fine because we just showed that $\binom{2}{1}$–XOT_2 is just as good as $\binom{2}{1}$–OT_2 for the purpose of implementing $\binom{2}{1}$–OT_2^k. For completeness, here is the protocol from [CS91a] to implement $\binom{2}{1}$–XOT_2 from an $\binom{2}{1}$–OT_2 channel running in the reverse direction.

Protocol 2.2 ($\binom{2}{1}$–$\mathbf{XOT}_2(b_0, b_1)(c)$)

1: **if** $c = \oplus$ then $\mathcal{B}$ sets $c_0 \leftarrow c_1 \leftarrow 1$ **else** $\mathcal{B}$ sets $c_0 \leftarrow \bar{c}$ and $c_1 \leftarrow c$.
$\mathcal{B}$ picks two random bits x_0^0, x_0^1 and sets $x_1^0 \leftarrow x_0^0 \oplus c_0$ and $x_1^1 \leftarrow x_0^1 \oplus c_1$.

2: $\mathcal{B}$ transfers $t_0 \leftarrow \binom{2}{1}\text{–}\mathrm{OT}_2(x_0^0, x_1^0)(b_0)$ and $t_1 \leftarrow \binom{2}{1}\text{–}\mathrm{OT}_2(x_0^1, x_1^1)(b_1)$ to $\mathcal{A}$.

3: $\mathcal{A}$ announces $u \leftarrow t_0 \oplus t_1$ to $\mathcal{B}$.

4: $\mathcal{B}$ recovers b_c by computing $u \oplus x_0^0 \oplus x_0^1$.

Theorem 2.3 *Protocol 2.2 is correct.*

Proof. The output calculated by $\mathcal{B}$ is

$$\begin{aligned} u \oplus x_0^0 \oplus x_0^1 &= x_{b_0}^0 \oplus x_{b_1}^1 \oplus x_0^0 \oplus x_0^1 \\ &= (x_{b_0}^0 \oplus x_0^0) \oplus (x_{b_1}^1 \oplus x_0^1) \\ &= (b_0 \wedge c_0) \oplus (b_1 \wedge c_1) \\ &\quad \text{since } x_i^j \oplus x_0^j = i \wedge c_j \text{ for } i, j \in \{0, 1\} \\ &= b_c. \end{aligned}$$

■

The fact that this protocol cannot be used by $\mathcal{A}$ to find out about c is guaranteed by the fact that $\binom{2}{1}$–OT_2 discloses only one of x_0^j or x_1^j. Similarly, the fact that $\mathcal{B}$ cannot use the protocol to find out more than b_0, b_1 or $b_\oplus$ is guaranteed by the fact that

- if $x_0^0 = x_1^0$ and $x_0^1 = x_1^1$ then $\mathcal{B}$ finds out nothing;
- if $x_0^0 \neq x_1^0$ and $x_0^1 = x_1^1$ then $\mathcal{B}$ finds out exactly b_0;
- if $x_0^0 = x_1^0$ and $x_0^1 \neq x_1^1$ then $\mathcal{B}$ finds out exactly b_1;
- if $x_0^0 \neq x_1^0$ and $x_0^1 \neq x_1^1$ then $\mathcal{B}$ finds out exactly $b_\oplus$.

2.5 Generalized Oblivious Transfer

A $\binom{2}{1}$–GOT_2 is a cryptographic protocol for two participants that enables a sender $\mathcal{A}$ to transfer a one-bit function evaluated on (b_0, b_1) to a receiver $\mathcal{B}$ who chooses secretly which one-bit function (f) he gets from her input bits. This is done in an all-or-nothing fashion: $\mathcal{B}$ cannot get more information about b_0 and b_1 than $f(b_0, b_1)$ for some f, however malicious or (computationally) powerful he is, and $\mathcal{A}$ finds out nothing about the choice f of $\mathcal{B}$.

The following table enumerates all 14 possible non-constant functions from two bits to one. (We ignore the two constant function since they would yield no information if used.) The symbols used refer to the common boolean functions. Example: $\overline{\wedge}$ stands for $\overline{b_0 \wedge b_1}$. Only the notations 0 and 1, which are used for the projection functions $b_0 0 b_1 = b_0$ and $b_0 1 b_1 = b_1$, are not standard. The ordinary $\binom{2}{1}$–OT_2 is a special case of $\binom{2}{1}$–GOT_2 where $\mathcal{B}$ is limited to the functions 0 and 1.

b_0	b_1	∇	$\not\Leftarrow$	$\overline{1}$	$\not\Rightarrow$	$\overline{0}$	$\oplus$	$\overline{\wedge}$	$\wedge$	$\overline{\oplus}$	0	$\rightarrow$	1	$\leftarrow$	$\vee$
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
biased		$\surd$	$\surd$		$\surd$			$\surd$	$\surd$			$\surd$		$\surd$	$\surd$

Figure 1: the 14 non-constant functions from two bits to one.

It has been shown in [BCR86] that $\binom{2}{1}$–GOT_2 is a sufficient primitive to implement $\binom{2}{1}$–OT_2. The reduction they presented uses $\Theta(s)$ runs of $\binom{2}{1}$–GOT_2 to achieve a single $\binom{2}{1}$–OT_2 in such a way that the reduction may fail and give both bits to $\mathcal{B}$ with probability 2^{-s}. If this protocol is combined with a standard reduction of $\binom{2}{1}$–OT_2^k we obtain a global cost of $\Theta(ks)$ runs of $\binom{2}{1}$–GOT_2 per $\binom{2}{1}$–OT_2^k. Contrary to reductions to $\binom{2}{1}$–OT_2, reductions to $\binom{2}{1}$–GOT_2 *must* absolutely involve an error probability since it is *always* possible to get all the information sent by $\mathcal{A}$ by selecting the appropriate biased function at each transfer. Using the new privacy amplification method we obtain a reduction of $\binom{2}{1}$–OT_2^k to $\binom{2}{1}$–GOT_2 at a cost of only $\Theta(k + s)$.

Consider a variation of protocol 2.1 in which the transfers of step 2 are performed through $\binom{2}{1}$–GOT_2 instead of $\binom{2}{1}$–OT_2. Our goal is to show that whatever program $\tilde{\mathcal{B}}$ is ran by $\mathcal{B}$, he is not able to obtain non-negligible information on both w_0 and w_1, except with a probability that is exponentially small in the security parameter s. For this we need to make n larger than $\gamma k + s$, but it will remain in $\Theta(k + s)$—see the proof of Theorem 2.4 for details. First we show that immediately after Step 3 of the protocol, whatever program $\tilde{\mathcal{B}}$ is ran by $\mathcal{B}$, he will have negligible information about one of m_0 or m_1 and negligible information allowing him to connect m_0 and m_1. We conclude a similar result about w_0 and w_1 at the end of the protocol from the fact that m_0 and m_1 are used as one-time pads to transfer them.

Suppose $\tilde{\mathcal{B}}$ reads the bits $x_{c_i}^i$ with $c_i \in \{\nabla, \not\Leftarrow, \overline{1}, \not\Rightarrow, \overline{0}, \oplus, \overline{\wedge}, \wedge, \overline{\oplus}, 0, \rightarrow, 1, \leftarrow, \vee\}$ at his choosing. As before, let g be a non-trivial linear function of m_0 and m_1, that is $g(m_0, m_1) = v_0 m_0 \oplus v_1 m_1$ for some non-zero binary line-vectors v_0 and v_1 of length k. We say that $\mathcal{B}$ can α*–bias* a bit if he can guess it with probability better than $\frac{1}{2} + \alpha$ of being correct.

Theorem 2.4 *Consider the knowledge that $\tilde{\mathcal{B}}$ has about m_0 and m_1 immediately after Step 3 of the protocol.*

$$\text{Prob}\left(\exists \textit{ non-trivial } g \textit{ such that } \tilde{\mathcal{B}} \textit{ can } 2^{-k-1-s/2}\textit{–bias } g(m_0, m_1)\right) < 2^{-s}.$$

Proof. Let γ and a be constants to be determined later and let $n = (a+1)(\gamma k + s)$. Let $Biased = \{i \,|\, c_i \in \{\overline{\vee}, \leftleftarrows, \rightrightarrows, \overline{\wedge}, \wedge, \rightarrow, \leftarrow, \vee\}\}$, the set of positions where $\mathcal{B}$ uses a biased function. If $\#Biased < a(\gamma k + s)$ then Theorem 2.1 applies with $\gamma \geq 2$ and $n = \gamma k + s$. We thus get the desired result. Otherwise $\#Biased \geq a(\gamma k + s)$ is the more interesting case to consider. Consider the set of positions where $\tilde{\mathcal{B}}$ has used a biased function. As before, $\tilde{\mathcal{B}}$ would have learned $g(m_0, m_1)$ exactly if he had simultaneously obtained

$$\begin{array}{rl} x_0^i & \text{when } (z_0^i, z_1^i) = (1,0) \\ x_1^i & \text{when } (z_0^i, z_1^i) = (0,1) \\ x_\oplus^i & \text{when } (z_0^i, z_1^i) = (1,1) \end{array}$$

for all i for which z_0^i and z_1^i are not both 0.

Remember that M_0, M_1, x_0 and x_1 are picked at random. Thus z_0 and z_1 are random binary words of length n. Since $\mathcal{B}$ has used a biased function in position i, with probability 1/4 he will have learned both x_0^i and x_1^i, and with probability 3/4 he will be able to 1/6–bias x_0^i, x_1^i and $x_\oplus^i$. (This is because each biased function has one output that uniquely defines a specific pair of inputs, while the other output leaves three pairs of inputs equally likely.) This means that in each such position i, $\tilde{\mathcal{B}}$ has obtained the bit he needs with probability 7/16 and with probability 9/16 he can only 1/6–bias the bit he needs. Of the $a(\gamma k + s)$ such values of i, less than $a(\gamma k + s)/4$ of them will fall in the second case with probability at most $2 \times e^{-25a(\gamma k+s)/1024} \approx 2^{-(\gamma k+s)}$ according to Berstein's law of large numbers, for $a \approx 28$. When $7(\gamma k + s)$ of the bits involved in the calculation of $g(m_0, m_1)$ are 1/6–biased, even if all the other bits are exactly known, $\mathcal{B}$'s can only $(1/3)^{7(\gamma k+s)}/2$–bias the value of $g(m_0, m_1)$. (In general, if one can δ–bias $x_1, x_2, ..., x_l$, she can $(2\delta)^l/2$–bias $x_1 \oplus x_2 \oplus ... \oplus x_l$ [Cré90].) It follows that for any set of choices $\{c_i\}$, and any $v_0, v_1 \neq 0^k$

$$\text{Prob}\left(\tilde{\mathcal{B}} \text{ can } 3^{-7(\gamma k+s)}/2\text{–bias } g(m_0, m_1)\right) < 2^{-(\gamma k+s)}.$$

Finally, given that there are less than 2^{2k} pairs v_0, v_1, taking $\gamma \geq 2$, and using the fact that $3^{-7(\gamma k+s)}/2 \leq 2^{-k-1-s/2}$, we conclude as desired that

$$\text{Prob}\left(\exists \text{ non-trivial } g \text{ such that } \tilde{\mathcal{B}} \text{ can } 2^{-k-1-s/2}\text{–bias } g(m_0, m_1)\right) < 2^{2k}2^{-(\gamma k+s)} \leq 2^{-s}.$$

■

To conclude that, except with probability 2^{-s}, $\mathcal{B}$ has no more than 2^{-s} bit of information on at least one of m_0 or m_1 immediately after Step 3, and therefore no more than 2^{-s} bit of information on at least one of w_0 or w_1 at the end of the protocol (even if he is given the other string—see the Appendix for formal definitions), it suffices to apply the following theorem with $\varepsilon = 1/2^{k+1+s/2}$.

Theorem 2.5 *Let k be an integer and $\varepsilon \leq 1/2^{k+1}$. Consider a k–bit string m so that $\mathcal{B}$ cannot ε–bias any non-trivial linear function of the bits of m. Then $\mathcal{B}$'s information on m in the sense of Shannon is less than $(2^{k+1}\varepsilon)^2$ bit.*

Proof (sketch). Let X be the random variable over the binary strings of length k that corresponds to $\mathcal{B}$'s probability distribution on m. Consider the set G of all non-trivial linear functions on k–bit strings: there are exactly $2^k - 1$ such functions. For any $g \in G$, let p_g be the probability that $g(X) = 0$. We have $\frac{1}{2} - \varepsilon < p_g < \frac{1}{2} + \varepsilon$ for all $g \in G$ by assumption that $\mathcal{B}$ cannot ε–bias non-trivial linear functions of the bits of m.

It is easily shown that the probability that $X = x$ for any given string x is given by

$$\mathrm{Prob}\,(X = x) \;=\; 2^{-k} + \left(\sum_{g \in G} s(g,x) \times (2p_g - 1) \right) / 2^k$$

for some function $s : G \times \{0,1\}^k \to \{-1,1\}$ whose detail does not concern us. It follows that $\mathrm{Prob}\,(X = x)$ differs from 2^{-k} by less than the largest value of $2p_g - 1$ in absolute value, which is less than 2ε. The random variable X that would give the most information to $\mathcal{B}$, yet respect the above constraint, would have half the strings with probability $2^{-k} - 2\varepsilon$ and the other half with probability $2^{-k} + 2\varepsilon$. Therefore,

$$\begin{aligned}
\mathbf{H}(X) &\leq -2^{k-1}(2^{-k} - 2\varepsilon)\lg(2^{-k} - 2\varepsilon) - 2^{k-1}(2^{-k} + 2\varepsilon)\lg(2^{-k} + 2\varepsilon) \\
&= \left(\frac{(2^{k+1}\varepsilon)^2}{1 \times 2} + \frac{(2^{k+1}\varepsilon)^4}{3 \times 4} + \frac{(2^{k+1}\varepsilon)^6}{5 \times 6} + \frac{(2^{k+1}\varepsilon)^8}{7 \times 8} + \cdots \right) / \ln 2 \\
&< (2^{k+1}\varepsilon)^2.
\end{aligned}$$

■

3 Open Problems

As a further generalization, consider any $\alpha < 2$. An α–$\binom{2}{1}$–UOT$_2$ is a cryptographic protocol for two participants that enables a sender $\mathcal{A}$ to transfer α bits of information, in the sense of Shannon, about two bits (b_0, b_1) to a receiver $\mathcal{B}$ who chooses secretly which information $\Omega_{(b_0,b_1)}$ he gets from her input bits. We require that $\Omega_{(x,y)}$ be a distribution such that $\mathbf{H}\left((B_0, B_1)|\Omega_{(B_0,B_1)}\right) \geq 2 - \alpha$ when B_0 and B_1 are uniformly distributed over $\{0,1\}$. This is done in an all-or-nothing fashion: $\mathcal{B}$ cannot get more information about b_0 and b_1 than a sample from $\Omega_{(b_0,b_1)}$ for some Ω, however malicious or (computationally) powerful he is, and that $\mathcal{A}$ finds out nothing about the choice Ω of $\mathcal{B}$.

Conjecture 3.1 *For all $\alpha < 2$ (or perhaps merely for all $\alpha \leq 1$?), Protocol 2.1 remains private even if occurrences of $\binom{2}{1}$–OT_2 are replaced with α–$\binom{2}{1}$–UOT_2, provided $n \geq \beta_\alpha(k+s)$ for an appropriate constant β_α to be determined, where s is the safety parameter.*

Conjecture 3.2 *If conjecture 3.1 fails as stated, it works if Shannon entropy is replaced with Rényi entropy of order 2 in the definition of α–$\binom{2}{1}$–UOT_2 [BBCM95].*

References

[Bea91] D. Beaver, "Foundations of Secure Interactive Computing", *Advances in Cryptology: Crypto '91 Proceedings*, Springer-Verlag, 1992, pp. 377 – 391.

[BBCM95] C.H. Bennett, G. Brassard, C. Crépeau and U.M. Maurer, "Generalized Privacy Amplification", *IEEE Transaction on Information Theory*, Vol 41, no. 6, November 1995, pp. 1915 – 1923.

[BBR88] C.H. Bennett, G. Brassard and J.–M. Robert, "Privacy Amplification by Public Discussion", *SIAM Journal on Computing*, Vol. 17, no. 2, April 1988, pp. 210 – 229.

[BCR86] G. Brassard, C. Crépeau and J.–M. Robert, "Information Theoretic Reductions Among Disclosure Problems", 27^{th} *Symp. of Found. of Computer Sci.*, 1986, pp. 168 – 173.

[BCR86a] G. Brassard, C. Crépeau and J.–M. Robert, "All-or-Nothing Disclosure of Secrets (Extended abstract)", *Advances in Cryptology: Proceedings of Crypto '86*, Springer-Verlag, 1987, pp. 234 – 238.

[BCS96] G. Brassard, C. Crépeau and M. Sántha, "Oblivious Transfers and Intersecting Codes", Submitted for publication to *IEEE Transactions on Information Theory*, 1996.

[CW79] J.L. Carter and M.N. Wegman, "New Hash Functions and Their Use in Authentication and Set Equality", *Journal of Computer and System Sciences*, Vol. 22, 1981, pp. 265 – 279.

[CL85] G.D. Cohen and A. Lempel, "Linear Intersecting Codes", *Discrete Mathematics*, Vol. 56, 1985, pp. 35 – 43.

[CZ94] G.D. Cohen and G. Zémor, "Intersecting Codes and Independent Families", *IEEE Transactions on Information Theory*, Vol. 40, no. 6, November 1994, pp. 1872 – 1881.

[Cré88] C. Crépeau, "Equivalence Between Two Flavours of Oblivious Transfers (Abstract)", *Advances in Cryptology: Proceedings of Crypto '87*, Springer-Verlag, 1988, pp. 350 – 354.

[Cré89] C. Crépeau, "Verifiable Disclosure of Secrets and Application", *Advances in Cryptology: Proceedings of Eurocrypt '89*, Springer-Verlag, 1989, pp. 181 – 191.

[Cré90] 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.

[Cré94] C. Crépeau, "Quantum Oblivious Transfer", *Journal of Modern Optics*, Vol. 41, no. 12, December 1994, pp. 2445 – 2454.

[CGT95] C. Crépeau, J. van de Graaf and A. Tapp, "Committed Oblivious Transfer and Private Multi-Party Computations", *Advances in Cryptology: Proceedings of Crypto '95*, August 1995, pp. 110 – 123.

[CK88] C. Crépeau and J. Kilian, "Achieving oblivious transfer using weakened security assumptions", 28^{th} *Symp. on Found. of Computer Sci.*, 1988, pp. 42 – 52,

[CS95] C. Crépeau and L. Salvail, "Oblivious Verification of Common String", to appear in *CWI Quarterly*.

[CS91a] C. Crépeau and M. Sántha, "On the Reversibility of Oblivious Transfer", *Advances in Cryptology: Proceedings of Eurocrypt '91*, Springer-Verlag, 1991, pp. 106 – 113.

[CS91b] C. Crépeau and M. Sántha, "Efficient Reductions Among Oblivious Transfer Protocols Based on New Self-Intersecting Codes", *Sequences II, Methods in Communications, Security and Computer Science*, Springer-Verlag, 1991, pp. 360 – 368.

[EGL83] S. Even, O. Goldreich and A. Lempel, "A Randomized Protocol for Signing Contracts", *Proceedings of Crypto 82*, Plenum Press, New York, 1983, pp. 205 – 210.

[GL89] S. Goldwasser and L. Levin, "Fair Computation of General Functions in Presence of Immoral Majority", *Advances in Cryptology: Crypto '90 Proceedings*, Springer-Verlag, 1991, pp. 77 – 93.

[GMR89] S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof-Systems", *SIAM Journal on Computing*, Vol. 18, 1989, pp. 186 – 208.

[GMW87] O. Goldreich, S. Micali and A. Wigderson, "How to Play Any Mental Game, or: A Completeness Theorem For Protocols With Honest Majority", *Proc. 19th ACM Symposium on Theory of Computing*, 1987, pp. 218 – 229.

[GV87] O. Goldreich, R. Vainish, "How to Solve Any Protocol Problem — An Efficiency Improvement", *Advances in Cryptology: Crypto'87*, Springer-Verlag, 1988, pp. 73 – 86.

[Kil88] J. Kilian, "Founding Cryptography on Oblivious Transfer", *Proc. 20th ACM Symposium on Theory of Computing*, 1988, pp. 20 – 31.

[Kil91] J. Kilian, "A General Completeness Theorem For Two-party Games", *Proc. 23th ACM Symposium on Theory of Computing*, 6 – 8 May 1991, pp. 553 – 560.

[MR91] S. Micali and P. Rogaway, "Secure Computation", *Advances in Cryptology: Crypto '91 Proceedings*, Springer-Verlag, 1992, pp. 392 – 404.

[Rab81] M. O. Rabin, "How to Exchange Secrets by Oblivious Transfer", Technical Memo TR–81, Aiken Computation Laboratory, Harvard University, 1981.

[Wie70] S. Wiesner, "Conjugate Coding", *Sigact News*, Vol. 15, no. 1, 1983, pp. 78 – 88. Original manuscript written circa 1970.

A Appendix

Appendix A.1 is taken almost verbatim from [BCS96]. On the other hand, Appendix A.2 contains original material.

A.1 Information Theoretic Definition of Oblivious Transfer

A cryptographic protocol is a multi-party synchronous program that describes for each party the computations to be performed or the messages to be sent to some other party at each point in time. The protocol terminates when no party has any message to send or information to compute. The protocols we describe in this paper all take place between two parties $\mathcal{A}$ and $\mathcal{B}$. We denote by $\bar{\mathcal{A}}$ and $\bar{\mathcal{B}}$ the *honest* programs to be executed by $\mathcal{A}$ and $\mathcal{B}$: honest parties behave according to $\bar{\mathcal{A}}$ and $\bar{\mathcal{B}}$ and no other program. In the following definitions of *correctness* and *privacy* we also consider alternative *dishonest* programs $\tilde{\mathcal{A}}$ and $\tilde{\mathcal{B}}$ executed by $\mathcal{A}$ or $\mathcal{B}$

in a effort to obtain unauthorized information from one another. The definitions specify the result of honest parties interacting together through a specific protocol as well as the possible information leakage of an honest party facing a dishonest party. We are not concerned with the situation where both parties may be dishonest as they can do anything they like in that case; we are only concerned with protecting an honest party against a dishonest party. At the end of each execution of a protocol, each party will issue an "accept" or "reject" verdict regarding their satisfaction with the behaviour of the other party. Two honest parties should always issue "accept" verdicts at the end of their interactions. An honest party will issue a "reject" verdict at the end of a protocol if he received some message from the other party of improper format or some message not satisfying certain conditions specified by the protocol. We also implicitly assume certain time limits for each party to issue messages to each other: after a specified amount of time a party will give up interacting with the other party and issue a "reject" verdict.

As discussed in the Introduction, an $\binom{2}{1}$–OT_2 is a cryptographic protocol for two participants that enables a sender $\mathcal{A}$ to transfer one of two bits b_0 or b_1 to a receiver $\mathcal{B}$ who chooses secretly which bit b_c he gets. This is done in an all-or-nothing fashion, which means that $\mathcal{B}$ cannot get partial information about b_0 and b_1 at the same time, however malicious or (computationally) powerful he is, and that $\mathcal{A}$ finds out nothing about the choice c of $\mathcal{B}$. A generalization of $\binom{2}{1}$–OT_2 is the $\binom{2}{1}$–OT_2^k, in which the bits b_0 and b_1 are replaced by k–bit strings w_0 and w_1 from which $\mathcal{B}$ is given to choose one.

Formally speaking we describe a two-party protocol that satisfies the following constraints of *correctness* and *privacy*. These notions have been defined before for general protocols by Crépeau [Cré90], Micali and Rogaway [MR91] and Beaver [Bea91] using simulators. In this paper, we use the language of information theory to express definitions similar to those introduced by Crépeau [Cré94]. Our goal is *not* to discuss definitions for general two-party protocols: we restrict our study to oblivious transfers.

Let $[P_0, P_1](a)(b)$ be the random variable (since P_0 and P_1 may be probabilistic programs) that describes the outputs obtained by $\mathcal{A}$ and $\mathcal{B}$ when they execute together the programs P_0 and P_1 on respective inputs a and b. Similarly, let $[P_0, P_1]^*(a)(b)$ be the random variable that describes the total information (including not only messages received and issued by the parties but also the result of any local random sampling they may have performed) acquired during the execution of protocol $[P_0, P_1]$ on inputs a and b. Let $[P_0, P_1]_P(a)(b)$ and $[P_0, P_1]^*_P(a)(b)$ be the marginal random variables obtained by restricting the above to only one party P. The latter is often called the *view* of P [GMR89]. In the following definition, the equality sign $(=)$ means that the distributions on the left hand side and the right hand side are the same. When required, we shall use more flexible definitions that would allow an exponentially small probability of failure or an exponentially small amount of unauthorized information leakage. The more flexible definitions could be formalized in much the same way: details are left to the reader.

Definition A.1 (Correctness) Protocol $[\bar{\mathcal{A}}, \bar{\mathcal{B}}]$ is *correct* for $\binom{2}{1}$–OT_2^k if

- $\forall\, w_0, w_1 \in \{0,1\}^k, c \in \{0,1\}$
$$[\bar{\mathcal{A}}, \bar{\mathcal{B}}](w_0, w_1)(c) = (\epsilon, w_c) \tag{1}$$
- for any program $\tilde{\mathcal{A}}$ there exists a probabilistic program $\tilde{\mathcal{A}}'$ s.t. $\forall\, w_0, w_1 \in \{0,1\}^k, c \in \{0,1\}$
$$\left[\tilde{\mathcal{A}}, \bar{\mathcal{B}}\right]_{\mathcal{B}} (w_0, w_1)(c) \mid \mathcal{B} \text{ accepts } = \left[\bar{\mathcal{A}}, \bar{\mathcal{B}}\right]_{\mathcal{B}} (\tilde{\mathcal{A}}'(w_0, w_1))(c) \mid \mathcal{B} \text{ accepts }. \tag{2}$$

Intuitively, condition (1) means that if the protocol is executed as described, it will accomplish the task it was designed for: $\mathcal{B}$ receives word w_c and $\mathcal{A}$ receives nothing. Condition (2) means that in situations in which $\mathcal{B}$ does not abort, $\mathcal{A}$ cannot induce a distribution on $\mathcal{B}$'s output using a dishonest $\tilde{\mathcal{A}}$ that she could not induce simply by changing the input words and then being honest (which she can always do without being detected). This second condition, called *awareness* in [MR91], is concerned with the future use of the outputs of a protocol. If the output of a protocol is to be used later in another protocol we wish to guarantee that the output distribution of the first protocol corresponds to the expected output distribution prescribed by the task definition. Otherwise trouble may force a party to abort and thus disclose situational information about past executions of the protocol. Although some authors [MR91] insist that correctness and privacy **must** be defined together in the general case, the particular case of Oblivious Transfers allows for independent definitions. Here, no correctness condition involving $\tilde{\mathcal{B}}$ is necessary since $\mathcal{A}$ receives no output.

Let W_0, W_1 and C be the random variables taking values over $\{0,1\}^k$ and $\{0,1\}$ that describe $\mathcal{A}$'s and $\mathcal{B}$'s inputs. We assume that both $\mathcal{A}$ and $\mathcal{B}$ are aware of the joint probability distribution of these random variables $P_{W_0,W_1,C}$. A sample w_0, w_1, c is generated from that distribution and w_0, w_1 is provided as $\mathcal{A}$'s secret input while c is provided as $\mathcal{B}$'s secret input.

We assume for the next definition that the reader is familiar with the notion of *entropy* $\mathbf{H}(X)$ of a random variable X. The mutual *information* of two random variables X and Y is given by $\mathbf{I}(X;Y) = \mathbf{H}(X) - \mathbf{H}(X \mid Y)$. Conditioned by a third random variable Z, $\mathbf{I}(X;Y \mid Z) = \mathbf{H}(X \mid Z) - \mathbf{H}(X \mid Y, Z)$.

Definition A.2 (Privacy) Protocol $[\bar{\mathcal{A}}, \bar{\mathcal{B}}]$ is *private* for $\binom{2}{1}$–OT_2^k if $\forall\, W_0, W_1 \in \{0,1\}^k, C \in \{0,1\}$

- $\forall\, w_0, w_1 \in \{0,1\}^k$ and for any program $\tilde{\mathcal{A}}$
$$\mathbf{I}\left(C; \left[\tilde{\mathcal{A}}, \bar{\mathcal{B}}\right]^*_{\mathcal{A}}(W_0, W_1)(C) \;\middle|\; (W_0, W_1) = (w_0, w_1)\right) = 0 \tag{3}$$
- $\forall\, c \in \{0,1\}$ and for any program $\tilde{\mathcal{B}}$ there exists a random variable $\tilde{C} = \Omega(C) \in \{0,1\}$ s.t.
$$\mathbf{I}\left(W_0, W_1; \left[\bar{\mathcal{A}}, \tilde{\mathcal{B}}\right]^*_{\mathcal{B}}(W_0, W_1)(C) \;\middle|\; C = c, W_{\tilde{C}}\right) = 0. \tag{4}$$

The above two conditions are designed to guarantee that each party is limited to the information he or she should get according to the honest task definition. Condition (3) means that $\tilde{\mathcal{A}}$ cannot acquire any information about C through the protocol. On the other hand, condition (4) means that $\tilde{\mathcal{B}}$ may acquire information about only one of W_0 or W_1 through the protocol. In particular, no joint information about the two words may be obtained by the protocol. This is why our condition assumes that $\tilde{\mathcal{B}}$ is given one of $\mathcal{A}$'s words. We do not require that $\tilde{\mathcal{B}}$ be given W_C because there is no way to prevent him from obtaining any other $W_{\tilde{C}}$ through otherwise honest use of the protocol.

A.2 Generalized Oblivious Transfer

We have now similar conditions to define Generalized Oblivious Transfer.

Definition A.3 (Correctness) Protocol $[\bar{\mathcal{A}}, \bar{\mathcal{B}}]$ is *correct* for $\binom{2}{1}$–GOT_2 if

- $\forall\, b_0, b_1 \in \{0,1\}, f : \{0,1\}^2 \to \{0,1\}$

$$[\bar{\mathcal{A}}, \bar{\mathcal{B}}](b_0, b_1)(f) = (\epsilon, f(b_0, b_1)) \tag{5}$$

- for any program $\tilde{\mathcal{A}}$ there exists a probabilistic program $\tilde{\mathcal{A}}'$ s.t.
  $\forall\, b_0, b_1 \in \{0,1\}, f : \{0,1\}^2 \to \{0,1\}$

$$\left[\tilde{\mathcal{A}}, \bar{\mathcal{B}}\right]_{\bar{\mathcal{B}}}(b_0, b_1)(f) \mid \bar{\mathcal{B}} \text{ accepts } = \left[\bar{\mathcal{A}}, \bar{\mathcal{B}}\right]_{\bar{\mathcal{B}}}(\tilde{\mathcal{A}}'(b_0, b_1))(f) \mid \bar{\mathcal{B}} \text{ accepts }. \tag{6}$$

Let B_0, B_1 and F be the random variables taking values over $\{0,1\}$ and $\{0,1\}^2 \to \{0,1\}$ that describe $\mathcal{A}$'s and $\mathcal{B}$'s inputs. We assume that both $\mathcal{A}$ and $\mathcal{B}$ are aware of the joint probability distribution of these random variables $P_{B_0,B_1,F}$. A sample b_0, b_1, f is generated from that distribution and b_0, b_1 is provided as $\mathcal{A}$'s secret input while f is provided as $\mathcal{B}$'s secret input.

Definition A.4 (Privacy) Protocol $[\bar{\mathcal{A}}, \bar{\mathcal{B}}]$ is *private* for $\binom{2}{1}$–GOT_2 if
$\forall\, B_0, B_1 \in \{0,1\}, F : \{0,1\}^2 \to \{0,1\}$

- $\forall\, b_0, b_1 \in \{0,1\}$ and for any program $\tilde{\mathcal{A}}$

$$\mathbf{I}\left(F; \left[\tilde{\mathcal{A}}, \bar{\mathcal{B}}\right]^*_{\mathcal{A}}(B_0, B_1)(F) \;\middle|\; (B_0, B_1) = (b_0, b_1)\right) = 0 \tag{7}$$

- $\forall f : \{0,1\}^2 \to \{0,1\}$ and for any program $\tilde{\mathcal{B}}$ there exists a random variable $\tilde{F} = \Omega(F) : \{0,1\}^2 \to \{0,1\}$ s.t.

$$\mathbf{I}\left(B_0, B_1; \left[\bar{\mathcal{A}}, \tilde{\mathcal{B}}\right]^*_{\mathcal{B}}(B_0, B_1)(F) \;\middle|\; F = f, \tilde{F}(B_0, B_1)\right) = 0. \tag{8}$$