Efficient Cryptographic Protocols based on Noisy Channels

Claude Crépeau
Université de Montréal *

April 9, 1996

Abstract

The Wire-Tap Channel of Wyner [20] shows that a Binary Symmetric Channel may be used as a basis for exchanging a secret key, in a cryptographic scenario of two honest people facing an eavesdropper. Later Crépeau and Kilian [9] showed how a BSC may be used to implement Oblivious Transfer in a cryptographic scenario of two possibly dishonest people facing each other. Unfortunately this result is rather impractical as it requires $\Omega(n^{11})$ bits to be transmitted through the BSC to accomplish a single OT. The current paper provides efficient protocols to achieve the cryptographic primitives of Bit Commitment and Oblivious Transfer based on the existence of a Binary Symmetric Channel. Our protocols respectively require sending $O(n)$ and $O(n^3)$ bits through the BSC. These results are based on a technique known as Generalized Privacy Amplification [1] that allow two people to extract secret information from partially compromised data.

1 Introduction

The cryptographic power of a noisy channel has been demonstrated by Wyner [20] who showed that two honest parties, say $\mathcal{A}lice$ and $\mathcal{B}ob$, can exchange a secret key on which an eavesdropper $\mathcal{E}ve$ may obtain only a small fraction of the information as long as $\mathcal{A}lice$ and $\mathcal{B}ob$ are connected by a Binary Symmetric Channel of better quality than a similar Channel connecting them to $\mathcal{E}ve$. Maurer [17] has later showed that even if $\mathcal{E}ve$'s Channel is better but independent of the Channel between $\mathcal{A}lice$ and $\mathcal{B}ob$, the same result is possible. More recently, a result of Bennett, Brassard, Crépeau and Maurer [1] provides a technique called Generalized Privacy Amplification to ensure that $\mathcal{E}ve$'s information is an arbitrary small fraction of a bit under the same conditions.

But cryptography is no longer interested solely in protecting communications. As a result of public-key cryptography [10], a large number of other cryptographic tasks have emerged. Examples of such tasks are Coin-flipping by telephone [3] and Mental Poker. These may involve two or more parties, some of which may be dishonest. The general concept of Distributed Function Evaluation was first introduced by Yao [21] and later extended to "Mental Games" by Goldreich, Micali and Wigderson [12].

Distributed Function Evaluation and Mental Games are multi-party algorithms which involve secret data that the parties want to keep from one another. In the model where we are ready to

* Département IRO, Université de Montréal, C.P. 6128, succursale centre-ville,
Montréal (Québec), Canada H3C 3J7. e-mail: crepeau@iro.umontreal.ca.

accept computational assumptions, such general tasks can be achieved from basic assumptions such as the existence of a One-Way Trapdoor Function [12].

The lesson derived in the computational model is that very simple protocols are sufficient to achieve the general ones. The two primitives known as Bit Commitment (defined in Section 3) and Oblivious Transfer (defined in Section 4) are elementary protocols that are sufficient in general to accomplish any Mental Games, even in a non-computational scenario [15, 8].

The current paper considers a scenario where only two people, $\mathcal{A}lice$ and $\mathcal{B}ob$, are involved and where we put no limitation on their computing power. If we made no further assumption, it would be impossible to accomplish Mental Games. Thus, the extra assumption we make is that $\mathcal{A}lice$ and $\mathcal{B}ob$ are connected by a Binary Symmetric Channel ($\mathbf{BS}_\epsilon$), that is a channel that will change the value of a bit b with probability ϵ as it travels from one party to the other.

A first protocol to accomplish Oblivious Transfer from a Noisy Channel was presented in [9]. Unfortunately, that protocol is quite complex and requires $\Omega(n^{11})$ bits sent through the BSC to perform a single Oblivious Transfer, where n is a security parameter that specifies the reliability of the protocol. As a consequence, any two-party computations may be performed from the assumption that there exists a reliable BSC. The current solution is by far more efficient than those suggested earlier. The current paper provides a protocol for Bit Commitment that uses $O(n)$ times the $\mathbf{BS}_\epsilon$ and a protocol for Oblivious Transfer that uses $O(n^3)$ times that primitive, where n is a security parameter that specifies the probabilities of failure of the protocols. These probabilities are all exponentially small in n.

2 General Tools

2.1 Error Channel

We consider a standard model of errors: the binary symmetric channel. In the binary symmetric channel $\mathcal{A}lice$ sends a bit to $\mathcal{B}ob$ that is flipped with probability ϵ

$$\mathbf{BS}_\epsilon(x) = \begin{cases} \bar{x} & \text{with prob. } \epsilon \\ x & \text{with prob. } 1-\epsilon. \end{cases}$$

By extension, we also write $\mathbf{BS}_\epsilon(w)$ as a shorthand for $\mathbf{BS}_\epsilon(w_1)\mathbf{BS}_\epsilon(w_2)...\mathbf{BS}_\epsilon(w_n)$ when $w = w_1 w_2 ... w_n$ is an n–bit word.

A nice property of the binary symmetric channel is that it is totally symmetrical between the participants: if $\mathcal{B}ob$ wants to send a bit x via $\mathbf{BS}_\epsilon(x)$ to $\mathcal{A}lice$ when it is only available from $\mathcal{A}lice$ to $\mathcal{B}ob$, they can do as follows:

Protocol 2.1 ($\overline{\mathbf{BS}}_\epsilon(x)$)

1: $\mathcal{A}lice$ picks $r \in_{\mathrm{R}} \{0,1\}$ and runs $\mathbf{BS}_\epsilon(r)$ with $\mathcal{B}ob$ who gets r',

2: $\mathcal{B}ob$ announces $y \leftarrow x \oplus r'$ to $\mathcal{A}lice$,

3: $\mathcal{A}lice$ returns $y \oplus r$.

In general, for the binary symmetric channel, any protocol may be inverted by permuting $\mathcal{A}lice$ and $\mathcal{B}ob$ and replacing $\mathbf{BS}_\epsilon$ by $\overline{\mathbf{BS}}_\epsilon$.

2.2 Coding theory

An $[n, k, d]$ linear code $\mathcal{C}$ is a linear subspace of $\{0,1\}^n$ of dimension k (and cardinality 2^k) such that no two words c_1, c_2 from $\mathcal{C}$ are such that $d_H(c_1, c_2) < d$, except if $c_1 = c_2$, where $d_H(x, y)$ is the Hamming distance between x and y: the number of positions where they differ.

Such a code is defined as the linear combinations of the rows of a generating matrix G of dimension $k \times n$. Alternatively, $\mathcal{C}$ may be defined as the kernel of a parity check matrix H of dimension $n \times (n-k)$. Knowledge of G or H is computationally equivalent as it is easy to get one from the other.

It is a well known fact [16, chap. 17, prob. (30)] that a random binary matrix G of size $Rn \times n$ defines a binary linear code with minimal distance at least ϵn except with probability $2^{(R+\mathbf{H}(\epsilon)-1)n}$.

Some of the codes we use in this paper are efficiently decodable. For instance, concatenated code used in Section 4 may be efficiently decoded. In some situations the information to be decoded is not a codeword. In such a case, as long as the syndrome $syn(w) = H^\top w$ of a word w is known the decoding algorithm may be used to recover w from a noisy version of that word and the value of $syn(w)$. Please consult [16] for more information on coding theory.

2.3 Generalized Privacy Amplification

Let W be a random variable uniformly distributed over $\{0,1\}^n$ and let $\mathbf{BS}_\epsilon(W)$ be another random variable obtained from W through a binary symmetric channel of error rate ϵ, i.e.

$$\text{Prob}\left[\mathbf{BS}_\epsilon(W) = v | W = w\right] = (1-\epsilon)^{n-d_H(w,v)}\epsilon^{d_H(w,v)}.$$

Let G be a random variable taking values $g : \{0,1\}^n \to \{0,1\}^r$ uniformly distributed from a *universal*$_2$ class of hash functions [6]. Let $\mathbf{H}(x) = -x \lg x - (1-x)\lg(1-x)$ be the binary entropy function. It is shown in [1] that

Theorem 2.1 *For any $\delta > 0$ and all sufficiently large n*

$$H(G(W)|\mathbf{BS}_\epsilon(W), G) \geq r - \frac{2^{-s}}{\ln 2}$$

where $s = n(\mathbf{H}(\epsilon) - \delta) - r$.

Let $syn : \{0,1\}^n \to \{0,1\}^t$ be a linear function. According to [2, 5, 1] for the special case where syn is a rank t linear function we have

Theorem 2.2 *For any $\sigma \in \{0,1\}^t$, $\delta > 0$ and all sufficiently large n*

$$H(G(W)|\mathbf{BS}_\epsilon(W), syn(W) = \sigma, G) \geq r - \frac{2^{t-s}}{\ln 2}.$$

where $s = n(\mathbf{H}(\epsilon) - \delta) - r$.

Since $H(G(W)|\mathbf{BS}_\epsilon(W), syn(W) = \sigma, G) = r$ means that no information about $G(W)$ is given by $(\mathbf{BS}_\epsilon(W), syn(W) = \sigma, G)$, the above result is exponentially close to the best possible: $(\mathbf{BS}_\epsilon(W), syn(W) = \sigma, G)$ contains almost no information about $G(W)$.

3 Bit Commitment

Assume that a party, $\mathcal{A}lice$, has a bit b in mind, to which she would like to be committed toward another party, $\mathcal{B}ob$. That is, $\mathcal{A}lice$ wishes, through a procedure $\mathbf{BC}(b)$, to provide $\mathcal{B}ob$ with a piece of evidence w that she has a bit b in mind and that she cannot change it (*unicity*). Meanwhile, $\mathcal{B}ob$ should not be able to tell from that evidence what b is (*confidentiality*). At a later time, $\mathcal{A}lice$ can reveal, through an unveiling procedure $\mathbf{UN}(b,p)$, the value of b and prove through p to $\mathcal{B}ob$ that the piece of evidence sent earlier (w) really corresponded to that bit.

Bit commitment schemes have several applications in the field of cryptographic protocols. In particular one can implement *zero-knowledge proofs* of a variety of statements using bit commitment schemes [14, 13, 4]. The first implementations of bit commitment schemes were given in a computational complexity scenario [3]. Unfortunately, proofs of their (computational) security have always required an unproven assumption since otherwise they would imply very strong results such as $\mathcal{P} \neq \mathcal{NP}$.

This section is inspired by that work of [5] to achieve Bit Commitment in the model of Quantum Cryptography.

3.1 Bit Commitment from Binary Symmetric Channel

3.1.1 Intuition behind Protocols BC & UN

After establishing a proper error-correcting code, $\mathcal{A}lice$ sends a codeword from that code to $\mathcal{B}ob$ through the BS_ϵ. The code is such that $\mathcal{B}ob$ should have many candidates for $\mathcal{A}lice$'s codeword after seeing it through the BS_ϵ. The secret bit of $\mathcal{A}lice$ is given by applying a random function from a $universal_2$ class to the codeword. To unveil her bit, $\mathcal{A}lice$ discloses her codeword. She should not be able to announce two codewords that $\mathcal{B}ob$ will find close enough to what he saw to believe her.

3.1.2 Formal Protocol

Let ϵ be the error probability of the channel, and $\gamma < 1$ be a positive number. Let $\delta > 0$ be such that $\mathbf{H}(\epsilon) - \delta > \mathbf{H}(\gamma\epsilon)$ and such that $(\mathbf{H}(\epsilon) - \delta)n$ is an integer. The following protocols work *for any value of* ϵ such that $0 < \epsilon < 1/2$, in contrast to the protocols of Section 4.

Protocol 3.1 (BC(b))

1: $\mathcal{B}ob$ chooses and announces to $\mathcal{A}lice$ a binary linear $[n,k,d]$-code $\mathcal{C}$ with parameters $k = (1 - \mathbf{H}(\epsilon) + \delta)n$ and $d \geq \gamma\epsilon n$.

2: $\mathcal{A}lice$

- picks a random n-bit string m and announces it to $\mathcal{B}ob$,
- picks a random codeword $c \in \mathcal{C}$ such that $c \odot m = b$,
- $\underset{i=1}{\overset{n}{\mathbf{DO}}}$ runs $BS_\epsilon(c_i)$ with $\mathcal{B}ob$ who receives c'_i,
- returns c, b.

3: $\mathcal{B}ob$ sets $c' \leftarrow (c'_1 c'_2 \ldots c'_n)$ and returns $(\mathcal{C}, m, c')$.

$\mathcal{B}ob$ keeps c' secret forever, whereas $\mathcal{A}lice$ keeps b and c secret until (and if) unveiling takes place. If $\mathcal{A}lice$ subsequently decides to unveil her commitment, she initiates the next protocol with $\mathcal{B}ob$.

There exists a positive number $\lambda < \gamma(1/2-\epsilon)/2$ such that an honest $\mathcal{A}lice$ is likely to satisfy the following with overwhelming probability while a dishonest $\mathcal{A}lice$ is unable to open the commitment as both bits with overwhelming probability.

Protocol 3.2 ($\mathbf{UN}(c,b),(\mathcal{C},m,c')$)

1: **if** $(c \in \mathcal{C}) \wedge (b = c \odot m) \wedge (d_H(c,c') - \epsilon n < \lambda n)$ **then** $\mathcal{B}ob$ accepts **else** $\mathcal{B}ob$ rejects.

3.1.3 Details of the Protocol

In the above Protocol **BC** we ask $\mathcal{B}ob$ to choose a code with specific parameters. The effect of these parameters on the security of the protocol explain why we require $\mathcal{B}ob$ to do this job and not $\mathcal{A}lice$: the bigger d is, the more unlikely it is for $\mathcal{A}lice$ to cheat and the bigger k is, the more unlikely it is for $\mathcal{B}ob$ to cheat. Coding theory give us limits on how big d and k can be at the same time. In order to have them as large as possible at the same time, the best construction known to this day is to pick the generating matrix of the code at random. Nevertheless, in this case the value of k is easy to figure out from the matrix (the rank of the matrix) while the exact value of d is more difficult to determine. All we know is that it is likely to be high.

As discussed in Section 2.2, a random binary matrix G of size $Rn \times n$ defines a binary linear code with minimal distance at least ϵn except with probability $2^{(R+\mathbf{H}(\epsilon)-1)n}$, thus $\mathcal{B}ob$ has an exponentially small probability of having d too small when he picks a $k \times n$ matrix at random. $\mathcal{A}lice$ can easily verify that the value of k is correct.

The random vector m is used to define a Privacy Amplification Function of $\{0,1\}^n$ to $\{0,1\}$.

3.2 Analysis of the Protocol

Confidentiality Let C and M be the random variables describing $\mathcal{B}ob$'s possibilities for c and m. Before c is sent through the BSC, C is uniformly distributed among all the possible codewords of $\mathcal{C}$ and M among all possible n–bit strings. Let $0 < \delta' < \delta$. We are in the scenario of Theorem 2.2 with $r = 1$, $t = (\mathbf{H}(\epsilon) - \delta)n$, and $s = (\mathbf{H}(\epsilon) - \delta')n - 1$. We therefore conclude that seeing a codeword c through a BSC and learning m is not enough to know much about $c \odot m$:

Theorem 3.1 *For any all sufficiently large n*

$$H(C \odot M | \mathbf{BS}_\epsilon(C), syn(C) = (0,0,...,0), M) \geq 1 - \frac{2^{(\delta'-\delta)n+1}}{\ln 2}.$$

Unicity An honest $\mathcal{A}lice$ sends a random codeword c through the channel. Consider the random variable $d_H(c, \mathbf{BS}_\epsilon(c))$. It is easy to calculate that $E(d_H(c,\mathbf{BS}_\epsilon(c))) = \epsilon n$ and by Bernstein's law of large numbers Prob $[d_H(c,\mathbf{BS}_\epsilon(c)) > \epsilon n + \lambda n]$ is exponentially small in n for all λ sufficiently small, and all sufficiently large n. A dishonest $\mathcal{A}lice$ sends any word w through the channel and later would like to claim c_0 or c_1 to unveil as 0 or 1. One of these, say c_z, is such that $d_H(c_z, w) > \gamma\epsilon n/2$. Consider the random variable $d_H(c_z, \mathbf{BS}_\epsilon(w))$. It is easy to calculate that $E(d_H(c_z,\mathbf{BS}_\epsilon(w))) \geq \epsilon n + \gamma(1/2-\epsilon)n$ and by Bernstein's law of large numbers Prob $[d_H(c_z,\mathbf{BS}_\epsilon(w)) < \epsilon n + \gamma(1/2-\epsilon)n - \lambda n]$ is exponentially small in n for all λ sufficiently small, and all sufficiently large n.

Thus any $\lambda < \gamma(1/2-\epsilon)/2$ will satisfy our requirements that an honest $\mathcal{A}lice$ will succeed except with probability exponentially small in n, while a dishonest $\mathcal{A}lice$ will succeed to open both ways only with probability exponentially small in n.

4 Oblivious Transfer

One-out-of-two Oblivious Transfer, denoted $\binom{2}{1}$–OT_2, is a primitive that originates with [19] (under the label of "multiplexing"). According to this primitive, one party $\mathcal{A}lice$ owns two secret strings w_0 and w_1, and another party $\mathcal{B}ob$ wants to learn w_c for a secret bit c of his choice. $\mathcal{A}lice$ is willing to collaborate provided that $\mathcal{B}ob$ does not learn any information about $w_{\bar{c}}$, but $\mathcal{B}ob$ will not participate if $\mathcal{A}lice$ can obtain information about c.

Similarly, in an Oblivious Transfer [18], $\mathcal{A}lice$ sends a message to $\mathcal{B}ob$ that he receives with probability ϵ (this fact is out of their control) while the empty string ε is otherwise received. $\mathcal{A}lice$ does not find out what happened. $\mathcal{B}ob$ knows if he got the message or nothing. Independently from [19] but inspired by [18], $\binom{2}{1}$–OT_2 was introduced subsequently in [11] with applications to contract signing protocols.

These two simple cryptographic tools have been extensively studied by several researchers because they turned out to be elementary blocks to build more elaborate cryptographic tasks known as "secure computations". This idea introduced by Yao [21] allows $\mathcal{A}lice$ and $\mathcal{B}ob$ to compute a two-argument function on data they would like to keep secret from one another. They find out the output of the function but not their respective inputs. It was shown in a *computational* model that One-out-of-two Oblivious Transfer suffices to perform general secure computations by Goldreich, Micali and Wigderson [12] and later in an abstract (not necessarily computational) model by Kilian [15]. Crépeau showed [7] that indeed Rabin's Oblivious Transfer can also do the job by describing a general technique to turn an Oblivious Transfer into a One-out-of-two Oblivious Transfer. The result of the current section is an extension of that technique.

4.1 Oblivious Transfer from Binary Symmetric Channel

4.1.1 Basic Idea

Simulate $\mathbf{OT}_\epsilon(b)$ with protocol $\widehat{\mathbf{OT}}_\epsilon(b)$ obtained by sending b twice through the binary symmetric channel and then reduce $\binom{2}{1}$–OT_2 to $\widehat{\mathbf{OT}}_\epsilon(b)$ with a Protocol similar to that of [7].

Protocol 4.1 ($\widehat{\mathbf{OT}}_\epsilon(b)$)

1: $\mathcal{A}lice$ runs $\mathbf{BS}_\epsilon(b)\mathbf{BS}_\epsilon(b)$ with $\mathcal{B}ob$ who receives b_0b_1.

2: **if** $b_0 = b_1$ **then** $\mathcal{B}ob$ returns b_0 **else** $\mathcal{B}ob$ returns ε.

The problems with this approach are that $\widehat{\mathbf{OT}}_\epsilon(b)$ makes errors and that Alice can send bad pairs $\bar{b}b$: if $\mathcal{A}lice$ is honest and sends bb through the binary symmetric channel then

$$\text{Prob}\left[\widehat{\mathbf{OT}}_\epsilon(b) = x\right] = \begin{cases} \epsilon^2 & \text{if } x = b \\ (1-\epsilon)^2 & \text{if } x = \bar{b} \\ 2\epsilon & \text{if } x = \varepsilon \end{cases}$$

Let $\alpha = \epsilon^2 + (1-\epsilon)^2$ be the probability that $\mathcal{B}ob$ receives a bit. If instead $\mathcal{A}lice$ is dishonest and sends $\bar{b}b$ or $b\bar{b}$ through the binary symmetric channel then the probability that $\mathcal{B}ob$ receives a bit is $2\epsilon = 1 - \alpha$. If no extra checks are performed, $\mathcal{A}lice$ could send bad pairs and figure out in Protocol 4.2 which set is *good* and which set is *bad* by the fact that good pairs are more likely to have been received.

The errors are first solved (in Protocol 4.2) by the same trick as in [2] using codes to fix them, while the cheating by $\mathcal{A}lice$ is later taken care of (in Protocol 4.3) by running statistics on the frequency of bb pairs. Protocol 4.2 introduces another kind of cheating $\mathcal{A}lice$ could perform that is also solved in Protocol 4.3.

4.1.2 Intuition behind Protocol $\binom{2}{1}$–$\widehat{\mathbf{OT}}_2$

For this first protocol we assume $\mathcal{A}lice$ behaves honestly and will remove this assumption in the final protocol. The idea of the first protocol is that $\mathcal{A}lice$ sends n/α random bits $r_1, r_2, ..., r_{n/\alpha}$ to $\mathcal{B}ob$ using $\widehat{\mathbf{OT}}_\epsilon$. $\mathcal{B}ob$ should receive roughly n of these and lose $(1-\alpha)n/\alpha$. $\mathcal{B}ob$ forms two sets I_0, I_1 of size n and thus defines two strings r'_{I_0}, r'_{I_1} of size n (r' restricted to I_0 and I_1). String r_{I_c} should be entirely known by $\mathcal{B}ob$, while string $r_{I_{\bar{c}}}$ should be entirely unknown by $\mathcal{B}ob$. Nevertheless, because $\widehat{\mathbf{OT}}_\epsilon$ is imperfect, we expect an average of $\frac{\epsilon^2}{\alpha}n$ differences between r_{I_c} and r'_{I_c}.

To correct these errors a code with minimum distance greater than $4\frac{\epsilon^2}{\alpha}n$ is established between the parties. The errors are corrected by having $\mathcal{A}lice$ send the syndrome of the two words $syn(r_{I_0}), syn(r_{I_1})$. Using r'_{I_c} and $syn(r_{I_c})$, $\mathcal{B}ob$ may recover r_{I_c} except with small probability of failure. Nevertheless, this correction information is not sufficient to find out both words $r_{I_c}, r_{I_{\bar{c}}}$ accurately, as long as the dimension of the code is greater than $n/2$.

A privacy amplification function is finally used to extract one secret bit per string, so that one bit may be recovered by $\mathcal{B}ob$ but not both. This function is the scalar product by a random n-bit word m.

4.1.3 First Protocol

Let γ be a number greater than 1.

Protocol 4.2 ($\binom{2}{1}$–$\widehat{\mathbf{OT}}_2(b_0, b_1)(c)$)

1: $\mathop{\mathbf{DO}}\limits_{i=1}^{n/\alpha}$ $\mathcal{A}lice$ picks a random bit r_i and runs $\widehat{\mathbf{OT}}_\epsilon(r_i)$ with $\mathcal{B}ob$ who gets r'_i.

2: $\mathcal{B}ob$ picks two random disjoint sets I_0, I_1 s.t. $|I_0| = |I_1| = n$, $(\forall i\ [i \in I_c \Rightarrow r'_i \neq \varepsilon])$ or $(\forall i\ [r'_i \neq \varepsilon \Rightarrow i \in I_c])$ and sends I_0, I_1.

3: $\mathcal{A}lice$ and $\mathcal{B}ob$ agree on a parity check matrix H of a concatenated code $\mathcal{C}$ with parameters $[n, k > (1/2+\delta)n, d > 4\gamma\frac{\epsilon^2}{\alpha}n]$.

4: $\mathcal{A}lice$

- computes and sends $s_0 \leftarrow syn(r_{I_0})$ and $s_1 \leftarrow syn(r_{I_1})$,
- picks and sends a random n-bit word m,
- computes and sends $\hat{b}_0 \leftarrow b_0 \oplus (m \odot r_{I_0})$ and $\hat{b}_1 \leftarrow b_1 \oplus (m \odot r_{I_1})$.

5: $\mathcal{B}ob$

- recovers r_{I_c} using r'_{I_c}, s_c and the decoding algorithm of $\mathcal{C}$,
- computes and returns $\hat{b}_c \oplus (m \odot r_{I_c})$.

4.1.4 Details and discussion of Protocol $\binom{2}{1}$–$\widehat{\mathbf{OT}}_2$

The code used for this protocol requires the extra property that it must be efficiently decodable. This can be done by using concatenated codes. For small enough ϵ the conditions of Step **3** can be satisfied. Therefore, contrary to Protocol **BC**, this new protocol works only for certain values of ϵ and not for all ϵ. Note that the code is used also to fill in missing bits in r_{I_c} when less than n "good" bits are received at Step **1**.

$\mathcal{B}ob$ is unable to cheat this protocol because whatever way he splits the "good" bits ($r_i' \neq \varepsilon$) between I_0, I_1, he will not be able to put more $(1+\delta)n/2$ good bits in at least one of I_0 or I_1. Since $k > (1/2+\delta)n$ then $syn(r_{I_0}), syn(r_{I_1})$ each contain $n-k$ bits of information, i.e. no more than $(1/2-\delta)n$ bits. Thus, at least one of the two words r_{I_0}, r_{I_1} will be undetermined by at least $\delta n/2 = n-(1+\delta)n/2-(1/2-\delta)n$ bits. Using privacy amplification, this word will contain an exponentially small amount of information about its related bit. Therefore, $\mathcal{B}ob$ cannot learn both of $\mathcal{A}lice$'s bits.

Unfortunately, $\mathcal{A}lice$ can cheat this protocol in two different ways that allow her to figure out $\mathcal{B}ob$'s secret input c: at Step **2** $\mathcal{A}lice$ can send "bad" pairs $r_i\bar{r}_i$ or $\bar{r}_i r_i$ instead of $r_i r_i$ increasing the probability that it is lost ($r_i' = \varepsilon$) by $\mathcal{B}ob$ and at Step **4** she can send a "bad" syndrome leading $\mathcal{B}ob$ to a decoding error. In the first cheat, "bad" pairs are more likely to end up in the "bad" set thus indicating to $\mathcal{A}lice$ which one is more likely to be the "good" and "bad" sets. In the second cheat, if $\mathcal{A}lice$ makes only one syndrome bad then $\mathcal{B}ob$ might have to abort depending on which bit he is trying to get. Protocol 4.3 solves these two problems.

4.1.5 Intuition behind Protocol $\binom{2}{1}$–$\mathbf{OT}_2$

The general idea of this new protocol is to repeat Protocol $\binom{2}{1}$–$\widehat{\mathrm{OT}}_2$ several times for random $b_{\ell,0}, b_{\ell,1}$ and c_ℓ and combine these instances in such a way to prevent $\mathcal{A}lice$'s cheating as above.

More precisely, Protocol $\binom{2}{1}$–$\widehat{\mathrm{OT}}_2$ is repeated n^2 times. We combine the n^2 instances of $\binom{2}{1}$–$\widehat{\mathrm{OT}}_2$ in such a way that $\mathcal{A}lice$ must cheat in each instance if she wants to discover the value of c. Protocol $\widehat{\mathbf{OT}}$ is used a total of n^3/α times. In order to obtain information $\mathcal{A}lice$ must send at least n^2 bad pairs in these protocols. This will make a statistical difference that will be detected with probability almost 1. If $\mathcal{A}lice$ uses less than n^2 bad pairs, she finds out nothing about c. Similarly, if $\mathcal{A}lice$ sends bad syndromes in protocol $\binom{2}{1}$–$\widehat{\mathrm{OT}}_2$ with probability $1/2$ she will be detected by $\mathcal{B}ob$ because he reads according to a random choice. If she uses $O(n)$ such syndromes it is almost certain that $\mathcal{B}ob$ will detect her cheating.

Let n be an odd number. The instances are combined by requesting that $b_{\ell,0} \oplus b_{\ell,1} = b_0 \oplus b_1$ for $1 \leq \ell \leq n^2$. Let $b_{0,0} = \bigoplus_{\ell=1}^{n^2} b_{\ell,0}$ and $b_{0,1} = \bigoplus_{\ell=1}^{n^2} b_{\ell,1}$. These requirements cause that $\bigoplus_{\ell=1}^{n^2} b_{\ell,c_\ell} = b_{0,z}$ for $z = \bigoplus_{\ell=1}^{n^2} c_\ell$. Thus in order to find out which of $b_{0,0}$ or $b_{0,1}$ $\mathcal{B}ob$ is trying to get, $\mathcal{A}lice$ must find out all the c_ℓ.

4.1.6 Full Protocol

Let γ be a number greater than 1 and n be an odd number. An extra index ℓ is added to each variable of the ℓ^{th} iteration of $\binom{2}{1}$–$\widehat{\mathrm{OT}}_2$.

Protocol 4.3 ($\binom{2}{1}$–$\mathbf{OT}_2(b_0, b_1)(c)$)

1: $\mathcal{A}lice$ picks n^2 random bits $b_{1,0}, b_{2,0}, ..., b_{n^2,0}$ and sets $b_{\ell,1} \leftarrow b_0 \oplus b_1 \oplus b_{\ell,0}$, for $1 \leq \ell \leq n^2$.

2: $\mathcal{B}ob$ picks n^2 random bits $c_1, c_2, ..., c_{n^2}$.

3: $\underset{\ell=1}{\overset{n^2}{\mathbf{DO}}}$

1. $\mathcal{A}lice$ runs $\binom{2}{1}$–$\widehat{\mathrm{OT}}_2(b_{\ell,0}, b_{\ell,1})(c_\ell)$ with $\mathcal{B}ob$ who gets b'_ℓ,
2. **if** $d_H(r_{\ell,I_{\ell.c_\ell}}, r'_{\ell,I_{\ell.c_\ell}}) > \gamma\frac{\epsilon^2}{\alpha}n$ **then** $\mathcal{B}ob$ aborts.

4: if $\left(\#\{\ell, i \mid r'_{\ell,i} \neq \varepsilon\} < n^3 - \frac{(1-2\epsilon)^2}{2}n^2\right)$ **then** $\mathcal{B}ob$ aborts

else $\mathcal{B}ob$ computes and sends $c' \leftarrow c \oplus \left(\bigoplus_{\ell=1}^{n^2} c_\ell\right)$.

5: $\mathcal{A}lice$ computes and sends $\hat{b}_0 \leftarrow b_0 \oplus \left(\bigoplus_{\ell=1}^{n^2} b_{\ell,c'}\right)$ and $\hat{b}_1 \leftarrow b_1 \oplus \left(\bigoplus_{\ell=1}^{n^2} b_{\ell,\bar{c}'}\right)$ to $\mathcal{B}ob$.

6: $\mathcal{B}ob$ computes and returns $\hat{b}_c \oplus \left(\bigoplus_{\ell=1}^{n^2} b'_\ell\right)$.

4.1.7 Details of the Protocol

The test of Step **3.2** is to decide if the syndrome sent by $\mathcal{A}lice$ was valid. The value $\gamma\frac{\epsilon^2}{\alpha}n$ is the scope of the decoding algorithm of the concatenated code. If the decoded word was further than this distance then clearly the syndrome was wrong. If the test of Step **4** is negative then $\mathcal{B}ob$ is almost certain that $\mathcal{A}lice$ has not cheated n^2 times over the n^3/α transmissions.

4.2 Analysis of the protocol

Let $z_{i,j} = \begin{cases} 0 & \text{if } r'_{i,j} = \varepsilon \\ 1 & \text{if } r_{i,j'} \neq \varepsilon \end{cases}$. When $\mathcal{A}lice$ sends valid pairs $r_i r_i$ in Protocol clearly we have $E\left(\sum_{i=1}^{n^2}\sum_{j=1}^{n/\alpha} z_{i,j}\right) = n^3$. On the other hand, if $\mathcal{A}lice$ is dishonest and wants to take advantage of this kind of cheating, she must cheat in each of the n^2 iterations of the protocol (if not she will loose completely one of the c_ℓ and thus c). In that case we get $E\left(\sum_{i=1}^{n^2}\sum_{j=1}^{n/\alpha} z_{i,j}\right) \leq n^3 - (2\alpha - 1)n^2 = n^3 - (1-2\epsilon)^2 n^2$.

Theorem 4.1 *There exists a constant $\rho < 1$ with the following properties: when $\mathcal{A}lice$ does not use "bad" pairs then*

$$\mathrm{Prob}\left[\sum_{i=1}^{n^2}\sum_{j=1}^{n/\alpha} z_{i,j} < n^3 - \frac{(1-2\epsilon)^2}{2}n^2\right] < \rho^n$$

whereas, when she cheats n^2 times,

$$\text{Prob}\left[\sum_{i=1}^{n^2}\sum_{j=1}^{n/\alpha} z_{i,j} > n^3 - \frac{(1-2\epsilon)^2}{2}n^2\right] < \rho^n.$$

Sketch of proof. Follows from Bernstein's law of large numbers.

Thus, except with exponentially small probability, an honest $\mathcal{A}lice$ will pass the test of Step **4** while a dishonest $\mathcal{A}lice$ will fail that same test.

If $\mathcal{A}lice$ is honest, the probability that more than $\gamma\frac{\epsilon^2}{\alpha}n$ errors occur during transmission by accident is exponentially small. Thus an honest $\mathcal{A}lice$ who send correct syndromes, is unlikely to fail the test of Step **3.2** while a dishonest $\mathcal{A}lice$ who deliberately sends a wrong syndrome will be detected with probability 1/2, if $\mathcal{B}ob$ happens to use that syndrome at random.

Finally, for the same reasons discussed in Section 4.1.4, because of Privacy Amplification $\mathcal{B}ob$ cannot obtain information about both b_0 and b_1 through the instances of protocol $\binom{2}{1}$–$\widehat{\text{OT}}_2$.

5 Conclusion and Open Question

We have obtained two new protocols for the cryptographic primitives of Bit Commitment and One-out-of-Two Oblivious Transfer based on the existence of a BSC using Privacy Amplification. The protocol for BC requires $O(n)$ uses of the BSC, while the protocol for $\binom{2}{1}$–OT$_2$ requires $O(n^3)$ uses of the BSC. If we combine these protocols with the protocol of Crépeau, van de Graaf and Tapp [8] for Private Multi-Party Computation to achieve any two-party function evaluation which requires $O(n^2)$ BCs and $O(n)$ $\binom{2}{1}$–OT$_2$ per gate, we end up with a protocol requiring a total of $O(n^4)$ uses of the BSC per gate of the computation. Our main open question is to obtain $\binom{2}{1}$–OT$_2$ with only $O(n^2)$ uses of the BSC and thus any two-party computation at a cost of $O(n^4)$ uses of the BSC per gate.

6 Acknowledgments

We thank Gilles Brassard, Jeroen van de Graaf, Joe Kilian, Ueli Maurer, Alain Tapp, and Louis Salvail for support, suggestions and comments on this work.

References

[1] C.H. Bennett, G. Brassard, C. Crépeau, and U.M. Maurer. Generalized Privacy Amplification. *IEEE Transaction on Information Theory*, Volume 41, Number 6, to appear November 1995.

[2] C. H. Bennett, G. Brassard, C. Crépeau, and M.-H. Skubiszewska. Practical quantum oblivious transfer. In *Advances in Cryptology: Proceedings of Crypto '91*, Lecture Notes in Computer Science, Vol. 576, pages 351–366. Springer-Verlag, 1992.

[3] M. Blum. Coin flipping by telephone. In *Proceedings of IEEE Spring Computer Conference*, pages 133–137. IEEE, 1982.

[4] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.

[5] G. Brassard, C. Crépeau, R. Jozsa and D. Langlois, "A quantum bit commitment scheme provably unbreakable by both parties," *Proceedings of 34th IEEE Symposium on Foundations of Computer Science*, 1993, pp. 362–371.

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

[7] C. Crépeau. Equivalence between two flavours of oblivious transfers (abstract). In C. Pomerance, editor, *Advances in Cryptology: Proceedings of Crypto '87*, pages 350–354, Springer-Verlag, 1988.

[8] 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.

[9] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *29th Symposium on Foundations of Computer Science*, pages 42–52. IEEE, 1988.

[10] W. Diffie and M. E. Hellman, "New directions in cryptography", *IEEE Transactions on Information Theory*, Vol. IT–22, 1976, pp. 644 – 654.

[11] 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.

[12] O. Goldreich, S. Micali and A. Wigderson, *How to play any mental game, or: A completeness theorem for protocols with honest majority* In *Proc. 19th ACM Symposium on Theory of Computing*, pages 218–229, ACM, 1987.

[13] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or All languages in $\mathcal{NP}$ have zero-knowledge proof systems. *Journal of the ACM*, 38:691–729, 1991.

[14] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing*, 18:186–208, 1989.

[15] J. Kilian, *Founding cryptography on Oblivious transfer*, 20^{th} ACM Symposium on Theory of Computation, 1988.

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

[17] U.M. Maurer, "Secret key agreement by public discussion from common information", *IEEE Transactions on Information Theory*, Vol. IT–39, 1993, pp. 733–742.

[18] M.O. Rabin, How to exchange secrets by oblivious transfer. Technical Memo TR–81, Aiken Computation Laboratory, Harvard University, 1981.

[19] S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983. Manuscript written *circa* 1970, unpublished until it appeared in SIGACT News.

[20] A.D. Wyner, "The wire-tap channel", *Bell System Technical Journal*, Vol. 54, no. 8, 1975, pp. 1355–1387.

[21] Yao, A. C.–C., "Protocols for secure computations", In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, November 1982, pp. 160 – 164.