Oblivious Verification of Common String

Claude Crépeau and Louis Salvail

Département d'Informatique et R.O.,
Université de Montréal,
C.P. 6128, succursale centre-ville,
Montréal (Québec), Canada H3C 3J7.
e-mail: {crepeau,salvail}@iro.umontreal.ca.

Abstract. We consider a situation where two parties, $\mathcal{A}lice$ and $\mathcal{B}ob$, share a common secret string and would like to mutually check their knowledge of that string. We describe a simple and efficient protocol based on oblivious transfer to check mutual knowledge of a common string in such a way that honest parties will always succeed in convincing each other, while a dishonest party interacting with an honest party will have vanishingly small probability of convincing him. Moreover, a dishonest party gains only a very small amount of information about the secret string from running the protocol: whoever enters the protocol with no knowledge of the secret string would have to enter this protocol an exponential number of times in order to gain non-negligible information about the string.

1 Introduction

Didn't you worry last time you typed your PIN (Personal Identification Number) to an unknown Automated Teller Machine (ATM) that it could be a fake and that the sole purpose of this ATM could be to memorize your PIN? According to a recent headline of the NY Times [24] maybe you should have worried:

"ONE LESS THING TO BELIEVE IN: FRAUD AT FAKE CASH MACHINE"

The problem with current identification systems is that the customer is expected to trust the equipment to which he types his PIN. It is completely trivial to modify an ATM to memorize the PINs that people type to it. PINs are meant to be checked, not transferred. (Consult [2] for more about ATM frauds.)

The problem we discuss in this paper is that of verifying knowledge of a common information for the purpose of identification. This common knowledge may be shared by a pair of people or several persons forming a select club. The problem is sometimes called "Identify Friend-or-Foe" but we call it "Oblivious Verification of Common String" (OVCS). Consider a situation where two parties, $\mathcal{A}lice$ and $\mathcal{B}ob$, want to check mutual knowledge of a common string in such a way that honest parties will always succeed in convincing each other (except with vanishingly small probability), while a dishonest party interacting with an honest party will have only vanishingly small probability of convincing him.

Moreover, a dishonest party gains only a very small amount of information about the secret string from running the protocol: whoever enters the protocol with no knowledge of the secret string would have to enter this protocol an exponential number of times in order to gain non-negligible information about the string.

Of course, it is always possible to completely solve the problem of identification and authentication of messages by classical methods [7] that require exchanging passwords which length are proportional to the number of uses. Unfortunately, this is completely impractical: we want to rely on the existence of a *short* secret to check identity. A similar approach has been suggested in the computational model through the construction of *pseudo-random bit generators* [5] which require only short secret seeds. This solution requires that the parties involved remain in perfect synchronization and that they keep track of the number of times they have identified to each other. They must know at all times which part of the pseudo-random sequence has been used so far. This approach is quite impractical because it does not tolerate unsuccessful identifications resulting in loss of synchronization and does not extend to more than two people because then synchronization is impossible since one never knows exactly who he is identifying.

The above suggests that solutions to the OVCS problem should be memoryless: new executions of the protocol should not depend on earlier runs, since otherwise there will be some synchronization problems. Section 2 considers a computational solution to this problem based on *pseudo-random functions* due to Goldreich, Goldwasser and Micali [16]. As we will discuss later, their solution may be based on the very weak assumption of the existence of *one-way functions.* In the computational model, more sophisticated tools were developed for similar purposes: *Zero-Knowledge Proofs of Identity* [15] introduced in order to provide means by which an honest party may convince another party of his identity in a way that cannot be replayed successfully to another party. This is true even if the verifying party tries his best to extract valuable information out of the proving party. Moreover, a dishonest party attempting to prove an invalid identity will be detected by the verifying party except with vanishingly small probability. Although it is possible to solve the OVCS problem through such proofs, the resulting solution when based on a general computational assumption is much less efficient and elegant than the above.

The OVCS problem has also been extensively studied by Fagin, Naor and Winkler [14] who provided a large number of scenarios where the problem may be considered. From the cryptographic point of view only one of their solutions is secure: a solution based on the existence of a simple cryptographic tool called *one-out-of-two Oblivious Transfer* [13]. This solution is covered in Section 4. Although in the computational model it takes in general a stronger assumption than the existence of a one-way function (as for the first solution) solutions based on Oblivious Transfers are more interesting because they can be implemented in non-computational scenarios. Oblivious Transfer can be implemented under the assumption that quantum mechanics is correct [9, 3] or under the assumption that reliable noisy channels exist [10].

Section 5 describes a new simple and efficient protocol based on the existence of a one-out-of-two Oblivious Transfer. The scheme of Fagin, Naor and Winkler uses $O(n^2)$ one-out-of-two Oblivious Transfers while ours, based on coding theory, uses only $O(n)$ such Transfers.

2 Computational solution to OVCS

This solution to the OVCS problem starts from the notion of a random function: $\mathcal{A}lice$ and $\mathcal{B}ob$ agree on a function $f_\phi : \{0,1\}^n \rightarrow \{0,1\}^n$ specified by a random description string ϕ. Each time they want to check that they share the same string they sample a random point from the corresponding function. If they agree on a single ϕ they will definitely get the same output. If they don't then they are unlikely to get the same output. Even after seeing several pairs $[x_1, f_\phi(x_1)], [x_2, f_\phi(x_2)], ..., [x_n, f_\phi(x_n)]$, one who does not know the random description string ϕ is unable to predict any new pair $[x_{n+1}, f(x_{n+1})]$ with probability better than 2^{-n}.

Protocol 2.1 ($GGM(\phi^A)(\phi^B)$)

1: $\mathcal{B}ob$ picks $x \in_{\mathrm{R}} \{0,1\}^n$ and send it to $\mathcal{A}lice$,
2: $\mathcal{A}lice$ computes $y \leftarrow f_{\phi^A}(x)$ and sends it to $\mathcal{B}ob$,
3: $\mathcal{B}ob$ accepts if and only if $y = f_{\phi^B}(x)$.

The problem with such a solution is that it is necessary that f_ϕ be a random function from all the functions in $\{0,1\}^n \rightarrow \{0,1\}^n$ in order for the value of any new $f_\phi(x_{n+1})$ to be independent of all previously used values. The description string ϕ of such functions has size $n2^n$ bits. The solution of Goldreich,Goldwasser and Micali is to replace the truly random function by a *pseudo-random function*[1].

2.1 Pseudo-random functions

A pseudo-random class of functions [16] is a set of functions with short descriptions (contrary to truly random functions) that achieves the same above property as truly random functions with respect to time-bounded observers: even after seeing several pairs $[x_1, f_\phi(x_1)], [x_2, f_\phi(x_2)], ..., [x_n, f_\phi(x_n)]$, one who does not know the random description string ϕ is unable to *efficiently* predict any new pair $[x_{n+1}, f(x_{n+1})]$ with probability significantly better than 2^{-n}.

2.2 Constructing pseudo-random functions

Pseudo-random functions can be constructed in general from any *cryptographically secure pseudo-random bit generator* by a very elegant construction of [16]. Similarly, these pseudo-random bit generators can be obtained from any *one-way function* by results of [19, 18]. This a rather weak standard computational

[1] a notion presented at the Crypto-Course together with this particular application.

assumption: a function f is *one-way* if there exists an efficient algorithm to calculate $f(x)$ from x but no efficient algorithm to calculate x from $f(x)$. In the OVCS situation, the one-way function f can be made public, a pseudo-random bit generator G can be defined from f and a pseudo-random function $f_\phi : \{0,1\}^n \to \{0,1\}^n$ can be defined from G for any n-bit string ϕ. At all times, ϕ must remain secret since the pseudo-randomness of f_ϕ is otherwise lost.

2.3 Example

Let N be the product of two large primes P, Q, such that $|N| > n$. Define a candidate one-way function $f : x \to x^2 \bmod N$ and an associated pseudo-random bit generator by $G : x \to lsb(x), lsb(f(x)), lsb(f(f(x))), ..., lsb(f^i(x)), ...$ due to Blum, Blum and Shub [4], where lsb stands for "least significant bit".

Denote $G_i^j(x)$ the bits from positions i to j of $G(x)$ and define recursively functions $f_\phi^i : \{0,1\}^i \to \{0,1\}^n$, for $0 \le i < n$ and words w of length i as follows:

$$f_\phi^0(\epsilon) = \phi,\ \ f_\phi^{i+1}(0w) = G_1^n(f_\phi^i(w)),\ \ f_\phi^{i+1}(1w) = G_{n+1}^{2n}(f_\phi^i(w)).$$

The pseudo-random function is f_ϕ^n. The difficulty of predicting outputs of this pseudo-random function is provably equivalent to the difficulty of factoring N by results of [16, 1].

3 Oblivious Transfer solutions to OVCS

In sections 4 and 5 we describe two alternative solutions to section 2. These solutions use more elaborate concepts that we now explore in details.

3.1 Mathematical Notations

For $b \in \{0,1\}$ and scalars x, y we define the selection function

$$(x,y)_{[b]} = \begin{cases} x \text{ if } b = 0 \\ y \text{ if } b = 1 \end{cases} .$$

If x and y are vectors with n components and $\mathbf{b} \in \{0,1\}^n$ then $(x,y)_{[\mathbf{b}]}$ is the concatenation $(x_1,y_1)_{[b_1]}(x_2,y_2)_{[b_2]}...(x_n,y_n)_{[b_n]}$. We denote $\Delta(s,\widehat{s})$ the Hamming distance between s and $\widehat{s}$: the number of positions where s and $\widehat{s}$ differ.

3.2 Fields and Codes

Let $\mathcal{F}_Q$ be the finite field with Q elements. An $[n,k,d]$ linear code C over $\mathcal{F}_Q$ is a linear subspace of $\mathcal{F}_Q^n$ of dimension k (and cardinality Q^k) such that no two words c_1, c_2 from C are such that $\Delta(c_1,c_2) < d$, except if $c_1 = c_2$.

Such a code is defined as the linear combinations of the lines of a generating matrix G of dimension $n \times k$. Alternatively, C maybe defined as the kernel of a parity check matrix H of dimension $n \times (n-k)$. To each such code C is associated a dual $[n, n-k, \widehat{d}]$ code $\widehat{C}$ defined as the kernel of G (or the linear combinations of the lines of H). No non-trivial relation is known between d and $\widehat{d}$.

3.3 Oblivious Transfers

Informally speaking in an Oblivious Transfer (defined by Rabin [22]), $\mathcal{A}lice$ sends a message to $\mathcal{B}ob$ that he receives half the time (this fact is out of their control). $\mathcal{A}lice$ does not find out what happened. $\mathcal{B}ob$ knows if he got the message or nothing. Similarly, in a One-out-of-two Oblivious Transfer (defined by Even, Goldreich and Lempel [13] and Wiesner [25]), $\mathcal{B}ob$ has a bit c and $\mathcal{A}lice$ has two messages m_0, m_1 that she sends him in such a way that he gets m_c at his choosing but not both. $\mathcal{A}lice$ never finds out which message $\mathcal{B}ob$ received. For the rest of the paper we denote this transfer $\binom{2}{1}$–$\mathrm{OT}_Q^n(m_0, m_1)(c)$ where m_0, m_1 are n-symbol messages in $\mathcal{F}_Q$ and c is a bit. When $n = 1$ it is generally omitted.

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 [26] 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 [17] and later in an abstract (non computational) model by Kilian [20]. Crépeau showed [8] that Rabin's Oblivious Transfer is as powerful as One-out-of-two Oblivious Transfer.

The OVCS problem is of course a special case of the general secure computation. Nevertheless, if we were to rely on general reductions from secure computation to Oblivious Transfers we would end up with a very costly protocol. Our goal is to obtain an efficient solution to this problem. We measure our efficiency in terms of One-out-of-two Oblivious Transfer of bits, $\binom{2}{1}$–OT_2.

It has been shown in a computational model that a sufficient hypothesis to construct an $\binom{2}{1}$–OT_2 is the existence of a *one-way trapdoor permutation* by a result of Goldreich, Micali and Wigderson [17]. This is another standard (but stronger than simple one-wayness) computational assumption: a function f is *one-way trapdoor* if it is one-way and if there exists a short auxiliary string y such that there exists an efficient algorithm to calculate x from $f(x)$ and y. The one-way function candidate from Example 2.3 is indeed a trapdoor function (but not a permutation) since the extra information $y = P$ is sufficient to calculate x from $f(x) = x^2$. In non-computational models, as mentioned in the introduction, $\binom{2}{1}$–OT_2 can be implemented under the assumption that quantum mechanics is correct [9, 3] or under the assumption that reliable noisy channels exist [10].

3.4 Oblivious Transfer of two bits

In protocol 5 below, we use a One-out-of-two Oblivious Transfer of two-bit elements from $\mathcal{F}_4$, denoted by $\binom{2}{1}$–OT_4. Each such transfer is achieved securely (even if one party tries to cheat) using only 3 instances of $\binom{2}{1}$–OT_2 by a simple technique of [6]. Let $m_0^0, m_0^1, m_1^0, m_1^1$ and c be some bits and $m_0 = m_0^0 m_0^1$ and $m_1 = m_1^0 m_1^1$ be elements of $\mathcal{F}_4$. One of m_0, m_1 is transferred to $\mathcal{B}ob$ as follows:

Protocol 3.1 ($\binom{2}{1}$–$\mathbf{OT}_4(m_0, m_1)(c)$)

1: $\mathcal{A}lice$ picks $r_0, r_1 \in_{\mathrm{R}} \{0,1\}$,

2: $\mathcal{A}lice$ runs $\left\{ \begin{array}{l} \binom{2}{1}\text{–OT}_2(r_0, r_1)(c) \\ \binom{2}{1}\text{–OT}_2(m_0^0 \oplus r_0, m_1^0 \oplus r_1)(c) \\ \binom{2}{1}\text{–OT}_2(m_0^1 \oplus r_0, m_1^1 \oplus r_1)(c) \end{array} \right\}$ with $\mathcal{B}ob$ who gets $\left\{ \begin{array}{c} s \\ w_0 \\ w_1 \end{array} \right\}$,

3: $\mathcal{B}ob$ returns $w_0 \oplus s, w_1 \oplus s$.

It is easy to see that if $\mathcal{A}lice$ and $\mathcal{B}ob$ are honest, $\binom{2}{1}$–$\text{OT}_4(m_0, m_1)(c)$ will give m_c to $\mathcal{B}ob$ without $\mathcal{A}lice$ learning c nor $\mathcal{B}ob$ learning $m_{\overline{c}}$. It is also straightforward to check that even when one of them misbehaves, she still cannot learn c and he still cannot learn information about both m_0 and m_1 at the same time, even if he uses different values for c in the three instances of $\binom{2}{1}$–OT_2. A complete formal treatment of this argument may be found in [6].

3.5 Oblivious Transfer of several bits

Similarly, in protocol 4 below, we use a One-out-of-two Oblivious Transfer of n-bit strings, denoted by $\binom{2}{1}$–OT_2^n. Each such transfer is achieved securely (even if one party tries to cheat) using only $5n$ instances of $\binom{2}{1}$–OT_2 by a generalization of the above technique due to [12]. Let $m_0^0, m_0^1, ..., m_0^n, m_1^0, m_1^1, ..., m_1^n$ and c be some bits and $m_0 = m_0^0 m_0^1 ... m_0^n$ and $m_1 = m_1^0 m_1^1 ... m_1^n$ be elements of $\{0,1\}^n$. One of m_0, m_1 is transferred to $\mathcal{B}ob$ as follows:

Protocol 3.2 ($\binom{2}{1}$–$\mathbf{OT}_2^n(m_0, m_1)(c)$)

1: $\mathcal{A}lice$ picks $G \in_{\mathrm{R}} \{5n \times n$ binary matrices$\}$ and announces it to $\mathcal{B}ob$,

2: $\mathcal{A}lice$ picks $w_0, w_1 \in_{\mathrm{R}} \{0,1\}^{5n}$ such that $Gw_0 = m_0$ and $Gw_1 = m_1$,

3: $\underset{i=1}{\overset{5n}{\mathbf{DO}}}$ $\mathcal{A}lice$ runs $\binom{2}{1}$–$\text{OT}_2(w_0^i, w_1^i)(c)$ with $\mathcal{B}ob$ who receives z_i,

4: $\mathcal{B}ob$ computes and returns Gz.

It is easy to see that if $\mathcal{A}lice$ and $\mathcal{B}ob$ are honest, $\binom{2}{1}$–$\text{OT}_2^n(m_0, m_1)(c)$ will give m_c to $\mathcal{B}ob$ without $\mathcal{A}lice$ learning c nor $\mathcal{B}ob$ learning $m_{\overline{c}}$. It is still straightforward to check that even when $\mathcal{A}lice$ misbehaves, she cannot learn c but more elaborate to show for $\mathcal{B}ob$. It is proven in [6] that $\mathcal{B}ob$ cannot learn information about both m_0 and m_1 at the same time even by using different values for c in the $5n$ instances of $\binom{2}{1}$–OT_2, as long as the $[5n, n, d]$ code C generated by G is an *intersecting* code: for any non-zero codewords $c_1, c_2 \in C$, there exists a position i where $c_1^i \neq 0 \neq c_2^i$. Furthermore, it is shown in [12] that there exists a constant $\alpha < 1$ such that a random $5n \times n$ binary matrix will define an intersecting code except with probability smaller than α^n. A complete formal treatment of this argument may be found in [12]. In conclusion, except with probability α^n the above protocol is a secure implementation of $\binom{2}{1}$–OT_2^n.

4 An $O(n^2)$ protocol for OVCS

We are now ready to consider a first solution to the OVCS problem based on Oblivious Transfer due to Fagin, Naor and Winkler [14]. In the *GGM* solution above, the string ϕ was used as a seed to define the pseudo-random function; here ϕ is used as a set of indices to access random information chosen by $\mathcal{A}lice$ and $\mathcal{B}ob$ through $\binom{2}{1}$–OT_2. $\mathcal{A}lice$ uses ϕ^A to access some of $\mathcal{B}ob$'s data and $\mathcal{B}ob$ uses ϕ^B to access some of $\mathcal{A}lice$'s data. They use the data they get from the other and the data they believe the other got from them in an identical calculation. If $\phi^A = \phi^B$ then the result of the calculation is the same and likely to be different otherwise. The detailed protocol follows.

Protocol 4.1 ($FNW(\phi^A)(\phi^B)$)

1: $\mathcal{A}lice$ picks $r_0^1, r_1^1, r_0^2, r_1^2, ..., r_0^n, r_1^n \in_{\mathrm{R}} \{0,1\}^n$,
2: $\mathcal{B}ob$ picks $s_0^1, s_1^1, s_0^2, s_1^2, ..., s_0^n, s_1^n \in_{\mathrm{R}} \{0,1\}^n$,
3: $\underset{i=1}{\overset{n}{\mathbf{DO}}}$ $\mathcal{A}lice$ runs $\binom{2}{1}$–$\mathrm{OT}_2^n(r_0^i, r_1^i)(\phi_i^B)$ with $\mathcal{B}ob$ who receives x_i,
4: $\underset{i=1}{\overset{n}{\mathbf{DO}}}$ $\mathcal{B}ob$ runs $\binom{2}{1}$–$\mathrm{OT}_2^n(s_0^i, s_1^i)(\phi_i^A)$ with $\mathcal{A}lice$ who receives y_i,
5: $\mathcal{A}lice$ computes $a \leftarrow \bigoplus_{i=1}^{n} r^i_{\phi_i^A} \oplus y_i$ and sends a to $\mathcal{B}ob$,
6: $\mathcal{B}ob$ computes $b \leftarrow \bigoplus_{i=1}^{n} x_i \oplus s^i_{\phi_i^B}$ and accepts if and only if $a = b$.

4.1 $FNW(\phi, \phi)$ with honest parties

If $\mathcal{A}lice$ and $\mathcal{B}ob$ share the same private sequence $\phi = \phi^A = \phi^B$ then

$$a = \bigoplus_{i=1}^{n} r^i_{\phi_i^A} \oplus y_i = \bigoplus_{i=1}^{n} r^i_{\phi_i^A} \oplus s^i_{\phi_i^B} = \bigoplus_{i=1}^{n} x_i \oplus s^i_{\phi_i^B} = b.$$

Therefore $\mathcal{B}ob$ accepts.

4.2 $FNW(\phi^A, \phi^B)$ with $\phi^A \neq \phi^B$

When at least one of $\mathcal{A}lice$ or $\mathcal{B}ob$ is honest, the r's or s's are still truly random. If $\phi^A \neq \phi^B$ the calculations of a and b involve different sets of r's and s's and thus these results will be completely independent of each other (this is a property of the $\oplus$ function). We therefore conclude that $\mathrm{P}\left(a = b | \phi^A \neq \phi^B\right) = 2^{-n}$.

4.3 Complexity of the FNW protocol

Protocol FNW uses n instances of $\binom{2}{1}$–OT_2^n, where n is the security parameter. Since each of these costs $5n$ instances of $\binom{2}{1}$–OT_2, the total number of $\binom{2}{1}$–OT_2 used is exactly $5n^2$.

5 New $O(n)$ protocol for OVCS

Suppose $\mathcal{A}lice$ and $\mathcal{B}ob$ meet to share $\phi = \phi^A = \phi^B \in_R \{0,1\}^n$. They also agree on a generating matrix G of an $[n,k,d]$ linear code C with some specific conditions[2] on n,k,d to be discussed later. The generating matrix G needs not be secret. In a scenario where many people use a similar verification protocol, the matrix G can be defined once and for all for everybody. Nevertheless, it is necessary that they agree on the properties of the code generated by G at the moment they trust each other and share ϕ.

In order for $\mathcal{A}lice$ to prove to $\mathcal{B}ob$ that she knows ϕ, she transmits a random codeword c taken from C, in such a way that if they agree on the same ϕ he receives c and verifies that it belongs to C, but otherwise receives a rather random string which is unlikely to be in C. As we will understand in Sections 5.2 and 5.3, two randomization steps (**1** and **3**) are necessary to protect an honest $\mathcal{B}ob$ from a malicious $\mathcal{A}lice$ and an honest $\mathcal{A}lice$ from a malicious $\mathcal{B}ob$.

Protocol 5.1 ($CS(\phi^A)(\phi^B)$)

1: $\mathcal{A}lice$ picks $r, s \in_{\mathrm{R}} \mathcal{F}_4^n$,
2: $\mathop{\mathbf{DO}}\limits_{i=1}^{n}$ $\mathcal{A}lice$ runs $\binom{2}{1}$–$\mathrm{OT}_4(r_i, s_i)(\phi_i^B)$ with $\mathcal{B}ob$ who receives v_i,
3: $\mathcal{B}ob$ picks $x, y \in_{\mathrm{R}} \mathcal{F}_4^n$ and announces them to $\mathcal{A}lice$,
4: $\mathcal{A}lice$ picks $c \in_{\mathrm{R}} C$, computes and sends $u \leftarrow c \oplus (r \oplus x, s \oplus y)_{[\phi^A]}$,
5: $\mathcal{B}ob$ accepts if and only if $u \oplus v \oplus (x,y)_{[\phi^B]} \in C$.

5.1 $CS(\phi,\phi)$ with honest parties

If $\mathcal{A}lice$ and $\mathcal{B}ob$ share the same private sequence $\phi = \phi^A = \phi^B$ then $\mathcal{B}ob$ accepts since the value he computes at step **5** is

$$u \oplus v \oplus (x,y)_{[\phi]} = c \oplus (r \oplus x, s \oplus y)_{[\phi]} \oplus (r,s)_{[\phi]} \oplus (x,y)_{[\phi]} = c.$$

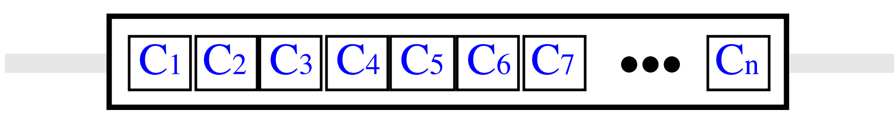

Fig. 1. word seen by an honest $\mathcal{B}ob$ facing an honest $\mathcal{A}lice$.

[2] In order to fulfill these conditions they take C to be a code over $\mathcal{F}_4$.

5.2 $CS(\phi^A, \phi^B)$ with a dishonest $\mathcal{B}ob^*$

First we analyze the situation from the point of view of an honest $\mathcal{A}lice$ facing a malicious $\mathcal{B}ob^*$. The following theorem from [11] tells us a condition on C to sustain the ignorance of a dishonest $\mathcal{B}ob^*$.

Theorem 1 [11]. *Protocol* $CS(\phi^A, \phi^B)$ *hides* ϕ^A *to* $\mathcal{B}ob^*$ *if* $\widehat{C}$ *is an* $[n, n-k, \widehat{d}]$ *code such that* $\widehat{d} \geq (\frac{1}{2} + \gamma)n$*, for* $0 < \gamma < \frac{1}{2}$*, except with exponentially small probability with respect to* n*.*

The intuition behind this theorem is that $\mathcal{B}ob^*$ will be roughly capable of guessing half the bits of ϕ^A and thus will get roughly half the bits of c. The other half of the bits will be random. If $\widehat{C}$ is an $[n, n-k, \widehat{d}]$–code such that $\widehat{d} \geq (\frac{1}{2}+\gamma)n$ for $0 < \gamma < \frac{1}{2}$, the bits seen from c will also be completely random as long as c was chosen at random. $\mathcal{B}ob^*$ thus receives truly random words unless he uses a string ϕ^B very close to ϕ^A. A formal proof of this theorem may be found in [11].

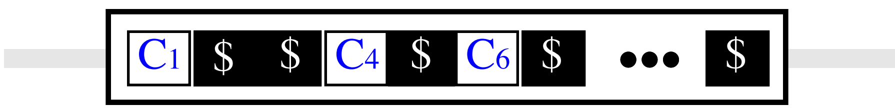

Fig. 2. word seen by a dishonest $\mathcal{B}ob^*$ facing an honest $\mathcal{A}lice$.

5.3 $CS(\phi^A, \phi^B)$ with a dishonest $\mathcal{A}lice^*$

Suppose a malicious $\mathcal{A}lice^*$ tries to impersonate the real $\mathcal{A}lice$. The following theorem from [11] specifies code parameters that allow $\mathcal{B}ob$ to reject $\mathcal{A}lice^*$ with probability exponentially close to one.

Theorem 2 [11]. *Protocol* $CS(\phi^A, \phi^B)$ *makes* $\mathcal{B}ob$ *reject* $\mathcal{A}lice^*$ *if* C *is an* $[n, k, d]$ *code such that* $d \geq \gamma n$*, for* $0 < \gamma < 1$*, except with exponentially small probability with respect to* n*.*

The intuition behind this theorem is as follows. $\mathcal{A}lice^*$ who has complete control over r and s has better set $r = s$ since in that case $v = r = s$. When she does this she knows v exactly. But now she is faced with sending a u that maximizes her probability that $u \oplus (x, y)_{[\phi^B]}$ is a codeword (her only unknown being ϕ^B). Since x and y are chosen at random by $\mathcal{B}ob$ they will vastly differ ($\mathrm{E}[\Delta(x, y)] = \frac{3n}{4}$). If the codewords of C are far enough from each other ($d \geq \gamma n$), only a small fraction of the possibilities for $u \oplus (x, y)_{[\phi^B]}$ are codewords whatever u is. Since $\mathcal{A}lice^*$ does not know ϕ^B, she is unlikely to provide a valid u. A formal proof of this theorem may be found in [11].

5.4 Example

Codes with the above two properties exist over $\mathcal{F}_4$ (but not over $\mathcal{F}_2$). For instance, according to the Varshamov-Gilbert curve [21], a random $n\times 0.91n$ matrix over $\mathcal{F}_4$ is very likely to define a $[n, 0.91n, d]$ code C with $d > 0.02n$ together with a $[n, 0.09n, \widehat{d}]$ dual code $\widehat{C}$ with $\widehat{d} > 0.52n$. Asymptotically, the probability that such a matrix does not define a code with these parameters is exponentially small in n.

5.5 Complexity of the *CS* protocol

Protocol CS uses n instances of $\binom{2}{1}$–OT_4, where n is the security parameter. Since each of these costs 3 instances of $\binom{2}{1}$–OT_2, the total number of $\binom{2}{1}$–OT_2 used is exactly $3n$.

Nevertheless, if we analyze the total running time of this protocol we realize that the most expensive operation (except for the $\binom{2}{1}$–OT_2) is the picking of a codeword and the checking of a codeword, both $O(n^2)$ operations. In general, we ignore this fact because $O(n)$ $\binom{2}{1}$–OT_2 will cost much more operations than that. If $\binom{2}{1}$–OT_2 is available at a cost below $O(n)$ then efficient codes (linear time encodable) such as those of Spielman [23] may be used so to get the code calculations at a cost of $O(n)$ operations.

6 Discussion

The three solutions we have described in this paper offer different advantages to each other. Again, in a computational model, we favor the GGM protocol since it can be built from a simpler assumption and in general will be faster. The other two solutions are more interesting in non-computational models. Of course, the main advantage of the new CS protocol is that it is more efficient than FNW.

Nevertheless, one advantage of protocol FNW is that even if one is given a lot of knowledge on ϕ he cannot progress toward the correct value any better than by running an exhaustive search over the remaining missing information using the protocol with his guesses. In protocol CS, a good guess of ϕ will lead to the correct value very fast, since $\mathcal{B}ob$ can tell when his guess is very close to the right value. For instance in Example 5.4, if $\mathcal{B}ob$ is given more than 52% of the bits of ϕ, he no longer receives truly random words and is able to progress toward ϕ faster than by exhaustive search. Still, this is so unlikely to happen that for all practical purposes, we believe the CS protocol should be used.

Acknowledgments

We thank Gilles Brassard and Jeroen van de Graaf for their interest in this research and fruitful comments. Claude thanks David Chaum for running the 1985 Crypto-Course which was an important event in his early life as a cryptographer.

References

1. Alexi, W., B. Chor, O. Goldreich, and C. P. Schnorr, "RSA and Rabin Functions: Certain Parts Are as Hard as the Whole". In *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, 1984, pp. 449 – 457.
2. Anderson, R. J., "Why Cryptosystems Fail", In *Proceedings of the 1993 ACM Conference in Computer and Communications Security*, pp. 215 – 227.
3. Bennett, C.H., 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, Springer-Verlag, August 1991, Berlin, pp. 351 – 366.
4. Blum, M., L. Blum and Shub, "A simple unpredictable pseudo-random number generator", *SIAM J. Computing*, 15(2):364–383, May 1986.
5. Brassard, G., "On computationally secure authentication tags requiring short secret shared keys", In *Advances in Cryptology: Proceedings of CRYPTO '82*, Plenum Press, 1983, pp. 79 – 86.
6. Brassard, G., C. Crépeau, and J.–M. Robert, "Information theoretic reductions among disclosure problems", In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, November 1986, pp. 168 – 173.
7. Carter, J.L., 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.
8. Crépeau, C., "Equivalence between two flavours of oblivious transfers", In *Advances in Cryptology: Proceedings of Crypto '87*, Lecture Notes in Computer Science, Springer-Verlag, August 1987, pp. 350 – 354.
9. Crépeau, C., "Quantum Oblivious Transfer", Journal of Modern Optics, vol. 41, no 12, pp. 2455 – 2466, 1994.
10. Crépeau, C. and J. Kilian, "Achieving oblivious transfer using weakened security assumptions", In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pp. 42 – 52, 1988.
11. Crépeau, C. and L. Salvail, "Quantum Oblivious Mutual Identification", In *Advances in Cryptology: Proceedings of Eurorypt '95*, Lecture Notes in Computer Science, Springer-Verlag, 1995, pp. 133 – 146.
12. Crépeau, C. and M. Sántha, "Efficient reductions among oblivious transfer protocols based on new self-intersecting codes", In *Sequences II, Methods in Communications, Security, and Computer Science*, pp. 360 – 368, Springer-Verlag, 1991.
13. Even, S., O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts", *Communications of the ACM*, vol. 28, 1985, pp. 637 – 647.
14. Fagin, R., M. Naor and P. Winkler, "Comparing Common Secret Information without Leaking it", submitted for publication, Communications of the ACM, 1994.
15. Fiat, A. and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems", In *Advances in Cryptology: Proceedings of Crypto '86*, Lecture Notes in Computer Science, Springer-Verlag, August 1986 pp. 186 – 194.
16. Goldreich, O., S. Goldwasser, and S. Micali, "How to construct random functions", *Journal of the ACM*, 133:792–807, 1986.
17. Goldreich, O., S. Micali and A. Wigderson, "How to play any mental game, or: A completeness theorem for protocols with honest majority", In *Proceedings of*

the 19th Annual ACM Symposium on Theory of Computing, May 1987, pp. 218 – 229.
18. Håstad, J., "Pseudo-random generation under uniform assumptions", In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, May 1990, pp. 395 – 440.
19. Impagliazzo, R., L. A. Levin and M. Luby, "Pseudo-random generation from one-way functions", In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, May 1989, pp. 12 – 24.
20. Kilian, J., "Founding cryptography on oblivious transfer", In *Proceedings of the 2oth Annual ACM Symposium on Theory of Computing*, May 1988, pp. 20 – 31.
21. Mac Williams, F. J. and N. J. A. Sloane, "The Theory of Error-Correcting Codes", North-Holland, 1977.
22. Rabin, M. O., "How to exchange secrets by oblivious transfer", Technical Memo TR–81, Aiken Computation Laboratory, Harvard University, 1981.
23. Spielman, D., "Linear-Time Encodable and Decodable Error-Correcting Codes", In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, , pp. 388 – 397, 1995.
24. "One Less Thing to Believe In: Fraud at Fake Cash Machine", New York Times, 13 May 1993, pp. A1 & B9.
25. Wiesner, S., "Conjugate coding", *Sigact News*, vol. 15, no. 1, 1983, pp. 78 – 88; Manuscript written *circa* 1970, unpublished until it appeared in SIGACT News.
26. 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.

This article was processed using the LaTeX macro package with LLNCS style