Oblivious Transfer based on the McEliece Assumptions

Rafael Dowsley* Jeroen van de Graaf† Jörn Müller-Quade‡
Anderson C. A. Nascimento§

March 27, 2008

Abstract

We implement one-out-of-two bit oblivious transfer (OT) based on the assumptions used in the McEliece cryptosystem: the hardness of decoding random binary linear codes, and the difficulty of distinguishing a permuted generating matrix of Goppa codes from a random matrix. To our knowledge this is the first OT reduction to these problems only.

1 Introduction

Oblivious transfer [30, 26, 11] is a primitive of central importance in modern cryptography as it implies two-party secure computation [15, 19] and multi-party computation [9]. There exist several flavors of OT, but they are all equivalent [8]. In this work, we focus on the so-called one-out-of-two oblivious transfer (OT). This is a two-party primitive where a sender (Alice) inputs two bits b_0, b_1 and a receiver (Bob) inputs a bit c called the *choice bit*. Bob receives b_c and remains ignorant about b_{1-c}, while Alice only receives a confirmation message from Bob after he completed his part of the protocol successfully. In particular, Alice cannot learn Bob's choice.

OT can be constructed based on computational assumptions, both generic such as enhanced trapdoor permutations [11, 13, 16] and specific such as factoring [26], Diffie-Hellman [3, 24, 1], Quadratic or Higher-Order Residuosity, or from the Extended Riemann Hypothesis [17].

Our result: We build OT based on the two assumptions used in the McEliece cryptosystem [22]: (1) hardness of decoding of a random linear code (known to be NP-complete [4], and known to be equivalent to the learning parity with noise (LPN) problem [27]); and (2) indistinguishability of the scrambled generating matrix of the Goppa code [21] from a random one.

Comparison to other work: To our knowledge, this is the first oblivious transfer protocol based on the McEliece assumptions only and, concurrently with [18], the first computationally secure oblivious transfer protocol not known to be broken by a quantum computer. However, for obtaining a protocol of equivalent complexity, [18] uses additional assumptions: the random oracle assumption and permuted kernels. Also,

*Department of Electrical Engineering, University of Brasilia. Campus Universitario Darcy Ribeiro,Brasilia, CEP: 70910-900, Brazil, Email:rafaeldowsley@redes.unb.br

†Laboratrio de Computao Cientfica, Universidade Federal de Minas Gerais, CEP 31270-901, Brazil. E-mail: jvdg@ufmg.br

‡Universität Karlsruhe, Institut fuer Algorithmen und Kognitive Systeme. Am Fasanengarten 5, 76128 Karlsruhe, Germany. E-mail: muellerq@ira.uka.de

§Department of Electrical Engineering, University of Brasilia. Campus Universitario Darcy Ribeiro,Brasilia, CEP: 70910-900, Brazil. E-mail: andclay@ene.unb.br

[18] needs Shamir's zero knowledge proofs [29] which are avoided in our simpler construction. Our protocol is unconditionally secure for Bob and computationally secure for Alice.

In this work, we consider only *static* adversaries, i.e., we assume that either Alice or Bob is corrupted *before* the protocol begins.

2 Preliminaries

In this section, we establish our notation and provide some facts from coding theory and formal definitions of security for oblivious transfer and bit commitment. Then, for the sake of completeness, we describe the McEliece cryptosystem and introduce the assumptions on which its security, and also the security of our protocol is based.

Henceforth, we will denote by $x \in_R D$ a uniformly random choice of element x from its domain D; and by $\oplus$ a bit-wise exclusive OR of strings. All logarithms are to the base 2.

Two sequences $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ of random variables are called *computationally indistinguishable*, denoted $X \stackrel{c}{=} Y$, if for every non-uniform probabilistic polynomial-time distinguisher D there exists a negligible function $\epsilon(\cdot)$ such that for every $n \in \mathbb{N}$,

$$|Pr[D(X_n) = 1] - Pr[D(Y_n) = 1]| < \epsilon(n)$$

2.1 Security Definition of Oblivious Transfer

Let us denote by $View_{\widetilde{A}}(\widetilde{A}(z), B(c))$ and $View_{\widetilde{B}}(A(b_0, b_1), \widetilde{B}(z))$ the *views* of dishonest Alice and Bob, respectively, which represent their inputs z, results of all local computations, and messages exchanged. Our definition of security is based on the one shown in [17] (conveniently adapted to protocols with more than two messages).

Definition 1. *A protocol $[A, B](b_0, b_1; c)$ is said to* securely implement oblivious transfer*, if at the end of its execution by the sender Alice and the receiver Bob which are modelled as probabilistic polynomial time (PPT) Turing machines having as their input a security parameter N, the following properties hold:*

- Completeness: *when the players honestly follow the protocol, Bob outputs b_c while Alice has no output.*
- Security for Alice: *For every PPT adversary $\tilde{B}$, every input z, and a (sufficiently long) random tape R_B chosen at random, there exists a choice bit c such that for $b_c \in \{0, 1\}$ the distribution (taken over Alice's randomness) of runs of $\tilde{B}(z)$ using randomness R_B with Alice having input b_c and $b_{\overline{c}} = 0$ is computationally indistinguishable from the distribution of runs with Alice having input b_c and $b_{\overline{c}} = 1$.*
- Security for Bob: *For any PPT adversary $\widetilde{A}$, any security parameter N and any input z of size polynomial in N, the view that $\widetilde{A}(z)$ obtains when Bob inputs $c = 0$ is computationally indistinguishable from that of when Bob inputs $c = 1$, denoted:*

$$View_{\widetilde{A}}(\widetilde{A}(z), B(0))|_z \stackrel{c}{=} View_{\widetilde{A}}(\widetilde{A}(z), B(1))|_z.$$

A protocol is said to be secure against honest-but-curious players, if the previous definition holds in the case Alice and Bob follow the protocol. An oblivious-transfer protocol is unconditionally secure against a player if the given properties hold even when this player is not computationally bounded.

2.2 Security Definition of String Commitment

We also need commitment schemes in our constructions. A string commitment protocol consists of two stages. In the first one, called *Commit*, the sender (Alice) provides the receiver (Bob) with evidence about her input bit-string b. Bob cannot learn it before the second stage, called *Open*, where Alice reveals her commitment to Bob, such that she cannot open a value different from b without being caught with high probability. Let us denote by $View_{\widetilde{A}}(\widetilde{A}(z), B(a))$ and $View_{\widetilde{B}}(A(b), \widetilde{B}(z))$ the *views* of dishonest Alice and Bob, respectively, which represent their inputs z, results of all local computations, and messages exchanged. Our definition is based on [23].

Definition 2. *A protocol* $[A, B](b)$ *is said to* securely implement string commitment*, if at the end of its execution by the sender Alice and the receiver Bob, which are represented as PPT Turing machines having as their input a security parameter* N*, the following properties hold:*

- Completeness: *when the players honestly follow the protocol, Bob accepts* b*.*
- Hiding: *For any PPT adversary* $\widetilde{B}$*, any security parameter* N*, any input* z *of size polynomial in* N*, and any* $k \in \mathbb{N}$*, after the Commit stage, but before the Open stage, the view of* $\widetilde{B}(z)$ *when Alice inputs* $b \in \{0,1\}^k$ *is computationally indistinguishable from the view where Alice inputs any other* $b' \in \{0,1\}^k$*,* $b' \neq b$*:*

$$View_{\widetilde{B}}(A(b), \widetilde{B}(z))|_z \stackrel{c}{=} View_{\widetilde{B}}(A(b'), \widetilde{B}(z))|_z$$

- Binding: *For any PPT adversary* $\widetilde{A}$*, any security parameter* N *and any input* z *of size polynomial in* N*, any* $k \in \mathbb{N}$*, there exists* $b \in \{0,1\}^k$ *which can be computed by Alice after the Commit stage, such that the probability that* $\widetilde{A}(b')$*,* $b' \neq b$ *is accepted by Bob in the Open stage is negligible in* N*.*

A string commitment protocol is unconditionally secure against a player if the properties in Definition 2 hold even when this player is not computationally bounded.

2.3 McEliece Cryptosystem

The folowing definition was taken from [18]. The McEliece cryptosystem [22] consists of a triplet of probabilistic algorithms $\mathrm{ME} = (\mathsf{Gen}_{\mathrm{ME}}, \mathsf{Enc}_{\mathrm{ME}}, \mathsf{Dec}_{\mathrm{ME}})$ and $M = \{0,1\}^k$.

- Key generation algorithm: The PPT key generation algorithm $\mathsf{Gen}_{\mathrm{ME}}$ works as follows:
  1. Generate a $k \times n$ generator matrix $\mathbf{G}$ of a Goppa code, where we assume that there is an efficient error-correction algorithm $\mathsf{Correct}$ which can always correct up to t errors.
  2. Generate a $k \times k$ random non-singular matrix $\mathbf{S}$.
  3. Generate a $n \times n$ random permutation matrix $\mathbf{T}$.
  4. Set $\mathbf{P} = \mathbf{SGT}$, and output $pk = (\mathbf{P}, t)$ and $sk = (\mathbf{S}, \mathbf{G}, \mathbf{T})$.
- Encryption algorithm: $\mathsf{Enc}_{\mathrm{ME}}$ takes a plaintext $m \in \{0,1\}^k$ and the public-key pk as input and outputs ciphertext $c = m\mathbf{P} \oplus e$, where $e \in \{0,1\}^n$ is a random vector of Hamming weight t.
- Encryption algorithm: $\mathsf{Dec}_{\mathrm{ME}}$ works as follows:
  1. Compute $c\mathbf{T}^{-1} (= (m\mathbf{S})\mathbf{P} \oplus e\mathbf{T}^{-1})$, where $\mathbf{T}^{-1}$ denotes the inverse matrix of $\mathbf{T}$.
  2. Compute $m\mathbf{S} = \mathsf{Correct}(c\mathbf{T}^{-1})$.
  3. Output $m = (m\mathbf{S})\mathbf{S}^{-1}$.

2.4 Security Assumptions

In this subsection, we briefly introduce and discuss the McEliece assumptions used in this work. First, we assume that there is no efficient algorithm which can distinguish the scrambled (according to the description in the previous Subsection) generating matrix of the Goppa code P and a random matrix of the same size. Currently, the best algorithm by Courtois et al. [7] works as follows: enumerate each Goppa polynomial and verify whether the corresponding code and the generator matrix $\mathbf{G}$ are "permutation equivalent" or not by using the *support splitting algorithm* [28], which is $n^t(1 + o(1))$-time algorithm, with n and t as defined in the previous subsection.

Assumption 3. *There is no PPT algorithm which can distinguish the public-key matrix P of the McEliece cryptosystem from a random matrix of the same size with non-negligible probability.*

We note that this assumption was utilized in [7] to construct a digital signature scheme.

The underlying assumption on which McEliece is the hardness of decoding random linear codes. This problem is known to be NP-complete [4], and all currently known algorithms to solve this problem are exponential. In particular, for small number of errors, the best one was presented by Canteaut and Chabaud [6].

Assumption 4. *The Syndrome Decoding Problem problem is hard for every PPT algorithm.*

2.5 String Commitment from Syndrome Decoding

We will need a bit commitment scheme based on the same assumption. Of course we could use a modification of the McEliece system which is semantical secure, see [25]. However, we can do better.

According to a well-known result by Naor [23], bit commitment scheme can be constructed using a pseudorandom generator. The latter primitive can be built efficiciently using the Syndrome Decoding problem as described by Fischer and Stern [12]. Naor's scheme is unconditionally binding, computationally hiding and meets the completeness property. So using this construction we are using only one of the McEliece assumption. In addition, for *string* commitment Naor's construction is very efficient.

3 Passively Secure Protocol for OT

For now, assume Alice and Bob to be honest-but-curious. We first sketch the intuition behind this protocol. We construct it according to the paradigm presented in [3]. Bob sends to Alice an object which is either a public key or a randomized public key for which the decoding problem is difficult. To randomize a public key, we use bitwise-XOR with a random matrix. Alice, in turn, computes the bitwise-XOR of the received entity with the same random matrix, hereby obtaining the second "key". She encrypts b_0 and b_1 with the received and computed keys, respectively, and sends the encryptions to Bob. The protocol is secure for Bob because Alice cannot distinguish a public key from a random matrix. The protocol is complete because Bob can always decrypt b_c. At the same time, it is also secure for Alice, because Bob is unable to decrypt the second bit as he cannot decode the random code.

Recall that Alice's inputs are the bits b_0 and b_1 while Bob inputs the bit c wishing to receive b_c.

Protocol 5.

1. Alice chooses a $k \times n$ random binary matrix Q and sends it to Bob.
2. Bob generates a secret key (S, G, T) following the procedures of the McEliece algorithm, sets $P_c = SGT$ and $P_{1-c} = P_c \oplus Q$ and sends P_0, t to Alice.

3. Alice computes $P_1 = P_0 \oplus Q$, then encrypts two random bit strings $r_0, r_1 \in_R \{0,1\}^k$ with P_0 and P_1, respectively, i.e., for $i = 0, 1 : y_i = r_i P_i \oplus z_i$, where $z_i \in \{0,1\}^n$, $w_H(z_i) = t$, computes for $i = 0, 1$: $m_i \in_R \{0,1\}^k$, encrypts b_0 and b_1 as follows: for $i = 0, 1 : \hat{b}_i = b_i \oplus \langle r_i, m_i \rangle$ where "$\langle \cdot, \cdot \rangle$" denotes a scalar product and finally sends for $i = 0, 1 : y_i, m_i, \hat{b}_i$ to Bob.
4. Bob decrypts r_c and computes $b_c = \hat{b}_c \oplus \langle r_c, m_c \rangle$.

The next theorem formally states the security of the above protocol.

Theorem 6. *Protocol 5 is complete and secure for both Alice and Bob against passive attacks according to Definition 1 under Assumptions 3 and 4.*

Proof. Given that under passive attacks, the players always follow the protocol, we argue the properties listed in Definition 1.
Completeness: This follows by observing that Bob always receives a valid encryption of r_c that allows him to compute b_c in Step 4.
Security for Alice: Let $\widetilde{B}$ be any PPT passively cheating receiver. Let c be the bit such that $\hat{b}_{1-c} = b_{1-c} \oplus \langle r_{1-c}, m_{1-c} \rangle$ and $y_{1-c} = r_{1-c}(P_c \oplus Q) \oplus z_{1-c}$. Note that Q is chosen randomly and independently from P_c, so from $\widetilde{B}$'s point of view, learning r_{1-c} is equivalent to decoding a random linear code with generating matrix $P_c \oplus Q$. This is known to be hard [4]. It was proven in [14] that $\langle r, m \rangle$ is a hard-core predicate for any one-way function f given $f(r)$ and m. Hence, by Assumption 4, the distribution (taken over Alice's randomness) of runs of $\tilde{B}(z)$ using randomness R with Alice having input b_c and $b_{\overline{c}} = 0$ is computationally indistinguishable from the distribution of runs with Alice having input b_c and $b_{\overline{c}} = 1$.
Security for Bob: This follows directly from Assumption 3. Honest-but-curious Alice is unable to distinguish between $P = SGT$ and a random $k \times n$ matrix, and hence she is also unable to tell $P_c = SGT$ from $P_{1-c} = SGT \oplus Q$ for any $c \in \{0,1\}$. This implies computational indistinguishability of the protocol views for Alice. □

Unfortunately, Protocol 5 is not secure if the parties cheat actively. One problem is that, given a random matrix Q, Bob can come up with two matrices P', P'', where $P' \oplus P'' = Q$, such that they are the generating matrices of the codes with some reasonably good decoding properties. It is clear that in this case, Bob will be able to partially decode *both* b_0 and b_1.

4 Fully Secure Protocol

In order to arm the passive protocol with security against malicious parties we will do the following:

1. Implement a randomized oblivious transfer in which Bob is forced to choose his the public key **before** and therefore **independent** of Q, if not he will be detected with probability at least $\frac{1}{2}$.
2. Convert the randomized oblivious transfer into an oblivious transfer for specific inputs with the same characteristics of security;
3. Reduce the probability that a malicious Bob learns *both* b_0 and b_1.

4.1 Random OT with high probability of B cheating

First, we implement a protocol that outputs two random bits a_0, a_1 to Alice and outputs a random bit d and a_d to Bob. In this protocol, Alice detects with probability at least $\frac{1}{2} - \epsilon$ a malicious Bob that chooses the public key *dependenting* of Q.

To achieve this, Bob generates two different McEliece keys by following the same procedures of protocol 5 and by using two random bits c_0, c_1. He commits to P_{0,c_0} and P_{1,c_1}. Then, Bob receives two random matrices Q_0 and Q_1 from Alice, computes

$P_{0,1-c_0} = P_{0,c_0} \oplus Q_0$ and $P_{1,1-c_1} = P_{1,c_1} \oplus Q_1$ and sends $P_{0,0}, P_{1,0}, t$ to her. Alice chooses one of the commitments for Bob to open and checks if the opened information is consistent with an honest procedure; otherwise, she stops the protocol. Finally, she encrypts a_0 and a_1 using the matrices associated to the commitment that was not opened.

Protocol 7.

1. Bob generates two McEliece secret keys (S_0, G_0, T_0) and (S_1, G_1, T_1). He chooses $c_0, c_1 \in_R \{0,1\}$ and sets $P_{0,c_0} = S_0 G_0 T_0$ and $P_{1,c_1} = S_1 G_1 T_1$. He commits to P_{0,c_0} and P_{1,c_1}.
2. Alice chooses Q_0 and Q_1 uniformly at random and sends them to Bob.
3. Bob computes $P_{0,1-c_0} = P_{0,c_0} \oplus Q_0$ and $P_{1,1-c_1} = P_{1,c_1} \oplus Q_1$. He sends $P_{0,0}, P_{1,0}, t$ to Alice.
4. Alice computes $P_{0,1} = P_{0,0} \oplus Q_0$ and $P_{1,1} = P_{1,0} \oplus Q_1$. Then she chooses the challenge $j \in_R \{0,1\}$ and sends it to Bob.
5. Bob opens his commitment to $P_{1-j,c_{1-j}}$ and sets $d = c_j$
6. Alice checks the following: $P_{1-j,c_{1-j}}$ must be equal to $P_{1-j,0}$ or $P_{1-j,1}$, otherwise she stops the protocol.
7. Alice encrypts two random bit strings $r_0, r_1 \in_R \{0,1\}^k$ with $P_{j,0}$ and $P_{j,1}$, respectively, i.e., for $i = 0,1 : y_i = r_i P_{j,i} \oplus z_i$, where $z_i \in \{0,1\}^n$, $w_H(z_i) = t$, computes for $i = 0,1$: $m_i \in_R \{0,1\}^k$, encrypts $a_0, a_1 \in_R \{0,1\}$ as follows: for $i = 0,1 : \hat{a}_i = a_i \oplus \langle r_i, m_i \rangle$ where "$\langle \cdot, \cdot \rangle$" denotes a scalar product and finally sends for $i = 0,1 : y_i, m_i, \hat{a}_i$ to Bob.
8. Bob decrypts r_d and computes $a_d = \hat{a}_d \oplus \langle r_d, m_d \rangle$. If Bob encounters a decoding error while decrypting r_d, then he outputs $a_d = 0$.

Theorem 8. *Assuming the used bit commitment scheme secure, protocol 7 implements a randomized oblivious transfer that is complete and secure for Bob against active attacks according to Definition 1 under Assumptions 3 and 4. Additionally, the probability that a malicious Bob learns both a_0 and a_1 is at most $\frac{1}{2} + \epsilon(n)$ where $\epsilon(n)$ is a negligible function.*

Proof. **Completeness:** An honest Bob always passes the test of Step 6 and receives a valid encryption of r_d, so he can compute a_d.
Security for Alice: In order to discover a_0 and a_1, Bob must learn r_0 and r_1. The encryptions of r_0 and r_1 only depend on $P_{j,0}$ and $P_{j,1}$, respectively.

If Bob sends both $P_{0,0}$ and $P_{1,0}$ chosen according to the protocol (honest procedure), then the probability that he learns both inputs of Alice is the same as in the passive protocol, i.e., it is negligible. If Bob chooses in a malicious way both $P_{0,0}$ and $P_{1,0}$, then with overwhelming probability Alice will stop the protocol in step 6 and Bob will learn neither r_0 nor r_1.

The best strategy for Bob is to choose honestly one of the matrices and choose the other in a malicious way, thus he can cheat and partially decode *both* r_0 and r_1 in case Alice asks him to open the matrix correctly chosen. However, note that with probability $\frac{1}{2}$, Alice asks him to open the matrix maliciously chosen. In this case, Bob will be able to open the commitment with the value that Alice expects in step 6 only with negligible probability. Thus, the probability that a malicious Bob learns both a_0 and a_1 is at most $\frac{1}{2} + \epsilon(n)$ where $\epsilon(n)$ is a negligible function.
Security for Bob: The commitment to $P_{j,c_j} = P_{j,d}$ is not opened, so the security for Bob follows from Assumption 3 as in the protocol 5.

As long as the commitment is secure, possible differences from the passive scenario are the following ones:

- Alice could cheat by sending a specially chosen matrix Q, however by Assumption 3, she cannot tell P_{j,c_j} from random, hence her choice of Q will not affect her ability to learn d;
- For some $i \in \{0,1\}$, Alice may use a different matrix instead of $P_{j,i}$ for encrypting r_i in Step 7 hoping that $i = d$ so that Bob will encounter the decoding error and then complain, hereby disclosing his choice. However, the last instruction of Step 8 thwarts such attack by forcing Bob to accept with a fixed output "0". Sending a "wrong" syndrome is then equivalent to the situation when Alice sets his input $a_i = 0$.

Thus, it follows that the protocol is secure against Alice. □

4.2 Derandomizing the previous protocol

Subsequently, we use the method of [2] to transform the randomized oblivious transfer into an (ordinary) oblivious transfer with the same characteristics of security.

Protocol 9.

1. Bob and Alice execute the protocol 7. Alice receives a_0, a_1 and Bob receives d, a_d.
2. Bob chooses c, sets $e = c \oplus d$ and sends e to Alice.
3. Alices chooses $b_0, b_1 \in \{0,1\}$, computes $f_0 = b_0 \oplus a_e$ and $f_1 = b_1 \oplus a_{1\oplus e}$ and sends f_0, f_1 to Bob.
4. Bob computes $b_c = f_c \oplus a_d$.

Theorem 10. *Protocol 9 implements an oblivious transfer with the same characteristics of security of the protocol 7.*

Proof. **Completeness:** $f_c = b_c \oplus a_{c\oplus e} = b_c \oplus a_d$, so an honest Bob can recover b_c because he knows a_d.
Security for Alice: $f_{1\oplus c} = b_{1\oplus c} \oplus a_{1\oplus c\oplus e} = b_{1\oplus c} \oplus a_{1\oplus d}$, so Bob can recover both b_0 and b_1 only if he knows a_0 and a_1.
Security for Bob: Alice has to discover d in order to compute c, thus the security for Bob follows from the protocol 7. □

4.3 Reducing the probability of B cheating

Finally, we use the reduction of [10] to minimize the probability that a malicious Bob learns both inputs of Alice. In this reduction, protocol 9 is executed s times in parallel, where s is a security parameter. The inputs in each execution are chosen in such way that Bob must learn both bits in all executions to be able to compute both inputs of Alice in protocol 11.

Protocol 11.

1. Alice chooses $b_0, b_1 \in \{0,1\}$ and $b_{0,1}, b_{0,2}, \ldots, b_{0,s}, b_{1,1}, b_{1,2}, \ldots, b_{1,s} \in_R \{0,1\}$ such that $b_0 = b_{0,1} \oplus b_{0,2} \oplus \ldots \oplus b_{0,s}$ and $b_1 = b_{1,1} \oplus b_{1,2} \oplus \ldots \oplus b_{1,s}$.
2. Bob chooses $c \in \{0,1\}$.
3. Protocol 9 is executed s times, with inputs $b_{0,i}, b_{1,i}$ from Alice and $c_i = c$ from Bob for $i = 1 \ldots s$.
4. Bob computes $b_c = b_{c,1} \oplus b_{c,2} \oplus \ldots \oplus b_{c,s}$.

Theorem 12. *Assuming that the bit commitment scheme used in protocol 7 is secure, protocol 11 is complete and secure for both Alice and Bob against active attacks according to Definition 1 under Assumptions 3 and 4.*

Proof. **Completeness:** An honest Bob learns all $b_{c,i}$ for $i = 1 \dots s$ in the s executions of protocol 9 and therefore he can compute b_c.
Security for Alice: Bob must discover both bits in all executions of protocol 9 in order to learn something simultaneously on b_0 and b_1. The probability that a malicious Bob learns both bits in an execution of protocol 9 is at most $\frac{1}{2} + \epsilon(n)$, where $\epsilon(n)$ is a negligible function. There exists an n_0 such that $\epsilon(n) < \frac{1}{4}$ for any $n > n_0$. We can choose $n > n_0$, so $\beta = \frac{1}{2} + \epsilon(n) < \frac{3}{4}$ and the probability that a malicious Bob learns both b_0 and b_1 is less than $(\frac{3}{4})^s$, which is negligible in s. Thus, the protocol is secure for Alice.
Security for Bob: Alice discovers c if she learns any c_i, but this probability is negligible because the probability that she learns a specific c_i in the respective execution of the protocol 9 is negligible and the number of executions of the protocol 9 is polynomial. □

Acknowledgements The authors acknowledge previous discussions with Kirill Morozov and Hideki Imai.

References

[1] W. Aiello, Y. Ishai, O. Reingold: Priced Oblivious Transfer: How to Sell Digital Goods. In EUROCRYPT'01, pp. 119–135, 2001.

[2] D. Beaver: Precomputing Oblivious Transfer. CRYPTO 1995: 97-109.

[3] M. Bellare, S. Micali: Non-Interactive Oblivious Transfer and Applications, CRYPTO'89, LNCS 435, pp. 547-557, 1990.

[4] E.R. Berlekamp, R.J. McEliece, H.C.A van Tilborg, "On the Inherent Intractability of Certain Coding Problems," IEEE Trans. Inf. Theory, vol. 24, pp.384–386, 1978.

[5] A. Blum, M. Furst, M. Kearns and R. J. Lipton, "Cryptographic primitives based on hard learning problems," Proc. CRYPTO '93, LNCS 773, pp. 278–291, 1994.

[6] A. Canteaut, F. Chabaud "A new algorithm for finding minimum-weight words in a linear code: application to primitive narrow-sense BCH codes of length 511," IEEE Trans. Inf. Theory, vol. 44(1), pp.367–378, 1998.

[7] N. Courtois, M. Finiasz, N. Sendrier: How to Achieve a McEliece Digital Signature Scheme. In: Asiacrypt'2001, LNCS 2248, pp. 157–174, 2001.

[8] C. Crépeau, "Equivalence between two flavors of oblivious transfers," *Proc. CRYPTO '87*, LNCS, vol. 293, pp. 350–354, 1988.

[9] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp: "Committed Oblivious Transfer and Private Multi-Party Computations," *Proc. CRYPTO '95*, pp. 110.

[10] Ivan Damgård, Joe Kilian, Louis Salvail: On the (Im)possibility of Basing Oblivious Transfer and Bit Commitment on Weakened Security Assumptions. *EUROCRYPT 1999*: 56-73.

[11] Even, S., Goldreich, O., and Lempel, A.: A randomized protocol for signing contracts. In: Proceedings CRYPTO '82. Plenum Press (1983) 205–210.

[12] Jean-Bernard Fischer, Jacques Stern: An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding. *EUROCRYPT 1996*: 245-255.

[13] O. Goldreich, "Foundations of Cryptography - Volume 2 (Basic Applications)," Cambridge University Press, 2004.

[14] O. Goldreich, L. A. Levin. Hard-Core Predicates for Any One-Way Function. In *21st ACM Symposium on the Theory of Computing*, pages 25-32, 1989.

[15] Goldreich, O., Micali, S., Wigderson, A.: How to Play Any Mental Game, or: A completeness theorem for protocols with honest majority. In: Proc. 19th ACM STOC. ACM Press (1987) 218–229.

[16] I. Haitner: implementing Oblivious Transfer Using Collection of Dense Trapdoor Permutations. In: TCC'04, LNCS 2951, pp. 394–409, 2004.

[17] Y. Kalai: Smooth Projective Hashing and Two-Message Oblivious Transfer. In EUROCRYPT'05, LNCS 3494, pp. 78–95, 2005.

[18] K. Kobara, K. Morozov, R. Overbeck, Oblivious Transfer via McEliece's PKC and Permuted Kernels, Cryptology ePrint Archive 2007/382.

[19] Kilian, J.: Founding Cryptography on Oblivious Transfer. In: 20th ACM STOC. ACM Press (1988) 20–31.

[20] K. Kobara and H. Imai: Semantically Secure McEliece Cryptosystems – Conversions for McEliece PKC. In: PKC 2001, LNCS 1992, pp. 19–35, 2001.

[21] R.J. McEliece, "The Theory of Information and Coding (Vol. 3 of The Encyclopedia of Mathematics and Its Applications.), Reading, Mass., Addison-Wesley, 1977.

[22] R.J. McEliece: A Public-Key Cryptosystem Based on Algebraic Coding Theory. In Deep Space Network progress Report, 1978.

[23] Naor, M.: Bit Commitment using Pseudo-Randomness. In: Advances in Cryptology–CRYPTO '89. LNCS, vol. 435. Springer-Verlag (1990) 128–136.

[24] M. Naor and B. Pinkas: Efficient Oblivious Transfer Protocols, SODA'01 (SIAM Symposium on Discrete Algorithms).

[25] R. Nojima, H. Imai, K. Kobara, K. Morozov, "Semantic Security for the McEliece Cryptosystem without Random Oracles," Accepted to WCC '07, Versailles, France, April 2007.

[26] Rabin, M.O.: How to Exchange Secrets by Oblivious Transfer. Technical Memo TR-81, Aiken Computation Laboratory, Harvard University (1981).

[27] O. Regev, "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography", Proc. 37th STOC, pp. 84-93, 2005.

[28] N. Sendrier, "Finding the Permutation Between Equivalent Linear Codes: The Support Splitting Algorithm," IEEE Trans. Inf. Theory, 46(4), pp.1193–1203, 2000.

[29] A. Shamir. An efficient identification scheme based on permuted kernels. In Proc. of Crypto89, volume 435 of LNCS, pages 606609. Springer Verlag, 1990.

[30] S. Wiesner, "Conjugate coding," *Sigact News*, vol. 15, no. 1, 1983, pp. 78–88; original manuscript written circa 1970.