Oblivious Transfer Based on the McEliece Assumptions

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

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. We present two different constructions for oblivious transfer, one based on cut-and-choose arguments and another one which is based on a novel generalization of Bennett-Rudich commitments which may be of independent interest. Finally, we also present a variant of our protocol which is based on the Niederreiter cryptosystem.[1]

Keywords: Oblivious Transfer, McEliece cryptosystem, Post-quantum Security.

1 Introduction

Oblivious transfer [35, 31, 14] is a primitive of central importance in modern cryptography as it implies two-party secure computation [20, 23] 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 bit oblivious transfer (OT). This is a two-party primitive in which 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 $b_{\overline{c}}$, 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.

[*]The author is with the Computer Science and Engineering Department, University of California at San Diego (UCSD), 9500 Gilman Drive, La Jolla, California 92093, USA. This work was done while the author was with the Department of Electrical Engineering, University of Brasilia. Email: rdowsley@cs.ucsd.edu.

[†]The author is with he Department of Computer Science, Federal University of Ouro Preto. Instituto de Ciências Exatas e Biológicas, Universidade Federal de Ouro Preto. Ouro Preto, Minas Gerais, CEP: 35400-000, Brazil. Email: jvdg@iceb.ufop.br.

[‡]The author is with the Universität Karlsruhe, Institut für Algorithmen und Kognitive Systeme. Am Fasanengarten 5, 76128 Karlsruhe, Germany. Email:muellerq@ira.uka.de.

[§]The author is with the Department of Electrical Engineering, University of Brasilia Campus Darcy Ribeiro, 70910-900, Brasilia, DF, Brazil. Email: andclay@ene.unb.br.

[1]A preliminary version of this work appeared at ICITS 2008 [12], this is the full version.

OT can be constructed based on computational assumptions, both generic such as enhanced trapdoor permutations [14, 18, 21] and specific such as factoring [31], Diffie-Hellman [3, 28, 1], Quadratic or Higher-Order Residuosity, or from the Extended Riemann Hypothesis [22].
Our result: We build OT based on the two assumptions used in the McEliece cryptosystem [26]:

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 [32])

2. Indistinguishability of the scrambled generating matrix of the Goppa code [25] from a random one.

It is noteworthy that there exists no black box reduction from public-key encryption to OT [17]. However, by exploiting some algebraic properties of ciphertexts generated by the McEliece cryptosystem we bypass the negative results of [17]. We present two different constructions (with similar complexities): one based on cut-and choose arguments and another one based on a generalization of Bennett-Rudich commitments to integers modulo q. Finally, we also present an OT protocol based on the Niederreiter cryptosystem [29] (the dual of the McEliece cryptosystem).
Comparison to other work: To our knowledge, this is the first oblivious transfer protocol based on the McEliece assumptions only and, concurrently with [24], the first computationally secure oblivious transfer protocol not known to be broken by a quantum computer. However, for obtaining a protocol of equivalent complexity, [24] uses additional assumptions: the random oracle assumption and permuted kernels. Also, [24] needs Shamir's zero knowledge proofs [34] which are avoided in our simpler construction.

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 and Niederreiter cryptosystems and introduce the assumptions on which their 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.

Two sequences $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ of random variables are called *computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, 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 the sender Alice and the receiver Bob be modeled as probabilistic polynomial time (PPT) Turing machines A and B, having as their input a security parameter N.

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 [22], suitably 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 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:*
$$View_{\widetilde{A}}(\widetilde{A}(z), B(0))|_z \overset{c}{\approx} View_{\widetilde{A}}(\widetilde{A}(z), B(1))|_z.$$

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.

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. Similarly to the notation used for Oblivious Transfer, we 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 [27].

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 $b' \in \{0,1\}^k$, $b' \neq b$:*

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

- 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 following definition was taken from [30]. The McEliece cryptosystem [26][2] consists of a triplet of probabilistic algorithms McE $= (\mathsf{Gen}_{\mathrm{McE}}, \mathsf{Enc}_{\mathrm{McE}}, \mathsf{Dec}_{\mathrm{McE}})$ and $M = \{0,1\}^k$.

- Key generation algorithm: The PPT key generation algorithm $\mathsf{Gen}_{\mathrm{McE}}$ works as follows:
  1. Generate a $k \times n$ generator matrix 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 S.
  3. Generate a $n \times n$ random permutation matrix T.
  4. Set $P = SGT$, and output $pk = (P, t)$ and $sk = (S, G, T)$.
- Encryption algorithm: $\mathsf{Enc}_{\mathrm{McE}}$ takes a plaintext $m \in \{0,1\}^k$ and the public-key pk as input and outputs ciphertext $c = mP \oplus e$, where $e \in \{0,1\}^n$ is a random vector of Hamming weight t.
- Decryption algorithm: $\mathsf{Dec}_{\mathrm{McE}}$ works as follows:
  1. Compute $cT^{-1}(= (mS)G \oplus eT^{-1})$, where T^{-1} denotes the inverse matrix of T.
  2. Compute $mS = \mathsf{Correct}(cT^{-1})$.
  3. Output $m = (mS)S^{-1}$.

[2]There are variants of this scheme that achieve the strongest notions of security for encryption schemes: IND-CPA [30] and IND-CCA2 [13].

2.4 Niederreiter Cryptosystem

The Niederreiter cryptosystem [29] consists of a triplet of probabilistic algorithms $\mathrm{NR} = (\mathsf{Gen}_{\mathrm{NR}}, \mathsf{Enc}_{\mathrm{NR}}, \mathsf{Dec}_{\mathrm{NR}})$.

- System parameters: $n, t \in \mathbb{N}$, where $t \ll n$.
- Key Generation Algorithm: The polynomial-time key generation algorithm $\mathsf{Gen}_{\mathrm{NR}}$ works as follows:
  1. Generate a $(n-k) \times n$ parity check matrix of a binary irreducible Goppa $\mathcal{G}$ code of maximal dimension k which can efficiently correct up to t errors.
  2. Generate a $(n-k) \times (n-k)$ random binary non-singular matrix M.
  3. Generate a $n \times n$ random permutation matrix P.
  4. Set $H' = MHP$, and output $pk = (H', t)$ and $sk = (M, P, D_{\mathcal{G}})$, where $D_{\mathcal{G}}$ is an efficient syndrome decoding algorithm for $\mathcal{G}$.
- Encryption algorithm: The polynomial-time encryption algorithm $\mathsf{Enc}_{\mathrm{NR}}$ takes a plaintext $m \in \{0,1\}^n$ of weight t and the public-key pk as input and outputs the syndrome $s = H'm^T$.
- Decryption algorithm: The polynomial-time decryption algorithm $\mathsf{Dec}_{\mathrm{NR}}$ takes as input an ciphertext s, the secret key sk and works as follows:
  1. Compute $M^{-1}s = HPm^T$.
  2. Compute $D_{\mathcal{G}}(HPm^T)$ to obtain Pm^T.
  3. Compute $m^T = P^{-1}Pm^T$ and output m.

2.5 Security Assumptions

In this subsection, we briefly introduce and discuss the McEliece assumptions used in this work.

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. For codes with very high rate $R = k/n$ there is an efficient distinguisher that can differentiate a Goppa code from a random code [15], but this is not the range in which the encryption scheme works. For the other cases, the currently best algorithm is by Courtois et al. [7] and works as follows: enumerate each Goppa polynomial and verify whether the corresponding code and the generator matrix G are "permutation equivalent" or not by using the *support splitting algorithm* [33], which is $n^t(1 + o(1))$-time algorithm, with n and t as defined in the previous subsection.

Assumption 1 *There exists 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.

We also assume that there is no efficient algorithm for solving the Syndrome Decoding Problem. This problem is known to be NP-complete [4], and all currently known algorithms to solve this problem are exponential. The best algorithms were presented by Canteaut and Chabaud [6] and recently by Bernstein et al. [5].

Assumption 2 *The Syndrome Decoding Problem is hard for every PPT algorithm.*

The security of the McEliece cryptosystem and the Niederreiter cryptosystem are equivalent [11] (the difficulty of breaking the Niederreiter cryptosystem is the same of breaking the McEliece cryptosystem with the same security parameters).

2.6 Bit Commitment based on the McEliece assumptions

We will also 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 [30]. However, we can do better.

According to a well-known result by Naor [27], bit commitment scheme can be constructed using a pseudorandom generator. The latter primitive can be built efficiently using the Syndrome Decoding problem as described by Fischer and Stern [16]. Naor's scheme is unconditionally binding, computationally hiding and meets the completeness property. So using this construction to obtain bit commitment 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. Denote the Hamming weight of a vector z by $w_H(z)$.

Protocol 1

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 $P_{\overline{c}} = P_c \oplus Q$ 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 $i = 0, 1 : y_i = r_i P_i \oplus e_i$, where $e_i \in \{0,1\}^n$, $w_H(e_i) = t$. Then she also computes for $i = 0, 1 {:} h_i \in_R \{0,1\}^k$, encrypts b_0 and b_1 as follows: for $i = 0, 1 : \hat{b}_i = b_i \oplus \langle r_i, h_i \rangle$ where "$\langle \cdot, \cdot \rangle$" denotes a scalar product modulo 2. Finally, she sends for $i = 0, 1 : y_i, h_i, \hat{b}_i$ to Bob.
4. Bob decrypts r_c and computes $b_c = \hat{b}_c \oplus \langle r_c, h_c \rangle$.

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

Theorem 1 *Protocol 1 is complete and secure for both Alice and Bob against passive attacks according to Definition 1 under Assumptions 1 and 2.*

Given that under passive attacks the players always follow the protocol, we argue that the properties listed in Definition 1 are satisfied.
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 $\hat{b}_{\overline{c}} = b_{\overline{c}} \oplus \langle r_{\overline{c}}, h_{\overline{c}} \rangle$ and $y_{\overline{c}} = r_{\overline{c}}(P_c \oplus Q) \oplus e_{\overline{c}}$. Note that Q is chosen randomly and independently from P_c, so from $\widetilde{B}$'s point of view, learning $r_{\overline{c}}$ 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 [19] that $\langle r, h \rangle$ is a hard-core predicate for any one-way function f given $f(r)$ and h. Hence, by Assumption 2, 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 1. 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 $P_{\overline{c}} = SGT \oplus Q$ for any $c \in \{0,1\}$. This implies computational indistinguishability of the protocol views for Alice.

Unfortunately, Protocol 1 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 $P' \oplus P'' = Q$, which 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, one could use a general *compiler* as the one in [18]. However, we present a direct and more efficient approach:

1. Implement a randomized oblivious transfer in which Bob is forced to choose 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 simultaneously information on *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 *depending* on Q.

To achieve this, Bob generates two different McEliece keys by following the same procedures of protocol 1 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 $P_{0,\overline{c_0}} = P_{0,c_0} \oplus Q_0$ and $P_{1,\overline{c_1}} = P_{1,c_1} \oplus Q_1$ 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 2

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 $P_{0,\overline{c_0}} = P_{0,c_0} \oplus Q_0$ and $P_{1,\overline{c_1}} = P_{1,c_1} \oplus Q_1$. 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 $P_{\bar{j},c_{\bar{j}}}$ and sets $d = c_j$.

6. Alice checks the following: $P_{\bar{j},c_{\bar{j}}}$ must be equal to $P_{\bar{j},0}$ or $P_{\bar{j},1}$, 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 $i = 0, 1 : y_i = r_i P_{j,i} \oplus e_i$, where $e_i \in \{0, 1\}^n$, $w_H(e_i) = t$. Then she also computes for $i = 0, 1$: $h_i \in_R \{0, 1\}^k$, encrypts $a_0, a_1 \in_R \{0, 1\}$ as follows: for $i = 0, 1 : \hat{a}_i = a_i \oplus \langle r_i, h_i \rangle$ where "$\langle \cdot, \cdot \rangle$" denotes a scalar product. Finally, she sends for $i = 0, 1 : y_i, h_i, \hat{a}_i$ to Bob.

8. Bob decrypts r_d and computes $a_d = \hat{a}_d \oplus \langle r_d, h_d \rangle$. If Bob encounters a decoding error while decrypting r_d, then he outputs $a_d = 0$.

Theorem 2 *Assuming that the bit commitment scheme used is secure, protocol 2 implements a randomized oblivious transfer that is complete and secure for Bob against active attacks according to Definition 1 under Assumptions 1 and 2. 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.*

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 obtain simultaneously information on 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 both $P_{0,0}$ and $P_{1,0}$ in a malicious way, 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 pairs of 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 pair of matrices 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 1 as in the protocol 1.

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 1, 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 $P_{j,i}$ to encrypt 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 3

1. Bob and Alice execute the protocol 2. 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. Alice chooses $b_0, b_1 \in \{0, 1\}$, computes $f_0 = b_0 \oplus a_e$ and $f_1 = b_1 \oplus a_{\overline{e}}$ and sends f_0, f_1 to Bob.

4. Bob computes $b_c = f_c \oplus a_d$.

Theorem 3 *Protocol 3 implements an oblivious transfer with the same characteristics of security of the protocol 2.*

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: $f_{\overline{c}} = b_{\overline{c}} \oplus a_{\overline{c\oplus e}} = b_{\overline{c}} \oplus a_{\overline{d}}$, 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 protocol 2.

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 3 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 4.

Protocol 4

1. Alice chooses her inputs bits to the oblivious transfer protocol, $b_0, b_1 \in \{0, 1\}$. She also chooses random bits $b_{0,1}, \ldots, b_{0,s}, b_{1,1}, \ldots, b_{1,s}$ such that $b_0 = b_{0,1} \oplus \ldots \oplus b_{0,s}$ and $b_1 = b_{1,1} \oplus \ldots \oplus b_{1,s}$.

2. Bob chooses $c \in \{0, 1\}$.

3. Protocol 3 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 4 *Assuming that the bit commitment scheme used in protocol 2 is secure, protocol 4 is complete and secure for both Alice and Bob against active attacks according to Definition 1 under Assumptions 1 and 2.*

Completeness: An honest Bob learns all $b_{c,i}$ for $i = 1 \ldots s$ in the s executions of protocol 3 and therefore he can compute b_c.
Security for Alice: Bob must discover both bits in all executions of protocol 3 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 3 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 3 is negligible and the number of executions of the protocol 3 is polynomial.

5 OT using BCX (Bit Commitments with XOR)

5.1 Protocol using BCX against malicious parties

Instead of using protocols 2, 3 and 4 to deal with malicious adversaries, we can use a modification of the BCX (Bit Commitments with XOR) protocol and obtain another OT construction. The obtained construction's complexity is similar to our previous protocol. However, we believe that our generalization of BCX might be of independent interest and thus present it here.

The BCX protocol is attributed to Bennett and Rudich and is described in [9]. The scheme is based on any secure commitment protocol and runs as follows: Alice commits to a bit b by committing to pairs of bits b_{jL} and b_{jR} such that $b = b_{jL} \oplus b_{jR}$. The BCX advantage when compared to a traditional commitment is that it enables to prove linear relations among commitments without revealing them. We modify the BCX slightly to work with integers modulo q, as described below (where v is a security parameter of the procedure to prove relations):

- Commitment: In order to commit to some value $b \in \mathbb{Z}_q$, Bob chooses $b_{jL} \in_R \mathbb{Z}_q$ and sets $b_{jR} = b - b_{jL} (\text{mod } q)$ for $j = 1, \ldots, v$. Bob commits to b_{jL} and b_{jR} for $j = 1, \ldots, v$.

- Unveil: In order to unveil the value b, Bob unveils all the $2v$ commitments. Alice checks if $b = b_{jL} + b_{jR} (\text{mod } q)$ for all j and accepts the value b only if this condition is satisfied.

- Proving relations: In order to prove a linear relation among u committed values $b^1, \ldots, b^u$ without revealing them, the parties do the following:

  1. Alice specifies u permutations (of v elements each).
  2. For $l = 1, \ldots, u$, Bob shuffles the v pairs of commitments related to b^l by using the permutation specified by Alice.
  3. For $j = 1, \ldots, v$, Bob evaluates the linear relation using the values $(b^1_{jL}, \ldots, b^u_{jL})$ and using the values $(b^1_{jR}, \ldots, b^u_{jR})$ and denotes the results as Rel_{jL} and Rel_{jR} respectively.
  4. Bob sends Rel_{jL} and Rel_{jR} to Alice for $j = 1, \ldots, v$.
  5. Alice checks whether the sum $Rel_{jL} + Rel_{jR}$ is the same for all j.

6. For $j = 1, \dots, v$, Alice asks Bob to open either $(b^1_{jL}, \dots, b^u_{jL})$ or $(b^1_{jR}, \dots, b^u_{jR})$ and checks if the result sent previously by Bob is the correct evaluation of the relation on these values. If the values are not equal, Alice stops the protocol execution. Otherwise, she accepts that the result of applying the linear relation on values $(b^1, \dots, b^u)$ is $Rel_{1L} + Rel_{1R} (\mathrm{mod}\ q)$.

- Coping the Commitment: Each commitment can be used only once to prove relations. If Bob is committed to some value b, he can obtain a new copy of this commitment as follow:

  1. Bob creates $3v$ pairs of commitment such that the sum modulo q of each pair is b.
  2. Alice partitions these $3v$ pairs in 3 subsets of cardinality v that she denoted as b^0, b^1, b^2. She asks Bob to prove that $b^0 = b$. If he succeeds, b and b^0 cannot be used more, but Alice is convinced that $b^1 = b^2 = b$. So Bob have two valid instances of commitment to the value b: b^1 and b^2.

Note that a cheating Bob trying to prove a false relation between committed values is caught with overwhelming probability in the security parameter v. Note also that only one commitment of each pair is unveiled in the procedure to prove relations, so the values of $b^1, \dots, b^u$ are not unveiled because they are the sum of the committed values of the same pair.

Using the above ideas, we can build an OT protocol based on McEliece cryptosystem that is secure against malicious adversaries. We describe the protocol below. We can interpret each matrix of dimension $k \times n$ as a binary string of length nk and use the above variant of Bennett-Rudich commitments computing the operations modulo $q = 2^{nk}$.

Protocol 5

1. Bob generates a secret key (S, G, T) following the procedures of the McEliece algorithm and commits to the string representation of the public key matrix SGT using Bennett-Rudich XOR Commitment. Denote the commitment as $d = (d_{1L}, d_{1R}, \dots, d_{vL}, d_{vR})$ and define $d_j = (d_{jL}, d_{jR})$.

2. Alice chooses a $k \times n$ random binary matrix Q and sends it to Bob.

3. Bob sets $P_c = SGT$ and $P_{\bar{c}} = P_c \oplus Q$ and commits to the string representation of the matrices P_0 and P_1. Denote the commitments as $f = (f_{1L}, f_{1R}, \dots, f_{vL}, f_{vR})$ and $g = (g_{1L}, g_{1R}, \dots, g_{vL}, g_{vR})$ respectively. Bob sends P_0, t to Alice. For $j = 1, \dots, v$ he also sends the pairs $f_j = (f_{jL}, f_{jR})$ and $g_j = (g_{jL}, g_{jR})$ to Alice, choosing randomly if f_j or g_j goes first.

4. Alice computes $P_1 = P_0 \oplus Q$, chooses randomly a challenge V of v bits and a permutation Π on v elements. Alice sends V and Π to Bob.

5. Denote as V_j the j-bit of V. For $j = 1, \dots, v$, Bob does as follows:

If $V_j = 0$, then Bob opens the commitments f_j and g_j. Alice checks if one of them is the string representation of matrix P_0 and the other the string representation of matrix P_1, otherwise she stops the protocol execution.

If $V_j = 1$, then Bob proves using the Bennett-Rudich Commitment scheme approach (i.e., sending the left and right sums and revealing either the left or the right commitments according to Alice choices) that $d_{\Pi(j)} = f_j$ or that $d_{\Pi(j)} = g_j$. If Bob fails to prove this, Alice stops the protocol execution.

6. Alice encrypts two random bit strings $r_0, r_1 \in_R \{0,1\}^k$ with P_0 and P_1, respectively, i.e., for $i = 0, 1 : y_i = r_i P_i \oplus e_i$, where $e_i \in \{0,1\}^n$, $w_H(e_i) = t$. Then she also computes for $i = 0, 1 : h_i \in_R \{0,1\}^k$, encrypts b_0 and b_1 as follows: for $i = 0, 1 :$ $\hat{b}_i = b_i \oplus \langle r_i, h_i \rangle$ where "$\langle \cdot, \cdot \rangle$" denotes a scalar product modulo 2. Finally, she sends for $i = 0, 1 : y_i, h_i, \hat{b}_i$ to Bob.

7. Bob decrypts r_c and computes $b_c = \hat{b}_c \oplus \langle r_c, h_c \rangle$. If Bob encounters a decoding error while decrypting r_c, then he outputs $b_c = 0$.

Theorem 5 *Assuming the used bit commitment scheme secure, protocol 5 implements an oblivious transfer that is complete and secure for Alice and Bob against active attacks according to Definition 1 under Assumptions 1 and 2.*

Completeness: An honest Bob always passes the test of Step 5 and receives a valid encryption of r_c, so he can compute b_c.

Security for Alice: In order to obtain simultaneously information on b_0 and b_1, Bob must learn r_0 and r_1. The encryptions of r_0 and r_1 only depend on P_0 and P_1, respectively.

If Bob follows the protocol instructions to choose P_0 and P_1, the security to Alice follows from the passive case. If a cheating Bob does not choose P_0 and P_1 according to the protocol procedures, he succeeds in the test of Step 5 only with negligible probability in the security parameter v. The reason for that is the following. If Alice chooses $V_j = 0$ in the challenge, she will learn f_j and g_j and so she may check if they are equal to P_0 and P_1. Therefore Alice can check that the values Bob committed to in step 3 are the values corresponding to P_0 and P_1 that she will use to send the bits. If Alice chooses $V_j = 1$ in the challenge, she learns that $d_{\Pi(j)} = f_j$ or $d_{\Pi(j)} = g_j$ and learns also what of the two cases happened. Therefore Alice has a way to verify that one of the matrices committed to in step 3 is equal to SGT that was chosen before Q. This put together with the previous fact guarantees that one of the public keys used by Alice is random from Bob's point of view.

Security for Bob: If Alice chooses $V_j = 0$ in the challenge, she will learn f_j and g_j and so she may check if they are equal to P_0 and P_1. But since f_j and g_j are sent in a random order, she learns nothing about c. If Alice chooses $V_j = 1$ in the challenge, she learns that $d_{\Pi(j)} = f_j$ or $d_{\Pi(j)} = g_j$ and learns also what of the two cases happened. But since f_j and g_j are sent in a random order and are not unveiled, she cannot compare the one that is equal to $d_{\Pi(j)}$ to the matrices P_0 and P_1 in order to discover c. So the security for Bob follows from Assumption 1 as in the protocol 1.

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 1, she cannot tell P_c from random, hence her choice of Q will not affect her ability to learn c;
- For some $i \in \{0,1\}$, Alice may use a different matrix instead of P_i for encrypting r_i in Step 6 hoping that $i = c$ so that Bob will encounter the decoding error and then complain, hereby disclosing his choice. However, the last instruction of Step 7 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 $b_i = 0$.

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

6 OT based on Niederreiter cryptosystem

Below we describe a variant of the OT protocol using the Niederreiter cryptosystem. The passive protocol can be implemented as follows.

Protocol 6

1. Alice chooses a $(n-k) \times n$ random binary matrix Q and sends it to Bob.
2. Bob generates a secret key $(M, P, D_{\mathcal{G}})$ following the procedures of the Niederreiter algorithm, sets $P_c = MHP$ and $P_{\overline{c}} = P_c \oplus Q$ and sends P_0, t to Alice.
3. Alice computes $P_1 = P_0 \oplus Q$, then encrypts two random bit strings $r_0, r_1 \in_R \{0,1\}^n$ of weight t with P_0 and P_1, respectively, i.e., for $i = 0,1 : s_i = P_i r_i^T$. Then she also chooses for $i = 0,1$: $h_i \in_R \{0,1\}^n$, encrypts b_0 and b_1 as follows: for $i = 0,1 : \hat{b}_i = b_i \oplus \langle r_i, h_i \rangle$ where "$\langle \cdot, \cdot \rangle$" denotes a scalar product modulo 2. Finally, she sends for $i = 0,1 : s_i, h_i, \hat{b}_i$ to Bob.
4. Bob decrypts r_c and computes $b_c = \hat{b}_c \oplus \langle r_c, h_c \rangle$.

Theorem 6 *Protocol 6 is complete and secure for both Alice and Bob against passive attacks according to Definition 1 under Assumptions 1 and 2.*

Given that under passive attacks, the players always follow the protocol, we argue the properties listed in Definition 1.

Completeness: As in the McEliece protocol, Bob can always decrypt r_c and so 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 $\hat{b}_{\overline{c}} = b_{\overline{c}} \oplus \langle r_{\overline{c}}, h_{\overline{c}} \rangle$ and $s_{\overline{c}} = (P_c \oplus Q) r_{\overline{c}}^T$. Note that Q is chosen randomly and independently from P_c, so from $\widetilde{B}$'s point of view, learning $r_{\overline{c}}$ is equivalent to decoding a random linear code with parity check matrix $P_c \oplus Q$. By the same arguments as in McEliece version, $\langle r, h \rangle$ is a hard-core predicate for this function and the security for Alice follows from assumption 2 .

Security for Bob: This follows directly from Assumption 1 and the equivalence of McEliece's and Niederreiter's cryptosystems security [11]. Honest-but-curious Alice is unable to distinguish between $P = MHP$ and a random $(n-k) \times n$ matrix, and hence she is also unable to tell $P_c = MHP$ from $P_{\bar{c}} = MHP \oplus Q$ for any $c \in \{0,1\}$. This implies computational indistinguishability of the protocol views for Alice.

In order to achieve security against malicious parties, we can use the same approach used in the protocol version that utilizes the McEliece's cryptosystem.

7 Conclusions

In this paper we presented the first protocol for implementing oblivious transfer based solely on the McEliece assumption. We also presented the first oblivious transfer protocol based on the Niederreiter cryptosystem. Our protocols are secure in the so-called half-simulation paradigm. We state as an open problem to obtain fully simulatable oblivious transfer protocols based on the McEliece assumptions.

Acknowledgments 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. *EUROCRYPT 2001*, pp. 119–135.

[2] D. Beaver. Precomputing Oblivious Transfer. *CRYPTO 1995*, pp. 97–109.

[3] M. Bellare, S. Micali. Non-Interactive Oblivious Transfer and Applications. *CRYPTO 1989*, pp. 547–557.

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

[5] D. J. Bernstein and T. Lange and C. Peters. Attacking and Defending the McEliece Cryptosystem. *PQCrypto 2008*, pp. 31–46.

[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 Transactions on Information Theory*, vol. 44(1), pp.367–378, 1998.

[7] N. Courtois, M. Finiasz, N. Sendrier. How to Achieve a McEliece Digital Signature Scheme. *ASIACRYPT 2001*, pp. 157–174.

[8] C. Crépeau. Equivalence Between Two Flavors of Oblivious Transfers. *CRYPTO 1987*, pp. 350–354.

[9] C. Crépeau, J. van de Graaf, and A. Tapp. Committed Oblivious Transfer and Private Multi-Party Computations. *CRYPTO 1995*, pp. 110–123.

[10] I. Damgård, J. Kilian, L. Salvail. On the (Im)possibility of Basing Oblivious Transfer and Bit Commitment on Weakened Security Assumptions. *EUROCRYPT 1999*, pp. 56–73.

[11] R. Deng, Y. Li, X. Wang. The Equivalence of McEliece's and Niederreiter's Public-key Cryptosystems. *IEEE Transactions on Information Theory*, Vol. 40, pp. 271–273, 1994.

[12] R. Dowsley, J. van de Graaf, J. Müller-Quade, A. C. A. Nascimento. Oblivious Transfer Based on the McEliece Assumptions. *ICITS 2008*, pp. 107–117.

[13] R. Dowsley, J. Müller-Quade, A. C. A. Nascimento. A CCA2 Secure Public Key Encryption Scheme Based on the McEliece Assumptions in the Standard Model. *CT-RSA 2009*. pp. 240–251.

[14] S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *CRYPTO 1982*, pp. 205–210.

[15] J. Faugère, V. Gauthier, A. Otmani, L. Perret, J. Tillich. A Distinguisher for High Rate McEliece Cryptosystems. Cryptology ePrint Archive, Report 2010/331.

[16] J. Fischer, J. Stern. An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding. *EUROCRYPT 1996*, pp. 245–255.

[17] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, M. Viswanathan. The Relationship between Public Key Encryption and Oblivious Transfer. *FOCS 2000*, pp. 325–335.

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

[19] O. Goldreich, L. A. Levin. Hard-Core Predicates for Any One-Way Function. *21st ACM STOC*, pp. 25–32, 1989.

[20] O. Goldreich, S. Micali, A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. *19th ACM STOC*, pp. 218–229, 1987.

[21] I. Haitner. Implementing Oblivious Transfer Using Collection of Dense Trapdoor Permutations. *TCC 2004*, pp. 394–409.

[22] Y. Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. *EUROCRYPT 2005*, pp. 78–95.

[23] J. Kilian. Founding Cryptography on Oblivious Transfer. *20th ACM STOC*, pp. 20–31, 1988.

[24] K. Kobara, K. Morozov, R. Overbeck. Oblivious Transfer via McEliece's PKC and Permuted Kernels. *MMICS 2008*, pp. 142–156.

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

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

[27] M. Naor. Bit Commitment using Pseudo-Randomness. *CRYPTO 1989*, pp. 128–136.

[28] M. Naor, B. Pinkas. Efficient Oblivious Transfer Protocols, *SODA 2001*.

[29] H. Niederreiter. Knapsack-type Cryptosystems and Algebraic Coding Theory. *Prob. of Control and Inf. Theory*, vol. 15(2), pp. 159–166, 1986.

[30] R. Nojima, H. Imai, K. Kobara, K. Morozov. Semantic Security for the McEliece Cryptosystem without Random Oracles. *WCC 2007*.

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

[32] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *37th STOC* pp. 84–93, 2005.

[33] N. Sendrier. Finding the Permutation Between Equivalent Linear Codes: The Support Splitting Algorithm. *IEEE Transactions on Information Theory*, 46(4), pp. 1193–1203, 2000.

[34] A. Shamir. An Efficient Identification Scheme based on Permuted Kernels. *CRYPTO 1989*, pp. 606–609.

[35] S. Wiesner. Conjugate coding. *Sigact News*, vol. 15, no. 1, pp. 78–88. 1983.