Oblivious Transfer via McEliece's PKC and Permuted Kernels

K. Kobara[1], Kirill Morozov[1] and R. Overbeck[2]*

[1] RCIS, AIST, Japan
{k-kobara,kirill.morozov}@aist.go.jp

[2] EPFL - I&C - ISC - LASEC
Station 14 - Building INF
CH-1015 Lausanne
Switzerland
overbeck@cdc.informatik.tu-darmstadt.de

Abstract. We present two efficient protocols for two flavors of oblivious transfer (OT): the Rabin and 1-out-of-2 OT using the McEliece cryptosystem and Shamir's zero-knowledge identification scheme based on permuted kernels. This is a step towards diversifying computational assumptions on which OT – the primitive of central importance – can be based.

Although we obtain a weak version of Rabin OT (where the malicious receiver may decrease his erasure probability), it can nevertheless be reduced to secure 1-out-of-2 OT.

Elaborating on the first protocol, we provide a practical construction for 1-out-of-2 OT.

Keywords: Oblivious transfer, coding-based cryptography, McEliece cryptosystem, permuted kernel problem.

1 Introduction

Oblivious transfer (OT) [21, 8, 27] is a cryptographic primitive of central importance since it implies secure two-party computation [10, 14]. Sending the data by OT ensures its partial erasure in such a way that the sender cannot learn what exactly was lost, he can only enforce some loss to occur.

A number of complexity assumptions can be used to construct OT: generic, e.g., enhanced trapdoor permutations [8, 9, 12], and specific, e.g., factoring [21], Diffie-Hellman [3, 18, 1], N'th or Quadratic Residuosity and Extended Riemann Hypothesis [13].

* Part of this work was done at the Cryptography and Computer Algebra Group, TU-Darmstadt, Germany. Part of this work was funded by the DFG.

At the same time, until recently, no construction has been devised using coding-based assumptions. We focus on those related to security of the McEliece public key cryptosystem [16]: hardness of bounded distance decoding and pseudorandomness of the generating matrix of a Goppa code (we refer to these two as *the McEliece assumption*). They are particularly interesting due to their "post-quantum" nature – currently, they do not have polynomial quantum solutions.

Related Work Currently, the list of McEliece-based cryptographic primitives include a digital signature [6] and identity-based identification and signature schemes [5].

We are also aware of the parallel work [19] on the topic of 1-out-of-2 oblivious transfer based on McEliece assumption. Their solution is different from ours and their work is currently in progress. In 1-out-of-2 (string) OT, the sender transmits two bits (or bit strings) such that only one of them gets through according to the receiver's choice. The sender is sure, however, that one of the bits (strings) is erased.

Our contribution We provide practical constructions for two flavors of OT based on the McEliece assumption and the Permuted Kernel Problem (PKP),[3] in the second one, we also employ the random oracle model. Our contribution goes in three ways:
1) In the Rabin OT, the input string is either erased with probability 1/2 or gets through unchanged, while the sender does not learn which event has occurred. We present a construction for the Rabin oblivious transfer of strings based on the assumption that the McEliece PKC is secure.

In more details, for transmitting a bit-string $\mathbf{m}$, the sender encrypts it using the McEliece public-key matrix constructed in the following way. Take a Goppa code correcting up to t errors and concatenate it with a random matrix, then randomly scramble and permute the resulting matrix as in the ordinary McEliece encryption. Now, if the error vector used for the encryption has the Hamming weight $\gg t$, the two cases are possible. Consider the part of the ciphertext corresponding to the Goppa code. Either up to t errors occur there, in which case $\mathbf{m}$ can be efficiently restored, or more than t occur, so that $\mathbf{m}$ cannot be decoded or, in other words, it is virtually erased. We show that for appropriate parameters the erasure probability of $\mathbf{m}$ is exactly 1/2 when the players follow the protocol. Note that the honest-but-curious sender cannot learn which of

[3] It was introduced to cryptography in Shamir's identification scheme [24].

the two cases takes place as he does not know the permutation made at the encryption stage.

Cheating behavior can be avoided by having the players prove their correct behavior in zero-knowledge (ZK). For this, we employ the ZK proofs from the Shamir's identification scheme based on permuted kernels [24] and the Stern's identification scheme based on syndrome decoding [25].

Note that we provide a new application for the above schemes which can be of independent interest in the coding-based protocols. Indeed, the Stern's scheme can be used for proving that the McEliece encryption is performed with an error vector of the correct weight. The Shamir's scheme can be employed to convince the verifier who possess a code G that the prover knows a permutation on a given subcode of G without revealing the permutation itself.

Unfortunately, the malicious receiver may attempt the general decoding, if he obtained erasure. We show that even in this case, his erasure probability will still be large enough given the proper choice of parameters and imposing a reasonable assumption on hardness of general decoding. We will call this weaker version a *gap-Rabin OT*. Even with this difference between the erasure probability for a honest receiver and that for a malicious receiver, this protocol remains meaningful:

2) Rabin OT and 1-out-of-2 OT are polynomially equivalent as it was shown by Crépeau in [7]. We use a slightly modified version of this reduction to build an L-out-of-M OT (without a gap for probabilities) from the gap-Rabin OT. This construction highlights an interesting theoretical property of the Crépeau's reduction which has not been thoroughly studied before – its tolerance to imperfectness of the Rabin OT.

The communication cost of this protocol is 143 KBytes for honest-but-curious and 129 MBytes for malicious parties and the workload of the best attack for honest-but-curious parties is estimated as 2^{80} (henceforth, we do not provide the best attack workload by the fully malicious parties as these attacks may depend on the security of underlying primitives – hash functions and pseudorandom generators – which we assume ideal). The above costs are provided for the case when the players are aided by the (partially) trusted third party (TTP). They can be substantially improved if the same techniques are used for building 1-out-of-2 OT directly:

3) We present a protocol for 1-out-of-2 String OT against fully malicious parties without relying on TTP. For the minimal parameter set, the com-

munication cost is about 13.3 MBytes and the workload of the best attack by honest-but-curious parties is 2^{80}.

At the core of our construction is the paradigm, where the receiver provides the sender with two matrices, a valid McEliece public key G^{pub} and a random matrix Q in the order corresponding to the choice bit c. The sender uses those to encrypt his inputs and sends the encryptions. The malicious sender cannot tell G^{pub} from Q, while the malicious receiver cannot decrypt both strings.

As the players do not trust each other, none is allowed to choose Q – they generate it by coin flipping. Furthermore, the receiver must prove a correct choice of G^{pub} and Q. We employ the PKP-based scheme by Shamir [24] in order to accomplish this. That requires the following modification to the protocol. The receiver discloses the generating matrix G of the Goppa code, corresponding to G^{pub}. Then, he provides the sender with the randomly permuted subcodes of G and Q, respectively, which now play the role of the keys. The ZK proof is used to assure the sender that the subcodes were indeed taken from G and Q. Employing the subcodes prevents the sender from telling the keys apart.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the security model and the main ingredients for our constructions. Our gap-Rabin OT protocol and its reduction to L-out-of-M OT is presented in Section 3. Section 4 introduces our direct construction for 1-out-of-2 OT. We provide concluding remarks and open questions in Section 5.

2 Preliminaries

Let us state what we want to achieve. In our definitions, the players are represented by probabilistic polynomial time (PPT) Turing machines having the security parameter N as their common input. The security requirements must hold for any large enough N and any input which is polynomial in N.

For vectors, the summation is elementwise in the corresponding field.

2.1 Security Requirements and Assumptions

Rabin oblivious transfer is the trusted erasure channel with fixed probability $1 - \mathcal{P}$ where the input is a bit-vector $\mathbf{b} \in \mathbb{F}_2^k$. The malicious sender $\widetilde{\mathsf{Sen}}$ has no knowledge on the output, while the malicious receiver $\widetilde{\mathsf{Rec}}$ cannot learn the erased input.

The definition of γ-gap-Rabin OT is analogous to the above, but $\widetilde{\mathsf{Rec}}$ can decrease the erasure probability from his point of view by γ. This probability is denoted by $\mathcal{Q} = 1 - \mathcal{P} - \gamma$.

In the other flavor of oblivious transfer, 1-out-of-2 String OT, the sender Sen inputs two bit-vectors $\mathbf{b}_0, \mathbf{b}_1 \in \mathbb{F}_2^a$. The receiver Rec obtains one of them according to his choice $c \in \{0, 1\}$. $\widetilde{\mathsf{Sen}}$ is unable to learn c, while $\widetilde{\mathsf{Rec}}$ remains ignorant about $\mathbf{b}_{1-c}$. See Appendix A for more formal security definitions of the above primitives.

The security of all schemes we present in this paper is based on the assumption, that the following problems are hard to solve in the average case [20, 4]:

Definition 2.1. *In the following let all matrices and vectors be over* $\mathbb{F}_q$.

(i) *Given a $k \times n$ matrix, decide if its row-space is within a Goppa code or was generated at random. (Goppa-code-distinguishing Problem)*

(ii) *Given a (random) $[n, k]$ code generated by the matrix G^{pub}, a word $\mathbf{c}$ and an integer w, find $\mathbf{e}$ of Hamming weight at most w such that $\mathbf{c} = \mathbf{m}\mathsf{G}^{\text{pub}} + \mathbf{e}$ for some $\mathbf{m}$. (General Syndrome Decoding)*

(iii) *Given a (random) $[n, k, d]$ code generated by the matrix G^{pub}, a word $\mathbf{c}$ and an integer $w \leq (d-1)/2$, find $\mathbf{e}$ of Hamming weight at most w such that $\mathbf{c} = \mathbf{m}\mathsf{G}^{\text{pub}} + \mathbf{e}$ for some $\mathbf{c}$. (Bounded Distance Decoding)*

(iv) *Given a random $[n, k]$ code and a random permuted subcode of dimension $l < k$, find the permutation. (Permuted Kernel Problem)*

Only Problems *(ii)* and *(iv)* are known to be $\mathcal{NP}$-hard in the general case [24, 25].

2.2 Tools

We shortly recall the principal protocols our OT scheme will be based on: the McEliece PKC and the zero-knowledge identification protocols (ZKIP) by Stern and Shamir connected to coding theory. Further, we give an intuition as for how to build a L-out-of-M OT from Rabin OT.

McEliece's asymmetric encryption scheme [16] works as follows: Upon input of the system parameters m, t, the key generation algorithm outputs the secret key consisting of three matrices: $(\mathsf{S}, \mathsf{G}, \mathsf{P})$, where $\mathsf{G} \in \mathbb{F}_2^{k \times n}$ is a canonical generator matrix of an $[n, k \geq n - mt, 2t + 1]$ binary irreducible Goppa code, $\mathsf{S} \in \mathbb{F}_2^{k \times k}$ is non-singular and $\mathsf{P} \in \mathbb{F}_2^{n \times n}$ is a permutation matrix. The corresponding public key is $(\mathsf{G}^{\text{pub}} = \mathsf{SGP}, t)$. To encrypt a message $\mathbf{m} \in \mathbb{F}_2^k$ the sender chooses a random binary vector

$\mathbf{e}$ of length n and Hamming weight t and computes the ciphertext $\mathbf{c} = \mathbf{m}\mathsf{G}^{\mathrm{pub}} + \mathbf{e}$. The secret key holder now can recover $\mathbf{m}$ from $\mathbf{c}$ using his secret key.

For properly chosen parameters, the McEliece PKC is secure [4] and there exist conversions to obtain CCA2 security [15]. For such variants, or if only random strings are encrypted, $\mathsf{G}^{\mathrm{pub}}$ can be chosen to be *systematic* (i.e. with the k-dimensional identity matrix Id_k in the first k columns), as we will do in the following. This reduces space needed to store $\mathsf{G}^{\mathrm{pub}}$.

The size of the ciphertexts can be reduced to $n - k$ if the message is represented by $\mathbf{e}$. This is known as the Niederreiter PKC, compare [22]. In the latter case (e.g. if a hash of $\mathbf{e}$ serves as a random seed or key for a symmetric encryption scheme) it is sufficient to send the syndrome $\mathbf{e}(\mathsf{G}^{\mathrm{pub}})^{\perp}$ as ciphertext, where $(\mathsf{G}^{\mathrm{pub}})^{\perp}$ refers to the systematic check matrix of $\mathsf{G}^{\mathrm{pub}}$.

Stern's ZKIP [25] has a check matrix $\mathsf{H} \in \mathbb{F}_q^{n\times(n-k)}$ and an integer w as system parameters. An user's identity $\mathbf{s}$ is computed from the user's secret, a vector $\mathbf{e} \in \mathbb{F}_q^n$ of Hamming weight w: $\mathbf{s} = \mathbf{e}\mathsf{H}$. By Stern's 3-round zero-knowledge protocol, the secret key holder can prove his knowledge of $\mathbf{e}$ using two blending factors: a permutation and a random vector. However, a dishonest prover not knowing $\mathbf{e}$ can cheat the verifier in the protocol with probability $2/3$. Thus, the protocol has to be run several times to detect cheating provers. Computing $\mathbf{e}$ from $\mathbf{s}$ is solving Problem *(ii)* from Definition 2.1. The communication cost is about $n(1 + \log_2(n))\log_2(q)$ plus three times the size of the employed commitments (e.g. a hash function).

Shamir's Permuted Kernel ZKIP [24] works quite similarly, i.e. it has a check matrix $\mathsf{H} \in \mathbb{F}_q^{n\times(n-k)}$ and an integer l as system parameters. (Shamir proposed to use $l = 1$ and q to be a large prime. However, taking q small and $l < (n - k)$ works as well [26].) The user's identity $\mathsf{K} \in \mathbb{F}_q^{l\times n}$ is computed from the user's secret, a permutation $\mathsf{\Pi}$ as follows: K is taken at random from the right kernel of $\mathsf{\Pi}\mathsf{H}$. In the following we can view K as an n-vector over $\mathbb{F}_{q^l}$. By Shamir's 5-round zero-knowledge protocol, the secret key holder can prove his knowledge of $\mathsf{\Pi}$ using two blending factors: a permutation and a random n-vector over $\mathbb{F}_{q^l}$. However, a dishonest prover not knowing $\mathsf{\Pi}$ can cheat the verifier in the protocol with probability $(q^l + 1)/(2 \cdot q^l)$. Thus, the protocol has to be repeated several times to detect cheating provers. Computing $\mathsf{\Pi}$ from K is is solving Problem *(iv)* from Definition 2.1. The communication cost is about $n(l + \log_2(n))\log_2(q)$ plus two times the size of the commitments.

Crépeau's protocol [7] allows to build an 1-out-of-2 from a Rabin OT and a hash function h: In a first stage, N random messages r_i are sent to the receiver by a Rabin OT with erasure probability $\mathcal{Q}$. Now K is chosen such that $K < \mathcal{P}N = (1-\mathcal{Q})N < 2K < N$, i.e. the receiver obtains at least K and at most $2K-1$ of the random messages r_i. Then, the receiver sends two disjoint sets $\mathcal{I}, \mathcal{J} \subseteq \{1, \cdots N\}$ of K indices to the sender, such that one of the sets contains only indices of not erased messages r_i. For the 1-out-of-2 OT, the messages m_0, m_1 are encrypted as $\mathbf{c}_0 = \mathbf{m}_0 + h((r_i)_{i\in\mathcal{I}})$ and $\mathbf{c}_1 = \mathbf{m}_1 + h((r_j)_{j\in\mathcal{J}})$. Since the receiver knows either the set $(r_i)_{i\in\mathcal{I}}$ or $(r_j)_{j\in\mathcal{J}}$, he obtains exactly one of the messages from $\mathbf{c}_0$ and $\mathbf{c}_1$. Crépeau's protocol fails with some probability which is negligible for large N and can easily be computed. Similarly one obtains an L-out-of-M protocol.

3 Rabin Oblivious Transfer

Our scheme implementing Rabin OT with erasure probability $1-\mathcal{P}$ consists of two phases: An initialization used for key generation, where a Goppa code is concatenated with a random code and used as substitute for the secret code in the McEliece PKC, see Algorithm 3.1. To ensure correct generation of the public key, we use a trusted third party (TTP) in Algorithm 3.1, which can be omitted as we will see later on. In the transmission phase, see Algorithm 3.2, en- and decryption work like in

Algorithm 3.1 Key generation

Input: Security parameters $m, t, t', l \in \mathbb{N}$.

Receiver	**TTP**
Set $n = 2^m$, $k = 2^m - mt$. Generate a McEliece PKC key pair with security parameters m, t. Let $(\mathsf{S}, \mathsf{G}, \mathsf{P})$ be the secret key with public key $(\mathsf{G}^{\mathrm{pub}} = \mathsf{SGP}, t)$. Send $\mathsf{G}^{\mathrm{pub}}$ to the TTP.	Generate a random matrix $\mathsf{G}' \in \mathbb{F}_2^{k\times l}$ Generate an $(n+l)\times(n+l)$ random permutation matrix P'
	Publish the systematic matrix $\mathsf{O}^{\mathrm{pub}}$ generating the same $[n+l,k]$ code as $\left[\,\mathsf{G}^{\mathrm{pub}} \middle| \mathsf{G}'\,\right]\mathsf{P}'$. Send P' and G' to the receiver.
Verify that $\mathsf{O}^{\mathrm{pub}}$ was generated correctly.	

the McEliece PKC. The difference lies in the modified public key, which ensures, that the receiver cannot decrypt all valid ciphertexts. The time complexity for Algorithm 3.2 is $\mathcal{O}(n \cdot k + n \cdot m \cdot t^2)$ operations [6]. The size of the ciphertexts is $n + l$, but as mentioned in Section 2, this can be reduced to $n + l - k$ by encoding the message into the error vector $\mathbf{e}$.

Algorithm 3.2 Transmission

```
Input: The security parameters $m, l, t'$ and a $k$-bit message $\mathbf{m}$.
Output: An OT ciphertext $\mathbf{c}$.

Encryption:
Obtain the receiver's public key $\mathsf{O}^{\mathrm{pub}}$.
Generate a random vector $\mathbf{e}$ of weight $t'$ and length $2^m + l$.
Compute the ciphertext $\mathbf{c} = \mathbf{m}\mathsf{O}^{\mathrm{pub}} + \mathbf{e}$.
Send $\mathbf{c}$ to the receiver.

Decryption:
Set $(\mathbf{c}_1, \mathbf{c}_2) = \mathbf{c}(\mathsf{P}')^{-1}$, where $\mathbf{c}_1$ is an $n$-bit vector.
Try to apply the error correction algorithm for $\mathsf{G}$ to $\mathbf{c}_1\mathsf{P}^{-1}$ in order to obtain $\mathbf{m}$.
if (previous step fails) or ($\mathbf{m}\mathsf{O}^{\mathrm{pub}} + \mathbf{c}$ has weight $\neq t'$) then
  return erasure.
else
  return $\mathbf{m}$.
```

3.1 Security Analysis

For an outside attacker, the presented OT scheme offers as much resistance as the McEliece PKC with the same parameters, i.e. with a public $k \times (n + l)$ matrix and an error vector of weight t'. Since we consider the McEliece PKC to resist cryptanalysis for appropriate parameters, there exist parameters, such that the presented OT scheme resists cryptanalytic attacks [4] of outside attackers. However, more important things are the security issues regarding the parties involved in our protocol.

Correctness. Observe, that if parameters are chosen carefully and every party follows the protocol, Algorithm 3.2 works correctly: Let us assume that $l = n$ and $t' = 2t+1$. Let $(\mathbf{e}_1, \mathbf{e}_2) = \mathbf{e}(\mathsf{P}')^{-1} = \mathbf{c}(\mathsf{P}')^{-1} + \mathbf{m}\left[\mathsf{G}^{\mathrm{pub}} \middle| \mathsf{G}'\right]$, where $\mathbf{e}_1$ is an n-bit vector. Then, iff $\mathbf{e}_1$ has weight $\leq t$, the decryption procedure returns the correct message $\mathbf{m}$. Else erasure occurs or the receiver obtains a false message $\mathbf{m'} \neq \mathbf{m}$. However, the latter case is unlikely to occur, since then, the weight of $\mathbf{m}'\mathsf{O}^{\mathrm{pub}} + \mathbf{c}$ is t'. Thus, $\mathbf{m}'\mathsf{O}^{\mathrm{pub}} + \mathbf{c} + \mathbf{m}\mathsf{O}^{\mathrm{pub}} + \mathbf{c}$ has weight $\leq 2t' = 4t + 2$. Thus, one has found a codeword of weight below the Gilbert-Varshamov (GV) bound in $\mathsf{O}^{\mathrm{pub}}$, which is infeasible even if such a codeword exists.

Since every choice of t' below half of the GV-bound of $\mathsf{O}^{\mathrm{pub}}$ leads to a correct scheme, parameters may be chosen, such that the probability $\mathcal{P}$ of obtaining the message $\mathbf{m}$ varies. We can compute $\mathcal{P}$ as the fraction of error vectors with no more than t entries on the positions of $\mathsf{G}^{\mathrm{pub}}$:

$$\mathcal{P} := \sum_{i=0}^{t} \frac{\binom{n}{i}\binom{l}{t'-i}}{\binom{n+l}{t'}} = 1 - \underbrace{\sum_{i=t+1}^{t'} \frac{\binom{n}{i}\binom{l}{t'-i}}{\binom{n+l}{t'}}}_{=:\mathcal{Q}^{\mathrm{H}}} .$$

Thus, if $n = l$ and $t' = 2t + 1$, the scheme works correctly and we have $\mathcal{P} = \mathcal{Q}^{\mathrm{H}} = 1/2$.

Dishonest receiver. In this scheme, a dishonest receiver has the possibility to raise its probability to receive the message $\mathbf{m}$. He might choose to guess a part of the error vector or try to apply a general decoding algorithm to the erroneous word $\mathbf{c}_2$ of the code G'. However, the dishonest receiver can obtain the message with probability 1 only if general decoding is easy:

The probability $\mathcal{Q}_A$ of an erasure for a dishonest receiver spending A operations on decryption can be computed as follows:

$$\mathcal{Q}_A = \sum_{i=t_0}^{t_1} \frac{\binom{n}{i}\binom{l}{t'-i}}{\binom{n+l}{t'}},$$

if the following conditions are true (we denote $\mathbf{e}(\mathsf{P}')^{-1} = (\mathbf{e}_1, \mathbf{e}_2)$):

(i) Solving the general decoding problem for $\mathbf{c}$ and $\mathsf{O}^{\mathrm{pub}}$ takes more than A operations. Note that A can be computed taking into account the (best known) attack by Canteaut and Chabaud [4] using the lower bound from [22]:

$$A \geq 2^{-t \log_2(1-k/n)}. \tag{1}$$

(ii) The general decoding problem for $\mathbf{c}_2 = \mathbf{m}\mathsf{G}' + \mathbf{e}_2$ and G' cannot be solved in A operations if $\mathbf{e}_2$ has weight $\geq t' - t_1$.

(iii) If the weight w of $\mathbf{e}_1 = \mathbf{c}_1 + \mathbf{m}\mathsf{G}^{\mathrm{pub}}$ is larger than t_0, the receiver cannot guess a sufficiently large subset of the support of $\mathbf{e}_1$ to apply the decoding algorithm for Goppa codes. This is

$$\frac{\binom{n}{w-t}}{\binom{w}{w-t}} m^3 t^2 \geq A, \tag{2}$$

since each decoding attempt takes $m^3 t^2$ operations [6] and there are $\binom{w}{w-t}$ correct guesses.

To our knowledge there exist neither codes with better error correction ratio than binary irreducible Goppa codes nor efficient list decoding algorithm for binary irreducible Goppa codes [11]. Thus, if $\mathrm{wt}(\mathbf{e}_1) > t_0$, the receiver either has to guess part of the error or is forced to use a general decoding algorithm.

There is a non-negligible gap between $\mathcal{P} = 1 - \mathcal{Q}^{\mathrm{H}}$ and $\mathcal{Q}_{2^{80}}$, i.e. $1 - \mathcal{P} - \mathcal{Q}_{2^{80}} > 0$ is not decreasing exponentially fast in m, t. We will call our Rabin OT an γ-gap Rabin OT with $\gamma = (1 - \mathcal{P} - \mathcal{Q}_{2^{80}})$. We can reach reasonable values for $\mathcal{P}$ and $\mathcal{Q}_{2^{80}}$, compare Table 3.1. In fact, one can even use the dishonest receiver's strategy in a positive way, i.e., to raise the chance of obtaining the message. The work factor for decryption is then given by equation (2). This is useful for protocols like in [7], where we need to ensure that the receiver gets at least half of the messages, compare Section 2. Note, that we do not need $\mathcal{P} = 1 - \mathcal{Q}_{2^{80}}$ in Crépeau's protocol but only $K \leq \mathcal{P}N \leq (1 - \mathcal{Q}_{2^{80}})N < 2K$ for some $K < N/2$. By Crépeau's protocol we can thus conclude:

Theorem 3.1. *γ-gap Rabin OT with erasure probability $\mathcal{Q}^{\mathrm{H}} > 0$ is polynomial time equivalent to $(L = \lfloor M\mathcal{P} \rfloor)$-out-of-$M$ OT if $(\mathcal{P} + \gamma) < \frac{L+1}{M}$.*

The next corollary follows from the above theorem with L=1, M=2 and the straightforward reduction from 1-out-of-2 OT to Rabin OT.

Corollary 3.1. *A γ-gap Rabin OT with $\gamma < \mathcal{Q}^{\mathrm{H}} < 1/2$ is polynomial time equivalent to Rabin OT.*

Parameters				Size Public Key	Size Ciphertext	Decryption		$\mathcal{Q}_A$	
m	t	t'	l	$= k(n+l-k)$	$= n+l-k$	runtime	$\mathcal{Q}^{\mathrm{H}}$	$A = 2^{38}$	$A = 2^{80}$
12	200	$2t+1$	2^{12}	1,377 KBytes	812 Bytes	2^{26}	0.5	0.41	0.11
13	402	$2t+13$	2^{13}	4,974 KBytes	1,677 Bytes	$2^{28.5}$	0.66	0.61	0.36
14	800	$2t+1$	2^{14}	17,874 KBytes	3,448 Bytes	2^{31}	0.5	0.46	0.29

Table 3.1. Parameter sets for the Rabin OT

Example 1. With the first parameter set from Table 3.1 and a receiver spending up to 2^{35} operations on each decryption, we can obtain a 1-out-of-2 OT by Crépeau's construction, which fails with probability less than 2^{-30} if we choose $N = 2K = 180$. This results in a communication cost of about 143KB.

Dishonest sender. Since there is no algorithm known, which allows to distinguish a Goppa code from a random code, a malicious sender cannot identify the columns of O^{pub} belonging to G^{pub}. In consequence, honest but curious sender cannot learn if a message is going to be erased or not. Nevertheless, a malicious sender could try to send a random string or to modify t'. However, the receiver can force the sender to prove that the ciphertext $\mathbf{c}$ is a valid one:

Let $\mathsf{H} = (\mathsf{O}^{\text{pub}})^{\perp}$ be systematic and $w = t'$. We will view Stern's ZKIP with security parameters H, w and the user identity $\mathbf{s} = \mathbf{c}\mathsf{H}$ [25] (compare Section 2). Now, if the encryption was made correctly, the error vector $\mathbf{e} = \mathbf{m}\mathsf{O}^{\text{pub}} + \mathbf{c}$ can be used by the sender as secret key. Thus, the honest sender can employ Stern's ZKIP to convince the receiver by a zero-knowledge proof that $\mathbf{c}$ is indeed a valid ciphertext, while the dishonest sender will be revealed.

Example 1. (continued) If we want to prove the correctness of every sent message with a security level of 2^{30} by Stern's ZKIP, this leads to a communication cost of 129 Megabytes for the 1-out-of-2 OT.

Reaction attack A third security issue is the reaction of the sender in the case he gets feedback if an erasure occurred or not. Suppose that an attacker learns that the receiver cannot decode a certain ciphertext. Then, the sender can choose to modify the corresponding error vector only slightly for the next encryption. By this and some statistics the sender could thus identify the columns of G^{pub} in O^{pub}, which breaks the scheme. Nevertheless, the sender should be cautious, as the receiver might detect such manipulations by comparing ciphertexts.

This might get important as feedback may well come from the higher level protocols (like Crépeau's protocol), for which oblivious transfer is used as a primitive. However, there are plenty of possible countermeasures against an attack by feedback, compare, e.g., [15].

3.2 Omitting a trusted third party

In a more advanced scheme, correct key generation can be verified by the sender if we follow the ideas of Shamir's PKP ZKIP [24] (compare Section 2). The basic idea is, that the sender provides the pseudorandom part G' of the private key and checks the correct construction of the public key by the PKP ZKIP. Our proposal is summarized in Algorithm B.1, see Appendix B.

Due to space limitations, we do not elaborate on the application of Shamir's ZKIP here, but rather present a new protocol where it allows us to provide security against fully malicious players with reasonable efficiency.

4 1-out-of-2 String Oblivious Transfer

Notation and Tools. For encryption, we employ IND-CCA2 McEliece PKC [15] where the message length is a bits, a is defined by the particular conversion (see [15]). As the McEliece private key the players will use a generating matrix G of the irreducible Goppa $[n, k, d]$-code with $n = 2^m$, $k = n - mt$, $d = 2t + 1$ (correcting up to t errors). By a slight abuse of notation, a code and its generating matrix will be referred to using the same letter.

In our protocol, we use two well-known cryptographic tools: pseudorandom generators (PRG) and collision-resistant hash functions, both are described in a variety of literature (e.g., see [17] and the references therein). The particular implementation is not essential in our case. We denote by $\text{PRG}(\mathbf{s}) = (\mathsf{G}, \mathsf{Q})$ the process of choosing the $[n, k, 2t + 1]$ Goppa code G and pseudorandom generation of the matrix $\mathsf{Q} \in \mathbb{F}^{k \times n}$ using a random seed $\mathbf{s}$. A hash function is denoted by $h(\cdot)$.

For proving in zero-knowledge that a permuted subcode belongs to a given code, we employ the Shamir's ZKIP [24] essentially in the same way as described in Subsection B.

For generation of the random entities, the parties use their local randomness unless otherwise stated.

Our protocol is presented in Algorithm 4.1 (see Appendix C.1 for intuition behind its construction).

4.1 Security Analysis

Correctness. Assume that the players behave honestly. Let us briefly argue that the receiver always obtains the correct $\mathbf{b}_c$. Both players will

Algorithm 4.1 1-out-of-2 String OT

```
Sender: input: $\mathbf{b}_0, \mathbf{b}_1 \in \mathbb{F}_2^a$; output: none
Receiver: input: $c \in \{0,1\}$; output: $b_c$

Sender

2. Save the received values as $\mathbf{w}_h$. Generate a random seed $\mathbf{v}$. Send $h(\mathbf{v})$.
4. Reject if $h(\mathbf{w}) \neq \mathbf{w}_h$, otherwise
   set $\mathbf{s} = \mathbf{v} + \mathbf{w}$, $\mathrm{PRG}(\mathbf{s}) = (\mathsf{G}, \mathsf{Q})$. Send $\mathbf{v}$.
6. Reject if the proof fails, otherwise compute $\mathsf{G}_0, \mathsf{G}_1$ such that
   $[\mathsf{G}_0|\mathsf{G}_1] = \left[\mathsf{Id}^{k'} \middle| \mathrm{NS}([\mathsf{C}_0|\mathsf{C}_1])\right]$.
   For $i = 0, 1$: Encrypt $\mathbf{b}_i$ using $\mathsf{G}_i$ as the public key and send the encryption.

Receiver

1. Generate a random seed $\mathbf{w}$. Send $h(\mathbf{w})$.
3. Save the received value as $\mathbf{v}_h$. Send $\mathbf{w}$.
5. Reject if $h(\mathbf{v}) \neq \mathbf{v}_h$, otherwise:
   – Set $\mathbf{s} = \mathbf{v} + \mathbf{w}$ and $\mathrm{PRG}(\mathbf{s}) = (\mathsf{G}, \mathsf{Q})$.
   – Generate a random permutation matrix $\mathsf{P}' \in \mathbb{F}_2^{n \times n}$ and a random matrix of rank $k'$: $\mathsf{S}' \in \mathbb{F}_2^{k' \times k}$.
   – Set $\mathsf{C}_c = \mathsf{S}'\mathsf{G}\mathsf{P}'$, $\mathsf{C}_{1-c} = \mathsf{S}'\mathsf{Q}\mathsf{P}'$.
   – Send $NS([\mathsf{C}_0|\mathsf{C}_1])$
   – Prove in ZK that $[\mathsf{C}_0|\mathsf{C}_1]$ is a permuted subcode of $[\mathsf{G}|\mathsf{Q}]$.
7. Decrypt and output $\mathbf{b}_c$.
```

not reject in Steps 4 and 5. Moreover, the sender will always accept the ZK proof.

Finally, the decoding algorithm of G can be used to correct errors in G_c which is a subcode of G. Indeed, the encryption keys G_0 and G_1 are equivalent to C_0 and C_1, respectively. Suppose that for some non-singular $\widetilde{\mathsf{S}}$: $\widetilde{\mathsf{S}}[\mathsf{C}_0|\mathsf{C}_1] = \left[\mathsf{Id}^{k'} \middle| \mathrm{NS}([\mathsf{C}_0|\mathsf{C}_1])\right] = [\mathsf{G}_0|\mathsf{G}_1]$. Then, by linearity, we have $[\mathsf{G}_0|\mathsf{G}_1] = [\widetilde{\mathsf{S}}\mathsf{C}_0|\widetilde{\mathsf{S}}\mathsf{C}_1]$.

Remark 4.1. In Steps 1-5, the players use a coin flipping protocol (commitments are implemented using hash functions) to compute the random seed $\mathbf{s}$ for Q. Our proof of security heavily relies on the fact that both players have no control over the choice of Q. This forces us to assume that the underlying PRG and hash functions are ideal. In other words, we assume for now that:

- The output sequence of the PRG is computationally indistinguishable from random for a randomly chosen seed.
- Hash functions are random oracles. In fact, the random oracle model is also required for the security of IND-CCA2 McEliece encryption (see [15] for details).

We will lift these assumptions in Subsection 4.2 where we provide a realistic non-asymptotic analysis for our construction.

Passive Security. Assume that the players follow the protocol.

Security for Sender. In order to learn $\mathbf{b}_{1-c}$, the receiver must decode $\mathsf{S}'\mathsf{Q}\mathsf{P}'$, i.e., the random code. If k' is not too small, this is infeasible using general decoding techniques. Moreover, as long as the IND-CCA2 conversion [15] is secure, the plaintext $\mathbf{b}_{1-c}$ remains pseudorandom from the dishonest receiver's point of view. This can be shown using Theorem 1 of [15].

Security for Receiver. The only way for the sender to learn c is to distinguish C_c and C_{1-c} which are the random permuted subcodes of G and Q, respectively. The following theorem establishes hardness of this problem (see Appendix C.2 for the proof sketch).

Theorem 4.2. *For the sender, distinguishing C_0 and C_1 is as hard as solving the PKP for $[C_0|C_1]$ and $[G|Q]$.*

Active Security.

Security for Sender. The malicious receiver may try to learn both $\mathbf{b}_0$ and $\mathbf{b}_1$ at least partially with non-negligible probability. As a straightforward attack, he could use a good code instead of Q. However, he will not succeed unless he can break Shamir's ZKIP in Step 5.

The malicious choice of S' is unlikely to help the receiver. If he could find a good subcode of Q, he can would be able to do the same for the McEliece public key, hence violating the security of this PKC.

The receiver cannot hope to substantially improve the code C_{1-c} by exchanging some of its columns with C_c. The ZK proof will still be valid, but C_c will become punctured. Note that there is no better decoding algorithm for the chosen Goppa code than the algebraic decoding. Hence, the error correcting capability will degrade very rapidly with the number of columns exchanged, prohibiting to decode even the "valid" code C_c, while gaining essentially no advantage for C_{1-c}. In the extreme case, when the Goppa code is split into the two subcodes equally, the adversary must guess the error vector corresponding to the random part.[4] It is easy to show that this attack has a higher workload than general decoding of the correctly constructed C_{1-c}.

Security for Receiver. One straightforward attack is to bias the choice of Q in order to efficiently distinguish C_c and C_{1-c}. This requires biasing the coin flipping into generating a seed such that PRG produces

[4] In the same way, as the honest-but-curious attacker against Algorithm 3.2 does.

the output sequence with efficiently computable correlations. Such correlation may allow the random subcode of Q to be distinguished from a random subcode of the Goppa code. Note that breaking the coin flipping requires finding collisions in hash functions. The details on this are out of scope of our paper – we assume the PRG and the hash function ideal. For simplicity of our analysis, we also assume that breaking any of those entities immediately leads to compromising our protocol. Finally, we note that in reality not every collision leads to the choice of a "bad" seed as well as not any "bad" seed leads to a distinguishable subcode.

Reaction Attack. The malicious sender may send a random string instead of one of the encryptions, say for the string $\mathbf{b}_i$. If $i = c$ then the receiver will likely encounter a decoding error, but he would be unable to complain as it reveals his choice. On the other hand, if $i = 1 - c$, the receiver accepts the protocol, yet again revealing c.

This attack can be avoided using the pre-computed oblivious transfer by Beaver [2]. The players can perform OT with random inputs, i.e., "precompute" it. Then, using the Beaver's technique, they perform oblivious transfer with their actual inputs.[5] Clearly, in this case, the above attack is not fruitful, because the receiver can safely reject the pre-computing protocol only revealing the random choice bit instead of c.

See Appendix C.3 for some implementation remarks on our protocol.

4.2 Proposed Parameters

Let us first consider the security for the sender. The dishonest receiver may try to cheat the ZK proof and provide two good codes in Step 5. He can succeed with probability $P_s = 2^{-r}$, where r is the number of rounds in the ZK proof. Let us require that $P_s \leq 10^{-9}$, then 30 rounds are enough.

The above attack must be performed online, while after the completion of the protocol, the receiver can always try to break the encryption of $\mathbf{b}_{1-c}$. The workload of this attack is computed according to equation (1), see Section 2.

For the $[2048, 1333, 131]$ Goppa code with $k' = 1157$ (i.e., the dimension k' of the subcode will be substituted to the above formula instead of k), it provides us with complexity at least $2^{80.02}$.

Now, let us consider the security for the receiver. First, it seems to be very hard to compute P' in the Shamir's ZKP. The best known attack of this kind is due to Poupard [20] but its workload is exponential in n.

[5] The resulting efficiency remains almost the same as only one "randomized" OT is required.

By approximating the theoretical lower bound on Fig. 4 in [20], one can see that the workload for attacking the scheme based on $[n, k]$ code is at least 2^n, even if the attacker can use an infinite memory. Although, we did not try to optimize this attack for our setting, we believe that some other, completely new approach is needed in order to mount a practical attack against the Shamir scheme with large binary matrices.

Hence, we believe that it is more fruitful for the dishonest sender to attempt distinguishing the keys G_0 and G_1 in Step 5 through influencing the choice of the matrix Q.

We assume that the workload of finding "bad" seeds (i.e., those which lead to generation of distinguishable G_0 and G_1) is B. Then, the workload for successful attack is the above plus the workload B' of finding collision in the hash function . Hence, the workload for computing c can be estimated as $\min\{B + B', 2^n\}$.

Finally, we discuss the problem of choosing an appropriate k'. On the one hand, this value should not be too small in order to prevent the general decoding attack. On the other hand, k' should not be too close to k in order to prevent a possible attack using the SSA algorithm. We merely propose to choose k and k' such that $k - k'$ is maximal while the general decoding attack takes at least 2^{80} binary operation. We take the best attack workload 2^{80} as the minimal security requirement.

Our choice $[2048, 1333, 131]$, $k' = 1157$ provides that $k - k' = 176$. The communication cost of our protocol with the above parameters is at most 13.5 Megabytes. This parameter set minimizes the communication cost for a reasonable level of security.

Taking into account the discussion in the previous section, we believe that the attack against ZK proof using the SSA algorithm will have the success probability which is negligible in $k - k'$, however, we do not know the particular parameters. Hence, we propose the second set of parameters (see Table 4.2) providing a stronger security.

Parameters				Success probability for	Workload of decoding	Communication
n	k	t	$k - k'$	online attack on ZKP	(offline) attack	cost, MBytes
2048	1311	67	158	10^{-9}	$2^{80.02}$	13.3
4096	2644	121	500	10^{-9}	$2^{129.38}$	50.4

Table 4.2. Parameter sets for 1-out-of-2 String OT.

5 Conclusion

We have presented a new approach for constructing oblivious transfer based on the McEliece cryptosystem. Two flavors are covered: Rabin OT and 1-out-of-2 OT which can be extended to L-out-of-M OT. Both constructions are practical: the first one with aid of TTP, the second one – against fully malicious players.

A new application for Stern's and Shamir's ZKIP in McEliece-based protocols has been proposed.

As these are one of the first constructions of such kind, the following open questions can be posed:

(i) Fix the gap for the Rabin OT protocol.
(ii) Improve communication efficiency of both constructions. Of particular importance is the L-of-M case.
(iii) Perform the realistic security analysis of Shamir's ZKIP with large binary matrices.
(iv) Investigate the efficiency of applying the SSA algorithm to random permuted subcodes.
(v) Propose the codes which can be used in our 1-out-of-2 OT protocol instead of the Goppa codes. (Note that the answer to this question immediately implies finding good candidate codes for the McEliece PKC).

Finally, we note that the communication cost of Shamir's ZKIP [24] can be decreased as the number of rounds can be reduced from 5 to 3. This is the matter of our ongoing research.

References

1. W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2001.
2. D. Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.
3. M. Bellare and S. Micali. Non-interactive oblivious transfer and spplications. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 547–557. Springer, 1989.
4. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEETIT: IEEE Transactions on Information Theory*, 44, 1998.

5. P.-L. Cayrel, P. Gaborit, and M. Girault. Identity based identification and signature schemes using correcting codes. In *Workshop on coding and cryptography 2007*, pages 69–78. INRIA, 2007.
6. N. Courtois, M. Finiasz, and N.Sendrier. How to achieve a McEliece-based digital signature scheme. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248, pages 157–174. Springer-Verlag, 2001.
7. Claude Crépeau. Equivalence between two flavours of oblivious transfers. In Carl Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 350–354. Springer, 1987.
8. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.
9. O. Goldreich. *Foundations of Cryptography - Volume 2 (Basic Applications)*. Cambridge University Press, 2004.
10. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.
11. V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.
12. I. Haitner. Implementing oblivious transfer using collection of dense trapdoor permutations. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 394–409. Springer, 2004.
13. Y. Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95. Springer, 2005.
14. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.
15. K. Kobara and H. Imai. Semantically secure McEliece public-key cryptosystems - conversions for McEliece PKC. In *Practice and Theory in Public Key Cryptography - PKC '01 Proceedings*. Springer Verlag, 2001.
16. R.J. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44:114–116, 1978.
17. A. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
18. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
19. A. Nascimento, J. Müller-Quade, M. Fossorier, H. Imai, and K. Morozov. Oblivious transfer based on McEliece-like assumptions. Manuscript, 2007.
20. G. Poupard. A realistic security analysis of identification schemes based on combinatorial problems. *European Transactions on Telecommuncations*, 8(5):417–480, September/October 1997.
21. M.O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Aiken Computation Laboratory, Harvard University, 1981. Tech. Memo TR-81.
22. N. Sendrier. On the security of the McEliece public-key cryptosystem. In M. Blaum, P.G. Farrell, and H. van Tilborg, editors, *Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday*, pages 141–163. Kluwer, 2002.
23. N. Sendrier. Finding the permutation between equivalent linear codes: the support splitting algorithm. *IEEE Transactions on Information Theory*, 46:1193–1203, Jul 2000.

24. A. Shamir. An efficient identification scheme based on permuted kernels. In *Proc. of Crypto'89*, volume 435 of *LNCS*, pages 606–609. Springer Verlag, 1990.
25. J. Stern. A new identification scheme based on syndrome decoding. In *Advances in Cryptology - CRYPTO'93*, volume 773 of *LNCS*. Springer Verlag, 1994.
26. Serge Vaudenay. Cryptanalysis of the Chor–Rivest cryptosystem. *J. Cryptology*, 14(2):87–100, 2001.
27. S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.

Appendix A Security Definitions

Definition A.2. *A protocol* $[\mathsf{Sen},\mathsf{Rec}](\mathbf{b})$ *is said to* securely implement Rabin oblivious transfer*, if at the end of its execution, for a random binary variable* E *such that* $Pr[E=0]=\mathcal{P}$*, the following properties hold:*

- Completeness: *Assume the players honestly follow the protocol. If* $E=0$*, then* Rec *outputs* $\mathbf{b}$*, otherwise "erasure".*
- Security for Sen: *After completion of the protocol, if* $E=1$*,* Rec *learns nothing about* $\mathbf{b}$ *in PPT.*
- Security for Rec: *For any* $\mathbf{b}\in\mathbb{F}_2^k$*,* Sen *learns nothing about* E *in PPT.*

Definition A.3. *A protocol* $[\mathsf{Sen},\mathsf{Rec}](\mathbf{b}_0,\mathbf{b}_1;c)$ *is said to* securely implement 1-out-of-2 String oblivious transfer*, if at the end of its execution the following properties hold:*

- Completeness: *when the players honestly follow the protocol,* Rec *outputs* $\mathbf{b}_c$*.*
- Security for Sen: *For any* $c\in\{0,1\}$*,* Rec *learns nothing about* $\mathbf{b}_{1-c}$ *in PPT.*
- Security for Rec: *For any* $\mathbf{b}_0,\mathbf{b}_1\in\mathbb{F}_2^a$ *after completion,* Sen *learns nothing about* c *in PPT.*

This definition can be extended to the L-out-of-M String OT in a natural way.

Appendix B Omitting a TTP in Algorithm 3.1.

Our proposal on the key generation that can be verified by the sender is summarized in Algorithm B.1. However, we do not estimate secure parameters, but just point out the theoretic possibility to omit the TTP.

The last step of Algorithm B.1 requires some additional explanation: After the second last step, the sender knows a $k'\times n$ submatrix K of $\left[\mathsf{G}^{\mathrm{pub}}\middle|\mathsf{G}'\right]$ and can compute the $n+l-k$ dimensional kernel of $\mathsf{O}^{\mathrm{pub}}$ given by a matrix H. Now we can take H and k' as system parameters for Shamir's Permuted Kernel ZKIP. If the receiver is honest and has followed the protocol, he knows the secret permutation $\Pi=\mathsf{P}'$ corresponding to the user's identity K, i.e. a permutation such that $\mathsf{K}\cdot\Pi\cdot\mathsf{H}=\mathbf{0}$. Thus, the honest receiver can employ Shamir's ZKIP to convince the sender by a zero-knowledge proof that he followed the protocol, while the dishonest receiver will be revealed.

Algorithm B.1 Public key generation without TTP

Input: Security parameters $m, t, t', l, k' < 2^m - mt \in \mathbb{N}$,
PRG a pseudo random number generator

Output: The public key $(\mathsf{O}^{\text{pub}}, t')$.

Receiver	**Sender**
Set $n = 2^m$, $k = 2^m - mt$. Generate a McEliece PKC key pair with security parameters m, t. Let $(\mathsf{S}, \mathsf{G}, \mathsf{P})$ be the secret key with public key $(\mathsf{G}^{\text{pub}}, t)$.	
Choose an $(n+l) \times (n+l)$ random permutation matrix P'. Send k commitments to the rows of G^{pub} to the sender.	Choose a seed s_0 and generate the matrix $\text{PRG}(s_0) = \mathsf{G}' \in \mathbb{F}_2^{k \times l}$.
	Send s_0 to the receiver
Publish the systematic matrix O^{pub} generating the same $[n+l, k]$ code as $\left[\mathsf{G}^{\text{pub}} \middle| \mathsf{G}'\right] \mathsf{P}'$, where $\mathsf{G}' = \text{PRG}(s_0)$.	
	Choose a random subset $\mathcal{K}'$ of cardinality k' from $\{1, \ldots, k\}$. Ask the sender to reveal the commitments for $\mathcal{K}'$. Compute the rows of $\left[\mathsf{G}^{\text{pub}} \middle| \mathsf{G}'\right]$ with indices in $\mathcal{K}'$.
Use the protocol of [24] to proof to the sender, that there is a permutation, such that the rows of $\left[\mathsf{G}^{\text{pub}} \middle| \mathsf{G}'\right]$ with indices in $\mathcal{K}'$ are simultaneously in the code generated by O^{pub}.	

Appendix C Details on Security of Algorithm 4.1.

C.1 Intuition

Let us provide some intuition behind Algorithm 4.1. A straightforward approach for the receiver would be to provide the sender (see Step 5) with two matrices: a McEliece public key (derived from the Goppa code G) and a (pseudo)random matrix Q (in the order which depends on his choice c). In the next step, the sender uses the matrices (while being unable to tell which one is the valid key) as the McEliece public keys to encrypt his secrets $\mathbf{b}_0$ and $\mathbf{b}_1$. Hence, the receiver can decrypt $\mathbf{b}_c$ while remaining ignorant about $\mathbf{b}_{1-c}$.

The cheating receiver might send arbitrary matrices in Step 5 (in particular, two valid keys!), hence we have him prove correctness of his input in zero-knowledge. The PKP-based proof requires some rearrangements:

the sender will obtain the random permuted subcodes of G and Q – however, they still can be used as the public keys for $\mathbf{b}_0$ and $\mathbf{b}_1$ with the same effect as before. To make the proof work, the receiver discloses G in Step 1, this will not harm his privacy due to the hardness of the PKP problem.

Finally, one cannot entrust the generation of Q to either player: the dishonest receiver may instead provide a good code, while the cheating sender may try to facilitate his distinguishing of the keys in Step 5. Hence, the players generate a seed for the pseudorandom Q by coin flipping (Steps 1-5).

C.2 Proof Sketch of Theorem 4.2

Let us informally introduce the *Decisional Permuted Kernel Problem (DPKP)* which is about distinguishing a randomly permuted subcode of the given code from a random matrix. First, we note that DPKP must be as hard as PKP.

Lemma C.1. *DPKP is polynomially reducible to PKP.*

Proof (Sketch). The reduction proceeds as follows. [6] First, we puncture the given $[n,k]$ code G at the i-th position and the subcode $\mathsf{S}'\mathsf{G}\mathsf{P}'$ at the j-th position. Then, we submit the two resulting codes to the DPKP oracle. If the oracle answers that the permutation still exists, it is very likely that P' maps the i-th position to the j-th position. Iterating this procedure gives us the permutation up to the automorphism group of $\mathsf{S}'\mathsf{G}\mathsf{P}'$ in $n(n-1)/2$ oracle queries.

Note that the above reduction works only for an oracle which always answers correctly. However, the adversary may have at his disposal an oracle which answers correctly only from time to time. Note that we have got $k' < k$, hence application of the SSA algorithm is a non-trivial task. However, the SSA algorithm can be generalized for computing a permutation of a random subcode of a given code. For k' which are close to k, such the modified algorithm may succeed with non-negligible probability. This will correspond to the non-negligible bias in the DPKP oracle's answers. In this case, we can use the same reduction as above except that we feed our oracle in parallel with several random k''-dimensional ($k'' < k'$) permuted subcodes of G and take the majority vote on the oracle's answers. Together with Lemma C.1 it concludes the proof.

[6] Essentially, the same way as the SSA algorithm [23] does.

As a side remark, let us provide an intuitive explanation on the performance of the modified SSA algorithm. In the original SSA, the so-called signature of a code is computed using the weight distribution of a hull of the code. Now, suppose that $k' = k - 1$. Consider the weight distribution $W(H(\mathsf{C}))$, where $H(\cdot)$ denotes the hull of a code, and C is a randomly permuted $k' \times n$ subcode of the code G. It is easy to see that the components of $W(H(\mathsf{C}))$ will be different from the respective components of $W(H(\mathsf{G}))$ for some constant fraction (which will depend on the particular matrix G). Nonetheless, some correlation of the two weight distributions will be preserved. Hence, the modified algorithm is likely to succeed in computing the permutation. Now, when $k-k'$ is increasing, the shape of $W(H(\mathsf{C}))$ will decorrelate with $W(H(\mathsf{G}))$ very quickly. In fact, for $k - k'$ of about 150, we conjecture that there is no better attack than the following brute force one. Given G and the subcodes $\mathsf{C}_c = \mathsf{S}'\mathsf{G}\mathsf{P}'$ and $\mathsf{C}_{1-c} = \mathsf{S}'\mathsf{Q}\mathsf{P}'$, for any rank k' matrix $\mathsf{S}'' \in \mathbb{F}_2^{k' \times k}$ do the following until the permutation is revealed. Compute $\mathsf{S}''\mathsf{G}$ and run in parallel SSA$(\mathsf{S}''\mathsf{G}, \mathsf{C}_c)$ and SSA$(\mathsf{S}''\mathsf{G}, \mathsf{C}_{1-c})$. There are $2^{k'(k-k')}$ possible matrices S'', hence the workload is exponential in k when k' is of the same order as k.

C.3 Implementation Remarks

Let us provide some remarks on construction of Algorithm 4.1.

- **L-out-of-M String Oblivious Transfer.** This protocol can be easily extended to the case of L-out-of-M Sting OT. Unfortunately, the communication cost is growing quite fast in M mainly due to the contribution by the ZK proof.
- **Trusted Third Party (TTP).** The presence of TTP would greatly simplify the protocol, as it could be delegated generation of the subcodes C_i in the way similar to Algorithm 3.1.
- **Independent Random Source.** Such source can be used by the players to avoid the coin flipping in Steps 1-5 by merely reading $\mathbf{s}$ from the source.
- **Fixed Code.** If the designer allows to use always the same Goppa code in the private key, G may be set as a system parameter but not generated each time anew.