Computational Oblivious Transfer and Interactive Hashing

Kirill Morozov* and George Savvides†

February 13, 2009

Abstract

We present a simple approach for constructing oblivious transfer (OT) using a trapdoor function (TDF) and interactive hashing (IH). In a nutshell, an OT-receiver inputs a (randomly chosen) function index (encoded as a binary string) into IH. The resulting output strings are interpreted by an OT-sender and used to encrypt his private inputs. Two functions are shown to be eligible: 1) A specific candidate function: a coding based McEliece PKC; 2) A collection of TDF with some special properties, loosely speaking: succinctly representable index set and a unique trapdoor for each index.

The aim of this presentation is to show a proof of concept in two ways: 1) Introduction of an apparent connection between OT and IH; 2) Emphasizing importance of IH as a cryptographic primitive in its own right and bringing up some aspects in which the further development of IH may be required.

Keywords: Oblivious transfer, interactive hashing, McEliece assumptions, special contrived trapdoor function

1 Introduction

Oblivious Transfer. It is one of the central cryptographic primitives, since it implies secure two- (and multi-)party computation [16]. It was initially proposed in several flavors [40, 36, 10] which all turned out to be equivalent [6]. We will focus on the *one-out-of-two (1-2) OT* [10] which is a two-party primitive, where the honest sender inputs two bits b_0 and b_1 and the honest receiver chooses to obtain one of them, the *choice bit* b_c, by inputting his *selection bit* c. The cheating sender must remain ignorant of c, while the cheating receiver – ignorant of the *secret bit* b_{1-c}. In this work, we will implement a version of 1-2 OT where both inputs and the selection bit are uniformly random. It can be done without loss of generality due to the (very efficient) randomized self-reduction of OT [1, Section 3.2].[1]

Several frameworks for constructing computationally secure[2] OT are known, in particular, using: generic assumptions [10, 17, 4], smooth projective hashing [22], lossy trapdoor functions [34], dual-mode cryptosystems [33]. Each of them enjoys different advantages, such as generality of underlying assumptions, efficiency, advanced security properties (e.g. composability).

Interactive Hashing. This primitive has first appeared as a sub-routine in the oblivious transfer protocol with computationally unbounded party [32]. Later, IH has proven to have numerous applications in cryptography, in particular, in bit commitment, zero-knowledge, and unconditional oblivious transfer protocol design (see, e.g., [37, Section 2.1] for a survey). IH is a two-party primitive, where the honest sender inputs a string $w \in \{0,1\}^n$, and both

*RCIS, AIST, Japan. E-mail: kirill.morozov@aist.go.jp

†Independent researcher, Germany. E-mail: gsavvides@gmail.com

[1] In other words, a standard 1-2 OT can be easily obtained from its version, where the player's inputs are uniformly random.

[2] That is OT based on computational assumptions, where at least one player is computationally bounded.

him and the honest receiver obtain the *first output* w and the *second output* $w' \neq w$ such that w' is uniform in $\{0,1\}^n \setminus w$. [3] The *hiding* property requires that the cheating receiver cannot tell which of (w, w') was the input. According to the *binding* property, at least one of (w, w') is effectively beyond the control of the cheating sender. IH comes in two flavors: with computational binding (initiated by [32]), we call it C-IH, and information-theoretic binding (initiated by [3]), we call it IT-IH. Hiding is information-theoretic in both flavors.

Our contribution. We present a new connection between the primitives of oblivious transfer and interactive hashing by constructing two (quite simple) protocols using two different flavors of IH and, correspondingly, two types of trapdoor functions. The first one uses a coding based McEliece PKC and IT-IH, while the second one uses a collection of trapdoor functions with special properties and C-IH. In the second one, protection is guaranteed only against passively cheating receiver.

Related Work. In fact, IH has already been used for implementing OT from one-way permutations (OWP) and one-way functions (OWF) [32], but in that work, one of the players was given unlimited computing power (and explicitly instructed to invert an OWP/OWF). In contrast, in our protocols the honest players are bound to run in probabilistic polynomial time.

OT from coding based assumptions was recently constructed in [8] and [24]. Both schemes are more efficient than our first protocol, whose bottleneck is the (round- and communication-)efficiency of the IH protocol. Taking an optimistic view, we note that improvements on IH will automatically apply to our protocol as well. Compared to [8], we use an extra (ad-hoc) coding based assumption. Compared to our first protocol, [24] uses, in addition, the hardness of permuted kernel problem [39], moreover they need an extra (ad-hoc) coding based assumption (different from ours) in order to formally argue the sender-security. In fact, our work is inspired by the protocol [8] in part, where the public key is masked by a random matrix.

OT from general trapdoor permutations is considered in a line of works [10, 17], where the players are assumed passively cheating, while protection against active attackers is achieved using "secure compilers" [16, 18]. Compared to [17], our OT protocol using special trapdoor functions is readily secure in the static adversary model. In fact, IH can be seen as sort of a secure complier and the most of communication overhead derives from it. On the other hand, our scheme does not enjoy the generality of [17] as the additional properties of trapdoor functions which we require are somewhat strict and contrived. Interestingly, they do not include the "density" property of [17].[4]

A construction in the spirit of our first protocol has been independently discovered by Claude Crépeau and Jörn Müller Quade [27].

Organization. Section 2 contains a description of basic notation and tools. The protocol for OT based on the McEliece assumptions and using IT-IH is presented in Section 3, while the one based on special trapdoor functions and C-IH – in Section 4. Section 5 sketches possible extensions and Section 6 presents open questions.

2 Preliminaries

Basic Notation. Summation is bitwise exclusive-or. Saying "weight", we refer to the Hamming weight. "$x \in_R X$" means that x is uniformly distributed in the domain X. Saying that x is *negligible* in n, we mean that x is decreasing faster than any polynomial fraction in n. When the statement is claimed to hold "on the average", it means that it holds for all but a negligible fraction of instances. We call an algorithm *efficient*, if it is (unless stated otherwise) probabilistic polynomial time (PPT). By "$\stackrel{c}{=}$", we denote computational indistinguishability.

[3] Hence, the interactive *hashing*.

[4] We are aware of the recent manuscript [4] reducing the assumptions for achieving OT to dense trapdoor functions. All our comments concerning [17] are relevant to this work as well.

A *view* of a player participating in the interactive protocol represents the player's inputs, results of all local computations and local coin tosses, and messages exchanged. The view of a player A having input x and interacting with a player B having input y is denoted by $View_{\mathsf{A}}(\mathsf{A}(x), \mathsf{B}(y))$.

Speaking of *information-theoretic* or *unconditional* security, we refer to protection against computationally unbounded adversary. A security failure probability (negligibly small in some security parameter) is admitted.

Adversary Model. We consider the *static* adversary, i.e. either a sender or a receiver gets corrupt prior to the protocol execution. A player is called *honest*, if he strictly follows the protocol. A *passive* attacker follows the protocol, but may use his view. Finally, an *active attacker*, in addition to the above, may deviate from the protocol arbitrarily.

Linear codes. A binary linear (n, k) code of *length* n and *dimension* k is a k-dimensional linear subspace of $\{0,1\}^n$. It can be represented by a rank-k *generating matrix* $G \in \{0,1\}^{k\times n}$ (i.e. its rows are linearly independent). For further information on linear codes, we refer the reader to [25].

Fact 1. *The fraction of the matrices in* $\{0,1\}^{k\times n}$ *of rank less than* k *is at most* $O(2^{k-n-1})$.

Proof sketch. Note that the number of all the $(k \times n)$ matrices of rank k is $\prod_{i=0}^{k-1}(2^n - 2^i)$. The claim follows by dividing this value by 2^{kn}, the number of all possible $(k \times n)$ matrices, and estimating the resulting expression. □

McEliece Public Key Cryptosystem (PKC) [26]. As a matter of fact, we do not use security of the McEliece PKC per se, but rather employ two assumptions which together imply its security[5] [38].

Define the space of public keys $PK \stackrel{def}{=} SGP$, where $S \in_R \{0,1\}^{k\times k}$ non-singular; $G \in \{0,1\}^{k\times n}$ a generating matrix of an irreducible Goppa code correcting t errors, $k \geq n - t \cdot \log_2 n$; $P \in \{0,1\}^{n\times n}$ random permutation matrix.[6] Note that $PK \subset D$, where D is the set of all linear (n, k) codes.

Encryption of a message $m \in \{0,1\}^k$ proceeds by choosing a random weight-t vector $e \in \{0,1\}^n$ and computing the ciphertext $c = m(SGP) + e$. Decryption proceeds by first applying P^{-1} to c and then decoding according to G.

The problems underlying the following two assumptions are discussed in details in [38, Section 6]. Let us denote by *Goppa-IND* the problem of distinguishing a randomly sampled McEliece public key from a random linear code.

Assumption 1. *Goppa-IND is hard on the average.*

Goppa-IND is not known to reduce to any hard problem.

Remark 1. *This assumption implies that one can perform the McEliece encryption on* $d \in_R D$ *without noticing that the encryption key does not belong to the public key space. Indeed, otherwise a (trivial) distinguisher for the McEliece public keys could use the encryption algorithm.*

Let *Goppa Bounded Decoding (GBD)* be a standard problem of syndrome decoding with the following promise: the number of errors is guaranteed to be up to t, as in the definition of the Goppa code.

Assumption 2. *GBD is hard on the average.*

In other words, without knowing a structure of the code (as the previous assumption suggests), it is hard to decode errors in the corresponding encoding (and hence, to invert the McEliece encryption). The underlying problem is not known to be NP-complete, but it is related to the bounded distance decoding problem conjectured to be NP-hard, and, in

[5]The opposite is not known to be true, so, technically speaking, these assumptions are stronger than just assuming a secure McEliece encryption.

[6]Strictly speaking, the public key is (SGP, t).

turn, the later is connected to NP-complete syndrome decoding problem (see for details [38, Section 6] and the references therein).

We will need one more ad-hoc assumption. We call a class of (n, k) linear codes *trapdoor Goppa-Bound decodable (trapdoor-GBD)*, if the following holds on the average for all the codes in the class:

- GBD problem is hard given only the code's description.
- GBD problem is easy given the code's description and some auxiliary input (a "decoding trapdoor").

In the above definition, we intend to cover all linear codes with the properties similar to that of the irreducible Goppa codes.

Assumption 3. *Trapdoor-GBD codes constitute a negligible (in $|D|$) fraction of D.*

We consider it to be a plausible assumption, since the opposite implies that a substantial part of all linear codes has an efficient (although "hidden") Goppa Bounded Decoding algorithm.[7]

Generic Functions. The following definitions are adapted from [14, Definitions 2.4.3, 2.4.4].

Definition 1. *A collection of functions $\{f_i : D_i \to \{0,1\}^*\}_{i \in I}$ is called a* collection of trapdoor functions (TDF) *if there exist three efficient algorithms Ind, Dom, $Func$, such that the following three conditions hold:*

1. Easy to sample and compute: *The output distribution of algorithm Ind on input 1^n is a random variable assigned values in the set $(I \times T) \cap (\{0,1\}^n \times \{0,1\}^n)$; I is called a set of* indices *and T a set of* trapdoors. *The output distribution of algorithm Dom on input $i \in I$ is a random variable assigned values in D_i. On input $i \in I$ and $x \in D_i$, algorithm $Func$ always outputs $f_i(x)$.*
2. Hard to invert: *There exists no efficient algorithm which on input $(I_n, f_{I_n}(X_n))$ returns $f_{I_n}^{-1}(f_{I_n}(X_n))$, except with probability negligible in n, where I_n is a random variable describing the distribution of the first element in the output of Ind on input 1^n, and X_n is a random variable describing the output of algorithm Dom on input (random variable) I_n.*
3. Easy to invert with trapdoor: *There exists a deterministic efficient algorithm $Func^{-1}$, such that for every (i, t) in the range of Ind and for every $x \in D_i$, it holds that $Func^{-1}(t, f_i(x)) = x$.*

In [4, Section 3.2], there listed some properties of trapdoor functions which are necessary in order to avoid the famous blackbox impossibility results. We borrow two of them (Properties 4 and 5) which are relevant to our work: 1) The expansion of our functions must be at most logarithmic, i.e., $f_i(x) \in \{0,1\}^m$ for some fixed $m = n + O(\log n)$; 2) our trapdoor function has polynomial pre-image size. Consult [4, Section 3.2] for further details.

A *oneway permutation (OWP)* is defined analogously to the above, except for: 1) Functions $\{f_i : D_i \to D_i\}_{i \in I}$ are one-to-one; 2) No trapdoor is specified (i.e. the algorithm Ind outputs only I) and no easy inversion is required (i.e. the third property in the above definition is absent).

Now, we define a special family of trapdoor functions to be used in our construction.

Definition 2. *A* special contrived *collection of trapdoor functions (SC-TDF) consists of a collection as in Definition 1 with the following additional properties:*

1. *The index set $I \stackrel{def}{=} \{0,1\}^{n'}$, for some $n' < n$, in other words, I has a succinct binary representation. Moreover, this representation is efficiently computable.*

[7] In fact, the trapdoor-GBD codes would become candidates to be used in the McEliece-style encryption instead of the Goppa codes.

2. *There exists a 1-to-1 correspondence*[8] *between* I *and the set of trapdoors* T *(i.e.,* $I \equiv T$*). Moreover, the binary representation for trapdoors is efficiently computable. Furthermore, each function has a unique trapdoor.*

3. *There exists a one-way permutation* $g : T \to I$*.*

Remark 2. *We believe that in the above definition, a slightly relaxed (but suitable for our purposes) version of Property 3 can be derived directly from the first two preceding properties. This would make this definition somewhat less contrived. We postpone the details for the full version of this paper.*

In the following, we will interchangeably use abbreviations (e.g. SC-TDF) to denote the corresponding collections and also the functions constituting them.

Interactive hashing (IH). Interactive Hashing is a two-party cryptographic primitive. It takes as input a string $w \in \{0,1\}^n$ from a sender S, and produces as output two n-bit strings, one of which is w and the other $w' \neq w$ (let us call w and w' the *first* and the *second* output, respectively). The output strings are available to both S and a receiver R, such that, loosely speaking, a dishonest sender $\widetilde{\mathsf{S}}$ cannot control both of them, while a dishonest receiver $\widetilde{\mathsf{R}}$ cannot tell which one was the input. The (dis)honest sender/receiver will be denoted henceforth in this paper as above.

The next definition was taken from [37, Definition 2.1] with some cosmetic changes.

Definition 3. Information-theoretic interactive hashing (IT-IH).

1. Correctness: *When both players are honest, the input is equally likely to be paired with any of the other strings. Let* w *be the sender's input and let* w' *be the second output of IH. Then provided that both players follow the protocol,* w' *will be uniformly distributed among all* $2^n - 1$ *strings different from* w*.*

2. Hiding: $\widetilde{\mathsf{R}}$ *cannot tell which of the two output strings was the original input. Let the two output strings be* w_0,w_1*, labeled according to lexicographic order. Then if both strings were a priori equally likely to have been the senders input* w*, then they are a posteriori equally likely as well.*

3. Binding: $\widetilde{\mathsf{S}}$ *cannot force both outputs to have a rare property. Let* G *be a subset of* $\{0,1\}^n$ *representing* $\widetilde{\mathsf{S}}$*'s target set, it must be fixed prior to the protocol execution. Let* $|G|$ *be the cardinality of* G*. Then if* $|G|/2^n$ *is negligible, the probability that* $\widetilde{\mathsf{S}}$ *will succeed in having both outputs* w, w' *be in* G *is also negligible.*

The following definition (focusing on a particular scenario of [30]) for the computational flavor of IH is adapted from [20, Section 3.1].

Definition 4. Computational interactive hashing (C-IH).

Let $\mathcal{H} : \{0,1\}^n \to \{0,1\}^{n-1}$ *be a family of two-to-one Boolean hash functions and* $g : \{0,1\}^n \to \{0,1\}^n$ *be a one-way permutation. Both players receive the security parameter* 1^n *as an input and* S *gets as a private input* $y \in \{0,1\}^n$*. At the end,* S *locally outputs* y *and, in addition, both* S *and* R *output* $(h, z) \in \mathcal{H} \times \{0,1\}^{n-1}$*. The following properties must be satisfied:*

1. Correctness: *For all* n*, all* $y \in \{0,1\}^n$*, and every pair* $(y, (h, z))$ *that may be output by* $(\mathsf{S}(1^n, y), \mathsf{R}(1^n))$*, it is the case that* $h(y) = z$*.*

2. Binding: *No PPT* $\widetilde{\mathsf{S}}$ *succeeds in the following game with more than negligible probability. On security parameter* 1^n*,* $\widetilde{\mathsf{S}}$ *interacts with* R *and* R *outputs pairs* $(x_0, y_0), (x_1, y_1)$ *such that* $y_0 = g(x_0)$*,* $y_1 = g(x_1)$ *and* $h(y_0) = h(y_1) = z$*.*

3. Hiding: *There exists a polynomial-time simulator* Sim *such that for every* $y \in \{0,1\}^n$ *and* $h \in \mathcal{H}$ *the distributions* $View_{\mathsf{R}}(\mathsf{S}(y), \mathsf{R})(1^n)$ *and* $Sim(1^n, h, h(y))$ *are identical.*

[8] Hence, there exists a succinct binary representation for trapdoors.

In our work, we abstract the implementation details as much as possible and use IH as a blackbox. We focus on a connection between OT and IH, and any IH protocols which satisfies one of the above definitions fits one of our constructions.

Concrete protocols realizing C-IH and IT-IH can be found, e.g., in [30, Section 3.1] (appears as a part of the commit stage) and [37, Protocol 2.1], respectively. These protocols are very similar, the main difference between them is in construction of receiver's queries (refer to Appendix A for description and some details). The main disadvantage of these protocols is their round and communication complexity: for n-bit input, they require $n-1$ rounds and $O(n^2)$ bits of communication.[9]

String Commitment. This primitive consists of two stages. In the first one, called *committing*, the sender S provides the receiver R with an evidence about his input bit-string b. The cheating receiver $\widetilde{\mathsf{R}}$ cannot learn b before the second stage, called *opening*, where S reveals b to R. The cheating sender $\widetilde{\mathsf{S}}$ cannot successfully open anything other than b.

We borrow the following definition from [8, Definition 1]. In this and the following definitions, we denote by z an auxiliary input of the cheating party.

Definition 5. *A protocol* $[\mathsf{S},\mathsf{R}](b)$ *is said to* secure implement bit commitment*, if at the end of its execution by* S *and* R *which are represented as PPT Turing machines having as their input a security parameter* n*, the following properties hold:*

- Correctness: *when the players honestly follow the protocol,* R *accepts* b *at the opening stage.*
- Hiding: *For any PPT adversary* $\widetilde{\mathsf{R}}$*, any security parameter* N*, any input* z *of size polynomial in* n*, and any* $l \in \mathbb{N}$*, after the committing stage, but before the opening stage, the view of* $\widetilde{\mathsf{R}}(z)$ *when* S *inputs* $b \in \{0,1\}^l$ *is computationally indistinguishable from the view where* S *inputs any other* $b' \in \{0,1\}^l$*,* $b' \neq b$*:*

$$View_{\widetilde{\mathsf{R}}}(\mathsf{S}(b), \widetilde{\mathsf{R}}(z))|_z \stackrel{c}{=} View_{\widetilde{\mathsf{R}}}(\mathsf{S}(b'), \widetilde{\mathsf{R}}(z))|_z$$

- Binding: *For any PPT adversary* $\widetilde{\mathsf{S}}$*, any security parameter* n *and any input* z *of size polynomial in* n*, any* $l \in \mathbb{N}$*, there exists* $b \in \{0,1\}^l$ *which can be computed by* S *after the committing stage, such that the probability that* $\widetilde{\mathsf{S}}(b')$*,* $b' \neq b$ *is accepted by* R *in the opening stage is negligible in* n*:*

$$Pr\left[View_{\mathsf{R}}\left(\widetilde{\mathsf{S}}_{commit}(b), \widetilde{\mathsf{S}}_{open}(b'), b \neq b'\right) = ACCEPT\right] < \frac{1}{poly(n)}.$$

We assume availability of commitments as a blackbox.[10]

Oblivious transfer. We focus on 1-out-of-2 bit oblivious transfer [10]: The honest sender S transmits two input bits b_0, b_1 such that one of them b_c is obtained by the honest receiver R according to his input - the selection bit c. The dishonest sender $\widetilde{\mathsf{S}}$ cannot learn c, while the dishonest receiver $\widetilde{\mathsf{R}}$ cannot learn both b_0 and b_1.

In our schemes, c is chosen independently of receiver, in the course of protocol execution. This is not standard as typically c is input by receiver. We claim that this is not a problem due to a (very efficient) reduction [1, Section 3.2] from OT with randomized (uniformly chosen) player's inputs to OT with inputs fixed by players. We can assume w.l.o.g. that, if standard OT is required, this reduction is applied upon completion of our protocol. Therefore, in our security argument, we will ignore protection against $\widetilde{\mathsf{R}}$ trying to bias the choice of c,[11] only concentrating on inability of $\widetilde{\mathsf{S}}$ to learn c.

[9] There exists a constant round (information-theoretic) IH protocol [7] with roughly the same communication cost. However, its security requirements are somewhat stricter compared to ours, so that we cannot use it in our schemes. Nonetheless, thinking optimistically, this result shows us that improvements are possible in principle.

[10] If one wants to restrain oneself to the coding assumptions as in [8, Section 2.5], then one can use an (efficient) construction [28] of commitment from pseudorandom generators which, in turn, can be constructed assuming hardness of syndrome decoding [11].

[11] Such the protection is in fact easy to ensure, as pointed out in Remark 3.

Our protocols involve sender transmitting two encryptions of his corresponding inputs. One standard sender's attack is to render one of the encryptions invalid, hoping that receiver complains and reveals his selection. Clearly, this attack is fruitless in our scenario, where the selection bit is random.

We borrow [8, Definition 1] (adapted from [22]) with a slight modification to randomized selection bit.

Definition 6 ([8], Definition 1)**.** *A protocol* $[\mathsf{S},\mathsf{R}](b_0,b_1)$ *is said to* securely implement randomized oblivious transfer*, if at the end of its execution by the sender* S *and the receiver* R *which are represented by PPT algorithms having as their input a security parameter* n*, such that the following properties hold:*

- Correctness: *when the players honestly follow the protocol,* R *outputs* (c, b_c) *for* $c \in_R \{0,1\}$ *while* S *has no output.*
- Sender-security: *For every PPT adversary* $\widetilde{\mathsf{R}}$*, every input* z*, a (sufficiently long) random tape* R_{S}*, there exists a selection bit* c *such that for* $b_c \in \{0,1\}$ *the distribution (taken over* S*'s randomness) of runs of* $\widetilde{\mathsf{R}}(z)$ *using randomness* R_{S} *with* S *having input* b_c *and* $b_{1-c}=0$ *is computationally indistinguishable from the distribution of runs with* S *having input* b_c *and* $b_{1-c}=1$*.*
- Receiver-security: *For any PPT adversary* $\widetilde{\mathsf{S}}$*, any security parameter* n *and any input* z *of size polynomial in* n*, the view that* $\widetilde{\mathsf{S}}(z)$ *obtains when* $c=0$ *is computationally indistinguishable from that of when* $c=1$*, denoted:*

$$\{View_{\widetilde{\mathsf{S}}}(\widetilde{\mathsf{S}}(z),\mathsf{R})\}_z \stackrel{c}{=} \{View_{\widetilde{\mathsf{S}}}(\widetilde{\mathsf{S}}(z),\mathsf{R})\}_z.$$

Hardcore Bit Encryption. Let x and r be bit vectors of appropriate length and denote the scalar product by "$\cdot$". It follows from [15] that for any TDF f, $x \cdot r$ is a *hardcore bit*, i.e. it cannot be guessed substantially better than at random, given $(f, f(x), r)$.

Then, encrypting a plaintext bit b given a (candidate) trapdoor function f amounts to generating (x, r) at random and computing the ciphertext as the triple $(b + (x \cdot r), f(x), r)$, where "+" is a bitwise exclusive-or. Decryption is performed in a straight forward way, inverting f using its trapdoor. Clearly, without the trapdoor, b is as hard to guess as the hardcore bit of f.

3 Coding Based Protocol

In this section, we restrict our consideration to the range of (n, k), which is relevant to the secure McEliece PKC with parameters (n, t).

Fact 2. *The fraction of all* $(k \times n)$ *matrices which do not represent a binary linear code is negligible in* n*.*

Proof. Follows taking account Fact 1, and that in the McEliece PKC parameter set: k is less than and in the order of n. □

Let us re-assign $D \stackrel{def}{=} \{0,1\}^{k\times n}$, most of these matrices represent linear codes as mentioned above. Let us assume a secure string commitment and a secure implementation of IT-IH (in this section, we call it "IH" for short) to be available as a blackbox. Let $Enc_{pk}(b)$ be a hardcore bit encryption of a bit b where trapdoor function is the McEliece encryption having a public key $pk \in PK$. When binary matrix is to be input into IH, it is represented as a bit string by concatenating the rows.

The sender S has inputs $b_0, b_1 \in_R \{0,1\}$, the receiver R has no input. Note that R plays the role of a sender in IH.

Protocol 1.

1. R generates $R \in_R D$ and commits to it.

2. R generates $pk \in_R PK$ and inputs $pk + R$ into IH, then both S and R obtain outputs which are assigned to (W_0, W_1) according to lexicographic order.
3. R opens his commitment to R and S aborts, if the opening stage is rejected.
4. For $i = 0, 1$: S sends $Enc_{W_i+R}(b_i)$ to R.
   R computes $c = \{i | W_i + R = pk\}$, decrypts b_c and outputs (c, b_c).

Remark 3. *The above protocol does not guarantee that $\widetilde{\mathsf{R}}$ is unable to fix the selection bit c. However, it can be easily modified to achieve this goal as follows. Upon completion of the IH protocol, the sender chooses uniformly at random whether (W_0, W_1) are assigned in lexicographic or reverse lexicographic order. Clearly, the selection bit is beyond $\widetilde{\mathsf{R}}$'s control in this case.*

Let us informally explain our rationale behind employment of commitment. Suppose that R directly inputs the public key pk into IH. First of all, on the output, the public key is a priori not likely to pair with other members of the domain D as Property 2, Definition 3 requires for the IH input. Indeed, the public keys constitute only a negligible fraction of D as established in Fact 3 below. Then, $\widetilde{\mathsf{S}}$ might deliberately drive the second input into some recognizable subset $D \setminus PK$ (Definition 3 does not forbid this), and hereby learn the selection bit c.[12]

A possible way out is to ensure that R's input looks uniform to $\widetilde{\mathsf{S}}$. At the same time, this input must be fixed prior to performing IH in order to prevent $\widetilde{\mathsf{R}}$ from cheating. Commitment seems to be a natural solution in this case.

Remark 4. *We believe, but do not prove formally, that the Niederreiter PKC [31] can be used in the above protocol in a similar manner.*

Proposition 1. *Protocol 1 is a secure implementation of 1-2 OT according to Definition 6 under Assumptions 1-3.*

Proof. **Correctness.** When both players are honest, one of the outputs of IH is indeed $pk + R$ by Property 1, Definition 3. Then in Step 4, R (knowing both W_0 and W_1) correctly identifies W_c, computes c and successfully decrypts b_c.
Sender-security. If $\widetilde{\mathsf{R}}$'s commitment is not rejected in Step 3, then the encryption keys are fixed to $(Z + R', Z' + R')$, where $Z, Z' \in D$ are the IH outputs and $R' \in D$ is the committed value.

Note that the receiver's only efforts in the protocol are connected with the choice of keys for b_0 and b_1. Hence, his attack must be about encrypting both of them on "efficiently decodable" keys. In other words, $\widetilde{\mathsf{R}}$ succeeds if he manages to invert the encryption on both $Z + R'$ and $Z' + R'$ with non-negligible probability. The sender-security proof is focused on arguing that this cannot be the case. If so, then at least one of the encryptions is one-way. In turn, it will imply that $\widetilde{\mathsf{R}}$'s views are indistinguishable no matter the value of b_{1-c} for some $c \in \{0, 1\}$, due to employment of hardcore bit encryption.

The key point of our proof is a use of Property 3, Definition 3. Let us denote by *the bad set* the set of all possible elements in D which allow for efficient inversion (in other words, the GBD problem is easy for such the elements of D). The above mentioned property implies that if the cardinality of the bad set is negligible in n, then the probability for $\widetilde{\mathsf{R}}$ to land both $Z + R$ and $Z' + R'$ in this set is also negligible in n. Hence, we are done by arguing that the bad set is indeed a negligible fraction of D. The rest of the sender-security proof is focused on arguing this point.

Let us show that the bad set is, in fact, the set of all (n, k) codes for which GBD problem can be efficiently solved using auxiliary input of polynomial size. We begin by identifying two important subsets constituting the bad set. The first subset consists of the codes for which GBD is easy given only a description of the code (e.g. some good LDPC codes). This subset (call it D') is handled in Lemma 1 below. Secondly, we note that although the

[12] It would be tempting to argue that the existence of such the malicious sender algorithm implies some bad thing, like a distinguisher for public keys. However, it does not seems to be easy.

public key set PK does not intersect much with D' (by Assumptions 1 and 2), it must also be included into the bad set. This is because $\widetilde{\mathsf{R}}$ can compute trapdoors for some $pk \in PK$ even during the protocol execution (hence making them easy-to-decode), while the bad set must be fixed prior to it (as required in Property 3, Definition 3). The easiest solution is to conservatively include the whole of PK into the bad set. This reasoning led us to introducing the trapdoor-GBD codes (which include PK as the special case) – they constitute the second subset.

Lemma 1. *Under Assumption 2, $|D'|/|D|$ is negligible.*

Proof. The opposite suggests that with non-negligible probability, for H as in Assumption 2 which is chosen uniformly at random, its corresponding generating matrix $H^{\perp} \in D'$. In other words, such the code is efficiently decodable, a contradiction. □

Fact 3. *$|PK|/|D|$ is negligible.*

Proof. Let us estimate the number of the McEliece public keys. According to [38, Section 2.2.2.], the number of the (Goppa) generator polynomials can be estimated as $2^{mt}/t = 2^{n-k}$. Each of them can be permuted ($n!$ variants) to yield the public key.[13] Remember that $m = \log n$ and let us for simplicity take $k = n - mt$. We have the following upper bound $|PK| < n! \cdot 2^{n-k} = n! \cdot 2^{-t \log n} = O(n^n) \cdot 2^{-t \log n} = 2^{O(n \log n)} \cdot 2^{-t \log n} = 2^{O((n-t) \log n)}$. The second equality follows by the Stirling approximation. Clearly, the latter number constitutes a negligible fraction of $|D| = 2^{nk}$.

□

As for the trapdoor-GBD codes in general, note that any large code family in this class is a potential candidate to be used in a coding based PKC. The search for such the candidates was not very successful so far as noted in [9, Section 1.1]. Although finding a trapdoor-GBD family of the non-negligible size (or superpolynomially many "small" families) does not seem to be an easy task, their existence do not contradict to Assumption 2. Hence, we employ Assumption 3 to simplify our security argument.

We conclude that Lemma 1 and Fact 3 along with Assumption 3 imply the sender-security.

Remark 5. *Now it is a right time to explain why the constant-round interactive hashing of [7] cannot be directly used in our protocol. Their security proof requires a priori knowledge of the upper bound on the bad set size. In our case, this translates into the explicit upper bound on the number of codes in the bad set, which seems to be hard to obtain.*

Receiver-security. By Property 2, Definition 3, $\widetilde{\mathsf{S}}$ cannot tell R's input among W_0 and W_1, hence IH does not help him. Moreover, if $\widetilde{\mathsf{S}}$ efficiently recognized a public key among $(W_0 + R, W_1 + R)$, this would contradict to Assumption 1.

Note that no matter what $\widetilde{\mathsf{S}}$ sends in Step 4, c will be defined by R as $c = \{i | W_i = pk + R\}$, where W_i, $i \in \{0, 1\}$ follows lexicographic order. Such c is out of $\widetilde{\mathsf{S}}$'s control, since R is not known to him (by the hiding property of commitment) during the IH execution.

We conclude that the views of $\widetilde{\mathsf{S}}$ are indistinguishable for either value of c, when the protocol is accepted. □

Remark 6. *We emphasize that we have no evidence that some proof similar to that of [30] will not work in our case. However in this work, we are mostly after presenting the concept. Thus, we prefer to go by Assumption 3 rather than constructing the whole proof in the computational IH scenario.*

[13] For simplicity, we disregard the permuted polynomials, which produce the same code, since there are are only polynomially many of them [38, Section 2.2.2.].

One argument for constructing the aforementioned new proof is as follows. In Protocol 1, it might be possible to substitute the McEliece PKC with any PKC having the public key whose binary representation is: 1) indistinguishable from random; 2) "sparse" in the sense of Fact 3. Unfortunately, the potential existence of "trapdoor-GBD" keys (along with necessity to include the whole public key space into the bad set) makes the analysis somewhat complicated and contrived. Proving a statement to the effect "if computationally bounded OT-receiver in Protocol 1 learns both inputs, then he also successfully cheats in IH" would substantially simplify the security proof, and would allow for the above mentioned generalization.

In the next section, we take another way around this problem. We explore limits of our approach by employing C-IH and trapdoor functions with special properties. These properties give us an insight as for which cryptosystems are admissible for our construction.

4 Protocol Using Special Trapdoor Functions

Intuition. The receiver uniformly samples a function from SC-TDF along with its trapdoor and inputs the function's index (but not the trapdoor) into C-IH (in this section, we call it "IH" for brevity), resulting in two function indices on the output. The sender uses these functions to encrypt his inputs, respectively (in addition, indicating which function was used for which input). The sender-security is guaranteed computationally since the cheating receiver is unlikely to find a trapdoor for both SC-TDF's. The receiver-security is guaranteed information-theoretically since for the cheating sender, either function looks equally likely to be chosen by the receiver, according to the hiding property of IH.

Let F be a collection of SC-TDF (according to Definition 2) with index set I. Define by $f_i^h(b)$ a hardcore bit encryption (as described in Section 2) of the bit b with a function $f_i \in F$ (determined by an index $i \in I$) as trapdoor function. IH is employed in a blackbox manner. When an index $i \in I$ is to be input into IH, its binary representation is used.

The sender S has inputs $b_0, b_1 \in_R \{0,1\}$, the receiver R has no input. Note that, as in the previous protocol, R plays the role of an IH-sender.

Protocol 2.

1. R generates $i \in_R I$.
   R inputs i to IH, both S and R obtain (i_0, i_1) as output.
   The outputs are assigned to i_j according to lexicographic order.
2. For $j = 0, 1 : \mathsf{S}$ sends $(i_j, f_{i_j}^h(b_j))$.
   R computes $c = \{j | i = i_j\}$ and decrypts b_c.

Note that in Step 2, in order to save on communication, S may send some (short) identifiers instead of i_j (say, a value of the least significant bit where i_0 and i_1 differ).

Proposition 2. *Protocol 2 is a secure implementation of 1-2 OT according to Definition 6 with information-theoretic receiver-security, assuming that SC-TDF exist and the receiver only cheats passively.*

Proof. **Correctness.** When both players are honest, one of the IH-outputs is indeed the index i by the IH-correctness property. Then in Step 2, R correctly identifies i_c and successfully decrypts the corresponding sender's input b_c.
Sender-security. By Property 1, Definition 2, any input and output of IH is an index in I. Due to the use of hardcore bit encryption, $\widetilde{\mathsf{R}}$ must invert both f_{i_c} and $f_{i_{1-c}}$ in order to learn both b_0 and b_1.

Since $\widetilde{\mathsf{R}}$ follows the protocol, his input to (and the first output of) IH is $i = i_c$ for some $c \in \{0,1\}$. The second output i_{1-c} will be uniformly distributed in $I \setminus i_c$ according to Property 1, Definition 3. Then, by the hard-to-invert property of TDF (Property 2, Definition 1), $\widetilde{\mathsf{R}}$ cannot invert $f_{i_{1-c}}$ with non-negligible probability. This concludes the sender-security proof.

Receiver-security. Note that $i \in_R I$, therefore by Property 3, Definition 4, even computationally unbounded $\widetilde{\mathsf{S}}$ cannot tell R's input among i_0 and i_1.

No matter what $\widetilde{\mathsf{S}}$ actually sends in Step 4, c will be defined by R as $c = \{j | i = i_j\}$, where i_j, $j \in \{0, 1\}$ follow lexicographic order. Such c is out of $\widetilde{\mathsf{S}}$'s control, since the latter cannot tell apart i_0 and i_1. We conclude that the views of $\widetilde{\mathsf{S}}$ are indistinguishable for either value of c, when the protocol is accepted. □

It is worth noting that the abstract nature of TDF is essential for the above construction to work. An actual candidate TDF (a public key cryptosystem) cannot be plugged into our scheme immediately. The main problem is to satisfy Property 1, Definition 1 which requires efficient indexing of the public keys. On the positive side, one has freedom of choosing a particular binary representation for public keys to be input to IH. This might help in constructing a more efficient implementation.

5 Possible Extensions

Next, we discuss possible extensions to our work.

Security Against Actively Cheating Receiver In Protocol 2. Note that the existence of OWP from index set I to trapdoor set T (Property 3, Definition 2) has never been used in the sender-security proof of Proposition 2. This is because it is tailored to provide an active security against receiver in Protocol 2. Unfortunately, the protection can be provided only in the limited sense: one can guarantee that $\widetilde{\mathsf{R}}$ does not learn trapdoors for both f_{i_0} and f_{i_1}. This is weaker than to guarantee that $\widetilde{\mathsf{R}}$ cannot *invert* both f_{i_0} and f_{i_1}. Below is an intuition behind the proof of the above fact.

Consider the proof of the binding property (i.e. Property 2, Definition 4) in [30] and compare it to our case. Remember that the IH-sender is assumed computationally bounded. The authors construct an unconditionally hiding commitment using OWP and prove that if the sender can successfully cheat, then he can also invert the underlying OWP. More precisely, their protocol [30, Section 3.1] is constructed in such a way that successfully cheating IH-sender will surely be able to invert OWP in the scenario described in Property 2, Definition 4. In our protocol, the input to IH is an index of SC-TDP. Intuitively, our scenario is related to the one above: the OT-receiver (who is the sender in IH) must not be able to compute a trapdoor for both f_{i_0} and f_{i_1}. We do not know how to prove the latter statement directly if only a standard TDF is assumed. Hence, we take a way around by constructing a special (and somewhat contrived) trapdoor function SC-TDF (see Definition 2) in which the relation between indices and trapdoors is OWP and so the proof of [30] applies directly.

Alternative SC-TDF Based Protocol. We believe that communication efficiency of our second construction (taking into account the cost of IH) can be improved, if we replace the McEliece PKC with SC-TDF in the protocol [8] in the trivial manner. The formal proof of this claim is postponed to the full version of this paper. Let us now compare the two constructions, calling the new scheme *Protocol 2-a*.

In contrast to Protocol 2, Protocol 2-a does not involve IH and admits a probability of sender-security failure which decreases exponentially fast in the number of honest-protocol iterations. Moreover, Protocol 2-a requires commitments. If one wants to have the information-theoretical receiver-security, the scheme [21] can be used, although it is quite involving, and hence it is unclear whether any savings in communication cost can be achieved. However, if computational receiver-security is in order, then an efficient construction [28] can be employed.

The sender-security proof is simpler in Protocol 2-a, either f_{i_0} or f_{i_1} is guaranteed to be uniformly chosen by construction. Hence, the hard-to-invert property is directly applicable (see the corresponding subsection in the proof of Proposition 2).

Protocol 2-a compares quite favorably to [17, 4] in terms of communication complexity. The main challenge for improving it is to reduce its computational assumptions, preferably all the way down to standard TDF.

Possible SC-TDP Candidate. We believe but do not prove formally that the ElGamal PKC [13] can be considered as a SC-TDP candidate. Let us present our intuition behind this claim. First, we very briefly sketch the key generation algorithm. Let G be a multiplicative cyclic group of a (large) prime order q with a generator g. The key generation algorithm picks $x \in_R \{0, \ldots, q-1\}$, computes $h = g^x \mod q$ and outputs the public key $pk = (G, q, g, h)$ and the secret key x. From now on, we will refer to only h as the "public key". The ElGamal function is a well-known TDF candidate. Let us show that it also satisfies the "special contrivance" properties according to Definition 2, starting with the second one. Indeed, both the index and the trapdoors sets can be represented by $\{0, \ldots, q-1\}$. And indeed, the function computing the public keys from elements of G is an OWP. The problem seems to be the first property of Definition 2 demanding the succinct representation of G. The number p must be prime, hence the binary representation will certainly have some invalid encodings.

In the context of IT-IH, the problem of set encoding was addressed in [3] (using [5]) and later developed in [7] and [37, Section 4.2.1]. Showing that the ElGamal PKC is a SC-TDF candidate will amount to arguing that the proof of [30] can accommodate some particular encoding of $\{0, \ldots, q-1\}$ to binary strings. The challenge comes from the fact that such a representation either must contain invalid encodings or some (or all) of set elements will be contained more than once.

The ultimate goal of investigation on an appropriate public key encoding would be to show that a standard TDF with a "dense" (but not succinct) index set in $\{0,1\}^n$ can be used in Protocol 2.

String Oblivious Transfer. Note that up to $\log k$ hardcore bits (where k is the range size of the underlying trapdoor function) are available in the hardcore bit encryption [15]. Hence, our constructions can be easily generalized to OT of short strings instead of bits.

If one needs obliviously transfer bit strings of arbitrary length, then either generic conversions [35, 12, 23] (which provide for indistinguishability of encryptions) can be used[14] or generic reductions from Bit OT to String OT can be employed (see [37] for the survey and some new constructions).

6 Open Questions

Interactive Hashing. It is interesting to look for new natural definitions of IH in cryptography. Another important question is efficiency of IH protocols. Although it was shown in [19] that based on OWP in blackbox manner, IH must have $\Omega(n/\log n)$ rounds, this question remains open for specific computational assumptions. Moreover, in our application, commitments are used along with IH. Providing an IH protocol with reduced communication complexity (possibly using commitments) will automatically improve our protocols as well.

Upgrading Security of Protocol 2. Proving security against actively cheating receiver.

Indistinguishable Public Key PKC's. It is interesting to generalize Protocol 1 for the case of any PKC with the public key indistinguishable from random (e.g., lattice based PKC's). To this effect, it would be interesting to show a rigorous proof of security showing that the OT-receiver, who adaptively pre-compute trapdoors, does not pose a threat.

SC-TDF Candidates. Can any PKC qualify as such?

References

[1] D. Beaver. Precomputing oblivious transfer. In D. Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.

[14] These schemes typically require random oracles which are not used in our scheme.

[2] G. Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*. Springer, 1990.

[3] C. Cachin, C. Crépeau, and J. Marcil. Oblivious transfer with a memory-bounded receiver. In *FOCS*, pages 493–502, 1998.

[4] K. Y. Cheong and T. Koshiba. Reducing complexity assumptions for oblivious transfer. Cryptology ePrint Archive, March 2008. Available at: http://eprint.iacr.org/2008/140.pdf.

[5] T. Cover. Enumerative source encoding. *IEEE Transactions on Information Theory*, 19:73–77, 1973.

[6] C. Crépeau. Equivalence between two flavours of oblivious transfers. In C. Pomerance, editor, *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 350–354. Springer, 1987.

[7] Y. Z. Ding, D. Harnik, A. Rosen, and R. Shaltiel. Constant-round oblivious transfer in the bounded storage model. *J. Cryptology*, 20(2):165–202, 2007. Conference version appears at TCC '04.

[8] R. Dowsley, J. van de Graaf, J. Müller-Quade, and A. C. A. Nascimento. Oblivious transfer based on the mceliece assumptions. In R. Safavi-Naini, editor, *ICITS*, volume 5155 of *Lecture Notes in Computer Science*, pages 107–117. Springer, 2008.

[9] D. Engelbert, R. Overbeck, and A. Schmidt. A summary of McEliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, 1(2):151–199, 2007.

[10] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[11] J.-B. Fischer and J. Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 245–255. Springer, 1996.

[12] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.

[13] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.

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

[15] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32. ACM, 1989.

[16] 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.

[17] I. Haitner. Implementing oblivious transfer using collection of dense trapdoor permutations. In Naor [29], pages 394–409.

[18] I. Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 412–426. Springer, 2008.

[19] I. Haitner, J. J. Hoch, O. Reingold, and G. Segev. Finding collisions in interactive protocols - a tight lower bound on the round complexity of statistically-hiding commitments. In *FOCS*, pages 669–679. IEEE Computer Society, 2007.

[20] I. Haitner and O. Reingold. A new interactive hashing theorem. In *IEEE Conference on Computational Complexity*, pages 319–332, 2007.

[21] I. Haitner and O. Reingold. Statistically-hiding commitment from any one-way function. In D. S. Johnson and U. Feige, editors, *STOC*, pages 1–10. ACM, 2007.

[22] Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95. Springer, 2005.

[23] K. Kobara and H. Imai. Semantically secure mceliece public-key cryptosystems-conversions for mceliece pkc. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2001.

[24] K. Kobara, K. Morozov, and R. Overbeck. Coding-based oblivious transfer. Accepted to MMICS 2008, Karlsruhe, Germany, December 2008. Available at: http://eprint.iacr.org/2007/382.pdf.

[25] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library. Elsevier, 12 edition, 2006.

[26] R. J. McEliece. A public key cryptosystem based on algebraic coding theory. *DSN progress report*, 42-44:114–116, 1978.

[27] J. Müller-Quade. Personal communication, December 2008.

[28] M. Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[29] M. Naor, editor. *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*. Springer, 2004.

[30] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for p using any one-way permutation. *J. Cryptology*, 11(2):87–108, 1998.

[31] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Probl. Control and Inform. Theory*, 15:19–34, 1986.

[32] R. Ostrovsky, R. Venkatesan, and M. Yung. Fair games against an all-powerful adversary. In *AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 155–169, 1993.

[33] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

[34] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *STOC*, pages 187–196. ACM, 2008.

[35] D. Pointcheval. Chosen-ciphertext security for any one-way cryptosystem. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 129–146. Springer, 2000.

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

[37] G. Savvides. *Interactive Hashing and reductions between Oblivious Transfer variants*. PhD thesis, School of Computer Science, McGill University, Montreal, Canada, 2007.

[38] N. Sendrier. On the security of the McEliece public-key cryptosystem. In M. Blaum, P. G. Farrell, and H. C. A. van Tilborg, editors, *Information, Coding and Mathematics*, pages 141–163. Kluwer, 2002. Proceedings of Workshop honoring Prof. Bob McEliece on his 60th birthday.

[39] A. Shamir. An efficient identification scheme based on permuted kernels (extended abstract). In Brassard [2], pages 606–609.

[40] S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.

Appendix A Interactive Hashing Protocol

The following protocol was introduced in [32]. Let w be a n-bit string that the sender S wishes to transmit to the receiver R. All operations take place in the binary field $\mathbb{F}_2$.

Protocol 3. Interactive Hashing

1. R chooses a $(n-1) \times n$ matrix Q uniformly at random among all binary matrices of rank $n-1$. Let q_i the i-th query, consisting of the i-th row of Q.
2. For $1 \leq i \leq n-1$ do:
   (a) R sends query q_i to S.
   (b) S responds with $c_i = qi \cdot w$.
3. Given Q and c (the vector of Rs responses), both parties compute the two values of w consistent with the linear system $Q \cdot w = c$. These solutions are labeled w_0, w_1 according to lexicographic order.

A proof that the above protocol is IT-IH (i.e. satisfies Definition 3) can be found in [37, Section 2.3]. A slight modification of Protocol 3 is shown to satisfy Definition 4 (i.e. to be C-IH) in [30], with a tighter security proof presented in [20] (along with several generalizations). Its only difference compared to Protocol 3 is the matrix Q chosen in a canonical way.