A Framework for Efficient and Composable Oblivious Transfer

Chris Peikert[1,⋆], Vinod Vaikuntanathan[2,⋆⋆], and Brent Waters[1,⋆⋆⋆]

[1] SRI International, `cpeikert@alum.mit.edu, bwaters@csl.sri.com`
[2] MIT, `vinodv@mit.edu`

Abstract. We propose a simple and general framework for constructing oblivious transfer (OT) protocols that are *efficient*, *universally composable*, and *generally realizable* under any one of a variety of standard number-theoretic assumptions, including the decisional Diffie-Hellman assumption, the quadratic residuosity and decisional composite residuosity assumptions, and *worst-case* lattice assumptions.

Our OT protocols are round-optimal (one message each way), quite efficient in computation and communication, and can use a single common string for an unbounded number of executions between the same sender and receiver. Furthermore, the protocols can provide *statistical* security to either the sender or the receiver, simply by changing the distribution of the common string. For certain instantiations of the protocol, even a common *uniformly random* string suffices.

Our key technical contribution is a simple abstraction that we call a *dual-mode* cryptosystem. We implement dual-mode cryptosystems by taking a unified view of several cryptosystems that have what we call "messy" public keys, whose defining property is that a ciphertext encrypted under such a key carries *no information* (statistically) about the encrypted message.

As a contribution of independent interest, we also provide a multi-bit *amortized* version of Regev's lattice-based cryptosystem (STOC 2005) whose time and space complexity are improved by a linear factor in the security parameter n. The resulting amortized encryption and decryption times are only $\tilde{O}(n)$ bit operations per message bit, and the ciphertext expansion can be made as small as a constant; the public key size and underlying lattice assumption remain essentially the same.

⋆ This material is based upon work supported by the National Science Foundation under Grants CNS-0716786 and CNS-0749931. Any opinions, findings, and conclusions or recommedations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

⋆⋆ Work performed while at SRI International.

⋆⋆⋆ Supported by NSF CNS-0749931, CNS-0524252, CNS-0716199; the US Army Research Office under the CyberTA Grant No. W911NF-06-1-0316; and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

1 Introduction

Oblivious transfer (OT), first proposed by Rabin [30], is a fundamental primitive in cryptography, especially for secure two-party and multiparty computation [34, 16]. Oblivious transfer allows one party, called the receiver, to obtain exactly one of two (or more) values from another other party, called the sender. The receiver remains oblivious to the other value(s), and the sender is oblivious to which value was received. Since its introduction, OT has received an enormous amount of attention in the cryptographic literature (see [23, 11, 13], among countless others).

OT protocols that are secure against semi-honest adversaries can be constructed from (enhanced) trapdoor permutations and made robust to malicious adversaries using zero-knowledge proofs for NP [16]; however, this general approach is quite inefficient and is generally not suitable for real-world applications.

For practical use, it is desirable to have efficient OT protocols based on concrete assumptions. Naor and Pinkas [28] and independently, Aiello, Ishai and Reingold [1] constructed efficient two-message protocols based on the decisional Diffie-Hellman (DDH) assumption. Abstracting their approach via the the projective hash framework of Cramer and Shoup [10], Tauman Kalai [22] presented analogous protocols based on the quadratic residuosity and decisional composite residuosity assumptions. The primary drawback of these constructions is that their security is proved only according to a "half-simulation" definition, where an ideal world simulator is shown only for a cheating receiver. Therefore, they are not necessarily secure when integrated into a larger protocol, such as a multiparty computation. Indeed, as pointed out in [28], such protocols may fall to selective-failure attacks, where the sender causes a failure that depends upon the receiver's selection.

Very recently (and independently of our work), Lindell [25] used a cut-and-choose technique (in lieu of general zero knowledge) to construct fully-simulatable OT protocols under the same set of assumptions as in [28, 1, 22]. For this level of security, Lindell's result is the most efficient protocol that has appeared (that does not rely on random oracles), yet it still adds a few communication rounds to prior protocols and amplifies their computational cost by a statistical security parameter (i.e., around 40 or so). There are also several recent works on constructing fully-simulatable protocols for other variants of OT, and on obtaining efficiency through the use of random oracles; see Section 1.3 for details.

We point out that all of the above fully-simulatable protocols are proved secure in the plain *stand-alone* model, which allows for secure sequential composition, but not necessarily *parallel* or *concurrent* composition. For multiparty computation or in complex environments like the Internet, composability can offer significant security and efficiency benefits (e.g., by saving rounds of communication through parallel execution).

In addition, while there is now a significant body of literature on constructing cryptographic primitives from *worst-case lattice assumptions* (e.g., [2, 3, 31, 26, 32, 29, 15]), little is known about obtaining oblivious transfer. Protocols for semi-honest adversaries can be easily inferred from the proof techniques of [3, 31, 32], and made robust to malicious adversaries by standard coin-flipping and zero-knowledge techniques. However, for efficiency it appears technically difficult to instantiate the projective hash framework for OT [10, 22] under lattice assumptions, so a different approach may be needed.

1.1 Our Approach

Our goal is to construct oblivious transfer protocols enjoying all of the following desirable properties:

1. **Secure and composable:** we seek OT protocols that are secure according to a full simulation-based definition, and that compose securely (e.g., in parallel) with each other and with other protocols.
2. **Efficient:** we desire protocols that are efficient in computation, communication, and usage of any external resources.
3. **Generally realizable:** we endeavor to design an *abstract framework* that is realizable under a variety of concrete assumptions, including those related to lattices. Such a framework would demonstrate the conceptual generality of the approach, and could protect against future advances in cryptanalysis, such as improved algorithms for specific problems or the development of practical quantum computers (which are known to defeat factoring- and discrete log-based cryptosystems [33], but not those based on lattices).

We present a simple and novel framework for reaching all three of the above goals, working in the universal composability (UC) model of Canetti [5] with *static* corruptions of parties.

Our protocol is based on a new abstraction that we call a *dual-mode* cryptosystem. The system is initialized in a setup phase that produces a common string CRS, which is made available to all parties (we discuss this setup assumption in more detail below). Depending on the instantiation, the common string may be uniformly random, or created according to some prescribed distribution.

Given a dual-mode cryptosystem, the concrete OT protocol is very simple: the receiver uses its selection bit (and the CRS) to generate a "base" public key and secret key, and delivers the public key to the sender. The sender computes two "derived" public keys (using the CRS), encrypts each of its values under the corresponding derived key, and sends the ciphertexts to the receiver. Finally, the receiver uses its secret key to decrypt the appropriate value. Note that the protocol is message-optimal (one in each direction), and that it is essentially as efficient as the underlying cryptosystem. The security of the protocol comes directly from the dual-mode properties, which we describe in more detail next.

Dual-mode cryptosystems. The setup phase for a dual-mode cryptosystem creates the CRS according to one of two chosen modes, which we call the *messy* mode and the *decryption* mode.

The system has three main security properties. In messy mode, for *every* (possibly malformed) base public key, at least one of the derived public keys hides its encrypted messages *statistically*. Moreover, the CRS is generated along with a "trapdoor" that makes it easy to determine (given the base public key) which of the derived keys does so. These properties imply *statistical* security against even an unbounded cheating receiver.

In decryption mode, the honest receiver's selection bit is likewise hidden *statistically* by its choice of base key. In addition, there is a (different) trapdoor for the CRS that makes it easy to generate a base public key together with *two* properly-distributed secret keys corresponding to each potential selection bit. This makes it possible to decrypt both of the sender's ciphertexts, and implies statistical security against even an unbounded cheating sender.

Finally, a dual-mode system has the property that messy mode and decryption mode are *computationally* indistinguishable; this is the only computational property in the definition. The OT protocol can therefore provide statistical security for *either* the sender or the receiver, depending on the chosen mode (we are not aware of any other OT protocol having this property). This also makes the security proof for the protocol quite modular and symmetric: computational security for a party (e.g., against a bounded cheating receiver in decryption mode) follows directly from statistical security in the other mode, and by the indistinguishability of modes.[3]

The dual-mode abstraction has a number of nice properties. First, the definition is quite simple: for any candidate construction, we need only demonstrate three simple properties (two of which are statistical). Second, the same CRS can be used for an *unbounded* number of OT executions between the same sender and receiver (we are not aware of any other OT protocol having this property). Third, we can efficiently realize the abstraction under any one of several standard assumptions, including the DDH assumption, the quadratic residuosity and decisional composite residuosity assumptions, and *worst-case* lattice assumptions (under a slight relaxation of the dual-mode definition).

Of course, the security of our protocol depends on a trusted setup of the CRS. We believe that in context, this assumption is reasonable (or even quite mild). First, it is known that OT in the plain UC model *requires* some type of trusted setup [6]. Second, as we have already mentioned, a single CRS suffices for any number of OT executions between the same sender and receiver. Third, several of our instantiations require only a common *uniformly random* string, which may be obtainable without relying on a trusted party via natural processes (e.g., sunspots).

1.2 Concrete Constructions

To construct dual-mode systems from various assumptions, we build upon several public-key cryptosystems that admit what we call *messy* public keys (short for "message-lossy;" these are also called "meaningless" keys in a recent work of Kol and Naor [24]). The defining property of a messy key is that a ciphertext

[3] Groth, Ostrovsky, and Sahai [19] used a similar "parameter-switching" argument in the context of non-interactive zero knowledge.

produced under it carries *no information* (statistically) about the encrypted message. More precisely, the encryptions of any two messages m_0, m_1 under a messy key are statistically close, over the randomness of the encryption algorithm. Prior cryptosytems based on lattices [3, 31, 32], Cocks' identity-based cryptosystem [9], and the original OT protocols of [28, 1] all rely on messy keys as a key conceptual tool in their security proofs.

Messy keys play a similarly important role in our dual-mode constructions. As in prior OT protocols [28, 1, 22], our constructions guarantee that for any base public key (the receiver's message), at least one of the derived keys is messy. A novel part of our constructions is in the use of a trapdoor for efficiently *identifying* messy keys (this is where the use of a CRS seems essential).

For our DDH- and DCR-based constructions, we obtain a dual-mode cryptosystem via relatively straightforward abstraction and modification of prior protocols (see Section 5). For quadratic residuosity, our techniques are quite different from those of [10, 22]; specifically, we build on a modification of Cocks' identity-based cryptosystem [9] (see Section 6). In both of these constructions, we have a precise characterization of messy keys and a trapdoor algorithm for identifying them.

Our lattice-based constructions are more subtle and technically involved. Our starting point is a cryptosytem of Regev [32], along with some recent "master trapdoor" techniques for lattices due to Gentry, Peikert, and Vaikuntanathan [15]. We do not have an *exact* characterization of messy keys for this cryptosystem, nor an error-free algorithm for identifying them. However, [15] presents a trapdoor algorithm that correctly identifies *almost every* public key as messy. By careful counting and a quantifier-switching argument, we show that for almost all choices of the CRS, *every* (potentially malformed) base public key has at least one derived key that is both messy and is correctly identified as such by the trapdoor algorithm.

As an additional contribution of independent interest, we give a multi-bit version of Regev's lattice-based cryptosystem [32] whose time and space efficiency are smaller by a linear factor in the security parameter n. The resulting system is very efficient (at least asymptotically): the amortized runtime per message bit is only $\tilde{O}(n)$ bit operations, and the ciphertext expansion is as small as a constant. The public key size and underlying lattice assumption are essentially the same as in [32]. Due to space constraints, we defer the exposition of our lattice-based constructions to the full version.

Our DDH and DCR constructions transfer (multi-bit) strings, while the QR and lattice constructions allow for essentially single-bit transfers. It is an interesting open question whether string OT can be achieved under the latter assumptions. Simple generalizations of our framework and constructions can also yield 1-out-of-k OT protocols.

1.3 Related Work

Under assumptions related to bilinear pairings, Camenisch, Neven, and shelat [4] and Green and Hohenberger [17] recently constructed fully-simulatable protocols in the plain stand-alone model (and the UC model, in [18]) for k-out-of-n *adaptive selection* OT, as introduced by Naor and Pinkas [27]. Adaptive selection can be useful for applications such as oblivious database retrieval.

Jarecki and Shmatikov [21] constructed UC-secure *committed string OT* under the decisional composite residuosity assumption, using a common *reference* string; their protocol is four rounds (or two in the random oracle model). Garay, MacKenzie, and Yang [14] constructed (enhanced) committed OT using a constant number of rounds assuming both the DDH and strong RSA assumptions. Damgard and Nielsen [12] constructed UC-secure OT against *adaptive corruptions*, under a public key infrastructure setup and the assumption that threshold homomorphic encryption exists.

There are other techniques for achieving efficiency in oblivious transfer protocols that are complementary to ours; for example, Ishai *et al.* [20] showed how to amortize k oblivious transfers (for some security parameter k) into many more, without much additional computation in the random oracle model.

2 Preliminaries

We let $\mathbb{N}$ denote the natural numbers. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. We let $\text{poly}(n)$ denote an unspecified function $f(n) = O(n^c)$ for some constant c. The security parameter will be denoted by n throughout the paper. We let $\mathsf{negl}(n)$ denote some unspecified function $f(n)$ such that $f = o(n^{-c})$ for *every* fixed constant c, saying that such a function is *negligible* (in n). We say that a probability is *overwhelming* if it is $1 - \mathsf{negl}(n)$.

2.1 The Universal Composability Framework (UC)

We work in the standard universal composability framework of Canetti [5] with *static* corruptions of parties. We use the definition of computational indistinguishability, denoted by $\stackrel{c}{\approx}$, from that work. The UC framework defines a probabilistic poly-time (PPT) *environment* machine $\mathcal{Z}$ that oversees the execution of a protocol in one of two worlds. The "ideal world" execution involves "dummy parties" (some of whom may be corrupted by an *ideal adversary* $\mathcal{S}$) interacting with a *functionality* $\mathcal{F}$. The "real world" execution involves PPT parties (some of whom may be corrupted by a PPT *real world adversary* $\mathcal{A}$) interacting only with each other in some protocol π. We refer to [5] for a detailed description of the executions, and a definition of the real world ensemble $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and the ideal world ensemble $\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. The notion of a protocol π *securely emulating* a functionality $\mathcal{F}$ is as follows:

Definition 1. *Let $\mathcal{F}$ be a functionality. A protocol π* UC-realizes *$\mathcal{F}$ if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$,*

$$\mathsf{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \stackrel{c}{\approx} \mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}.$$

The common reference string functionality $\mathcal{F}_{\text{CRS}}^{\mathcal{D}}$ produces a string with a fixed distribution that can be sampled by a PPT algorithm $\mathcal{D}$.

Oblivious Transfer (OT) is a two-party functionality, involving a sender $\mathbf{S}$ with input x_0, x_1 and a receiver $\mathbf{R}$ with an input $\sigma \in \{0,1\}$. The receiver $\mathbf{R}$ learns x_σ (and nothing else), and the sender $\mathbf{S}$ learns nothing at all. These requirements are captured by the specification of the OT functionality $\mathcal{F}_{\text{OT}}$ from [7], given in Figure 1.

Functionality $\mathcal{F}_{\text{OT}}$

$\mathcal{F}_{\text{OT}}$ interacts with a sender $\mathbf{S}$ and a receiver $\mathbf{R}$.

- Upon receiving a message $(\mathsf{sid}, \mathsf{sender}, x_0, x_1)$ from $\mathbf{S}$, where each $x_i \in \{0,1\}^\ell$, store (x_0, x_1) (The lengths of the strings ℓ is fixed and known to all parties).
- Upon receiving a message $(\mathsf{sid}, \mathsf{receiver}, \sigma)$ from $\mathbf{R}$, check if a $(\mathsf{sid}, \mathsf{sender}, \ldots)$ message was previously sent. If yes, send (sid, x_σ) to $\mathbf{R}$ and (sid) to the adversary $\mathcal{S}$ and halt. If not, send nothing to $\mathbf{R}$ (but continue running).

Fig. 1. The oblivious transfer functionality $\mathcal{F}_{\text{OT}}$ [7].

Our OT protocols operate in the common reference string model, or, in the terminology of [5], the $\mathcal{F}_{\text{CRS}}$-hybrid model. For efficiency, we would like to reuse the same common reference string for distinct invocations of oblivious transfer whenever possible. As described in [8], this can be achieved by designing a protocol for the *multi-session extension* $\hat{\mathcal{F}}_{\text{OT}}$ of the OT functionality $\mathcal{F}_{\text{OT}}$. Intuitively, $\hat{\mathcal{F}}_{\text{OT}}$ acts as a "wrapper" around any number of independent executions of $\mathcal{F}_{\text{OT}}$, and coordinates their interactions with the parties via subsessions (specified by a parameter ssid) of a single session (specified by sid).

The *UC theorem with joint state* (JUC theorem) [8] says that any protocol π operating in the $\mathcal{F}_{\text{OT}}$-hybrid model can be securely emulated in the real world by appropriately composing π with a *single* execution of a protocol ρ implementing $\hat{\mathcal{F}}_{\text{OT}}$. This single instance of ρ might use fewer resources (such as common reference strings) than several independent invocations of some other protocol that only realizes $\mathcal{F}_{\text{OT}}$; in fact, the protocols ρ that we specify will do exactly this.

3 Dual-Mode Encryption

Here we describe our new abstraction, called a *dual-mode* cryptosystem. It is initialized in a trusted setup phase, which produces a common string crs known to all parties along with some auxiliary "trapdoor" information t (which is only used in the security proof). The string crs may be either uniformly random or selected from a prescribed distribution, depending on the concrete instantiation. The cryptosystem can be set up in one of two modes, called *messy* mode and *decryption* mode. The first crucial security property of a dual-mode cryptosystem is that no (efficient) adversary can distinguish, given the crs, between the modes.

Once the system has been set up, it operates much like a standard public-key cryptosystem, but with an added notion that we call *encryption branches*. The key generation algorithm takes as a parameter a chosen *decryptable* branch $\sigma \in \{0, 1\}$, and the resulting secret key sk corresponds to branch σ of the public key pk. When encrypting a message under pk, the encrypter similarly specifies a branch $b \in \{0, 1\}$ on which to encrypt the message. Essentially, messages encrypted on branch $b = \sigma$ can be decrypted using sk, while those on the other branch cannot. Precisely what this latter condition means depends on the mode of the system.

When the system is in messy mode, branch $b \neq \sigma$ is what we call *messy*. That is, encrypting on branch b *loses all information* about the encrypted message — not only in the sense of semantic security, but even *statistically*. Moreover, the trapdoor for crs makes it easy to find a messy branch of *any* given public key, even a malformed one that could never have been produced by the key generator.

In decryption mode, the trapdoor circumvents the condition that only one branch is decryptable. Specifically, it allows the generation of a public key pk and corresponding secret keys sk_0, sk_1 that enable decryption on branches 0 and 1 (respectively). More precisely, for both values of $\sigma \in \{0, 1\}$, the distribution of the key pair (pk, sk_σ) is statistically close to that of an honestly-generated key pair with decryption branch σ.

We now proceed more formally. A dual-mode cryptosystem with message space $\{0, 1\}^\ell$ consists of a tuple of probabilistic algorithms having the following interfaces:

- $\mathsf{Setup}(1^n, \mu)$, given security parameter n and mode $\mu \in \{0, 1\}$, outputs (crs, t). The crs is a common string for the remaining algorithms, and t is a trapdoor value that enables either the $\mathsf{FindMessy}$ or $\mathsf{TrapKeyGen}$ algorithm, depending on the selected mode.

  For notational convenience, we define a separate messy mode setup algorithm $\mathsf{SetupMessy}(\cdot) := \mathsf{Setup}(\cdot, 0)$ and a decryption mode setup algorithm $\mathsf{SetupDec}(\cdot) := \mathsf{Setup}(\cdot, 1)$.

  All the remaining algorithms take crs as their first input, but for notational clarity, we usually omit it from their lists of arguments.
- $\mathsf{KeyGen}(\sigma)$, given a branch value $\sigma \in \{0, 1\}$, outputs (pk, sk) where pk is a public encryption key and sk is a corresponding secret decryption key for messages encrypted on branch σ.
- $\mathsf{Enc}(pk, b, m)$, given a public key pk, a branch value $b \in \{0, 1\}$, and a message $m \in \{0, 1\}^\ell$, outputs a ciphertext c encrypted on branch b.
- $\mathsf{Dec}(sk, c)$, given a secret key sk and a ciphertext c, outputs a message $m \in \{0, 1\}^\ell$.
- $\mathsf{FindMessy}(t, pk)$, given a trapdoor t and some (possibly even malformed) public key pk, outputs a branch value $b \in \{0, 1\}$ corresponding to a messy branch of pk.
- $\mathsf{TrapKeyGen}(t)$, given a trapdoor t, outputs (pk, sk_0, sk_1), where pk is a public encryption key and sk_0, sk_1 are corresponding secret decryption keys for branches 0 and 1, respectively.

Definition 2 (Dual-Mode Encryption). *A* dual-mode cryptosystem *is a tuple of algorithms described above that satisfy the following properties:*

1. ***Completeness for decryptable branch:*** *For every* $\mu \in \{0,1\}$, $(\mathsf{crs}, t) \leftarrow \mathsf{Setup}(1^n, \mu)$, $\sigma \in \{0,1\}$, $(pk, sk) \leftarrow \mathsf{KeyGen}(\sigma)$, *and* $m \in \{0,1\}^\ell$, *decryption is correct on branch* σ*:* $\mathsf{Dec}(sk, \mathsf{Enc}(pk, \sigma, m)) = m$.
2. ***Indistinguishability of modes:*** *the first outputs of* $\mathsf{SetupMessy}$ *and* $\mathsf{SetupDec}$ *are computationally indistinguishable, i.e.,* $\mathsf{SetupMessy}_1(1^n) \stackrel{c}{\approx} \mathsf{SetupDec}_1(1^n)$.
3. ***(Messy mode) Trapdoor identification of a messy branch:*** *For every* $(\mathsf{crs}, t) \leftarrow \mathsf{SetupMessy}(1^n)$ *and every (possibly malformed)* pk, $\mathsf{FindMessy}(t, pk)$ *outputs a branch value* $b \in \{0,1\}$ *such that* $\mathsf{Enc}(pk, b, \cdot)$ *is messy. Namely, for every* $m_0, m_1 \in \{0,1\}^\ell$, $\mathsf{Enc}(pk, b, m_0) \stackrel{s}{\approx} \mathsf{Enc}(pk, b, m_1)$.
4. ***(Decryption mode) Trapdoor generation of keys decryptable on both branches:*** *For every* $(\mathsf{crs}, t) \leftarrow \mathsf{SetupDec}(1^n)$, $\mathsf{TrapKeyGen}(t)$ *outputs* (pk, sk_0, sk_1) *such that for every* $\sigma \in \{0,1\}$, $(pk, sk_\sigma) \stackrel{s}{\approx} \mathsf{KeyGen}(\sigma)$.

It is straightforward to generalize these definitions to larger sets $\{0,1\}^k$ of branches, for $k > 1$ (in this generalization, $\mathsf{FindMessy}$ would return $2^k - 1$ different branches that are all messy). Such a dual-mode cryptosystem would yield a 1-out-of-2^k oblivious transfer in an analogous way. All of our constructions can be suitably modified to satisfy the generalized definition; for simplicity, we will concentrate on the branch set $\{0,1\}$ throughout the paper.

4 Oblivious Transfer Protocol

Here we construct a protocol dm that emulates the multi-session functionality $\hat{\mathcal{F}}_{\mathrm{OT}}$ functionality in the $\mathcal{F}_{\mathrm{CRS}}$-hybrid model. The dm protocol, which uses any dual-mode cryptosystem, is given in Figure 2.

The protocol can actually operate in either mode of the dual-mode cryptosystem, which affects only the distribution of the CRS that is used. In messy mode, the receiver's security is computational and the sender's security is *statistical*, i.e., security is guaranteed even against an *unbounded* cheating receiver. In decryption mode, the security properties are reversed.

To implement the two modes, we define two different instantiations of $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ that produce common strings according to the appropriate setup algorithm: $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{ext}}$ uses $\mathcal{D} = \mathsf{SetupMessy}_1$, and $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{dec}}$ uses $\mathcal{D} = \mathsf{SetupDec}_1$.

Theorem 3. *Let* $\mathsf{mode} \in \{\mathsf{ext}, \mathsf{dec}\}$. *Protocol* $\mathsf{dm}^{\mathsf{mode}}$ *securely realizes the functionality* $\hat{\mathcal{F}}_{OT}$ *in the* $\mathcal{F}_{CRS}^{\mathsf{mode}}$*-hybrid model, under* static *corruptions.*

For $\mathsf{mode} = \mathsf{ext}$*, the sender's security is statistical and the receiver's security is computational; for* $\mathsf{mode} = \mathsf{dec}$*, the security properties are reversed.*

Proof. Given all the properties of a dual-mode cryptosystem, the proof is conceptually quite straightforward. There is a direct correspondence between completeness and the case that neither party is corrupted, between messy mode

Protocol $\mathsf{dm}^{\mathsf{mode}}$ for Oblivious Transfer

The $\mathsf{dm}^{\mathsf{mode}}$ protocol is parameterized by $\mathsf{mode} \in \{\mathsf{ext}, \mathsf{dec}\}$ indicating the type of crs to be used.

Sender Input: $(\mathsf{sid}, \mathsf{ssid}, x_0, x_1)$, where $x_0, x_1 \in \{0,1\}^\ell$.
Receiver Input: $(\mathsf{sid}, \mathsf{ssid}, \sigma)$, where $\sigma \in \{0,1\}$.
When activated with their inputs, the sender $\mathbf{S}$ queries $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$ with $(\mathsf{sid}, \mathbf{S}, \mathbf{R})$ and gets back $(\mathsf{sid}, \mathsf{crs})$. The receiver $\mathbf{R}$ then queries $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$ with $(\mathsf{sid}, \mathbf{S}, \mathbf{R})$ and gets $(\mathsf{sid}, \mathsf{crs})$.
$\mathbf{R}$ computes $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, \sigma)$, sends $(\mathsf{sid}, \mathsf{ssid}, pk)$ to $\mathbf{S}$, and stores $(\mathsf{sid}, \mathsf{ssid}, sk)$.
$\mathbf{S}$ gets $(\mathsf{sid}, \mathsf{ssid}, pk)$ from $\mathbf{R}$, computes $y_b \leftarrow \mathsf{Enc}(pk, b, x_b)$ for each $b \in \{0,1\}$, and sends $(\mathsf{sid}, \mathsf{ssid}, y_0, y_1)$ to $\mathbf{R}$.
$\mathbf{R}$ gets $(\mathsf{sid}, \mathsf{ssid}, y_0, y_1)$ from $\mathbf{S}$ and outputs $(\mathsf{sid}, \mathsf{ssid}, \mathsf{Dec}(sk, y_\sigma))$, where $(\mathsf{sid}, \mathsf{ssid}, sk)$ was stored above.

Fig. 2. The protocol for realizing $\hat{\mathcal{F}}_{\mathrm{OT}}$.

and statistical security for the sender, and between decryption mode and statistical security for the receiver. The indinstinguishability of modes establishes computational security for the appropriate party in the protocol.

Let $\mathcal{A}$ be a static adversary that interacts with the parties $\mathbf{S}$ and $\mathbf{R}$ running the $\mathsf{dm}^{\mathsf{mode}}$ protocol. We construct an ideal world adversary (simulator) $\mathcal{S}$ interacting with the ideal functionality $\hat{\mathcal{F}}_{\mathrm{OT}}$, such that no environment $\mathcal{Z}$ can distinguish an interaction with $\mathcal{A}$ in the above protocol from an interaction with $\mathcal{S}$ in the ideal world. Recall that $\mathcal{S}$ interacts with both the ideal functionality $\hat{\mathcal{F}}_{\mathrm{OT}}$ and the environment $\mathcal{Z}$.

$\mathcal{S}$ starts by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with $\mathcal{Z}$ and the players $\mathbf{S}$ and $\mathbf{R}$. More specifically, $\mathcal{S}$ works as follows:

Simulating the communication with $\mathcal{Z}$: Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written into the adversary $\mathcal{A}$'s input tape (as if coming from $\mathcal{A}$'s environment). Every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}$'s own output tape (to be read by the environment $\mathcal{Z}$).

Simulating the case when only the receiver $\mathbf{R}$ *is corrupted:* Regardless of the mode of the protocol, $\mathcal{S}$ does the following. Run the messy mode setup algorithm, letting $(\mathsf{crs}, t) \leftarrow \mathsf{SetupMessy}(1^n)$. When the parties query the ideal functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$, return $(\mathsf{sid}, \mathsf{crs})$ to them. (Note that when $\mathsf{mode} = \mathsf{ext}$, the crs thus returned is identically distributed to the one returned by $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$, whereas when $\mathsf{mode} = \mathsf{dec}$, the simulated crs has a different distribution from the one returned by $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$ in the protocol).

When $\mathcal{A}$ produces a protocol message $(\mathsf{sid}, \mathsf{ssid}, pk)$, $\mathcal{S}$ lets $b \leftarrow \mathsf{FindMessy}(\mathsf{crs}, t, pk)$. $\mathcal{S}$ then sends $(\mathsf{sid}, \mathsf{ssid}, \mathsf{receiver}, 1-b)$ to the ideal functionality $\hat{\mathcal{F}}_{\mathrm{OT}}$, receives the output $(\mathsf{sid}, \mathsf{ssid}, x_{1-b})$, and stores it along with the value b.

When the dummy $\mathbf{S}$ is activated for subsession $(\mathsf{sid}, \mathsf{ssid})$, $\mathcal{S}$ looks up the corresponding b and x_{1-b}, computes $y_{1-b} \leftarrow \mathsf{Enc}(pk, 1-b, x_{1-b})$ and $y_b \leftarrow$

$\mathsf{Enc}(pk, b, 0^\ell)$ and sends the adversary $\mathcal{A}$ the message $(\mathsf{sid}, \mathsf{ssid}, y_0, y_1)$ as if it were from $\mathbf{S}$.

Simulating the case when only the sender $\mathbf{S}$ *is corrupted:* Regardless of the mode of the protocol, $\mathcal{S}$ does the following. Run the decryption mode setup algorithm, letting $(\mathsf{crs}, t) \leftarrow \mathsf{SetupDec}(1^n)$. When the parties query the ideal functionality $\mathcal{F}_{\text{CRS}}^{\mathsf{mode}}$, return $(\mathsf{sid}, \mathsf{crs})$ to them.

When the dummy $\mathbf{R}$ is activated on $(\mathsf{sid}, \mathsf{ssid})$, $\mathcal{S}$ computes $(pk, sk_0, sk_1) \leftarrow \mathsf{TrapKeyGen}(\mathsf{crs}, t)$, sends $(\mathsf{sid}, \mathsf{ssid}, pk)$ to $\mathcal{A}$ as if from $\mathbf{R}$, and stores $(\mathsf{sid}, \mathsf{ssid}, pk, sk_0, sk_1)$. When $\mathcal{A}$ replies with a message $(\mathsf{sid}, \mathsf{ssid}, y_0, y_1)$, $\mathcal{S}$ looks up the corresponding (pk, sk_0, sk_1), computes $x_b \leftarrow \mathsf{Dec}(sk_b, y_b)$ for each $b \in \{0, 1\}$ and sends to $\hat{\mathcal{F}}_{\text{OT}}$ the message $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sender}, x_0, x_1)$.

Simulating the remaining cases: When both parties are corrupted, the simulator $\mathcal{S}$ just runs $\mathcal{A}$ internally (who itself generates the messages from both $\mathbf{S}$ and $\mathbf{R}$).

When neither party is corrupted, $\mathcal{S}$ internally runs the honest $\mathbf{R}$ on input $(\mathsf{sid}, \mathsf{ssid}, \sigma = 0)$ and honest $\mathbf{S}$ on input $(\mathsf{sid}, \mathsf{ssid}, x_0 = 0^\ell, x_1 = 0^\ell)$, activating the appropriate algorithm when the corresponding dummy party is activated in the ideal execution, and delivering all messages between its internal $\mathbf{R}$ and $\mathbf{S}$ to $\mathcal{A}$.

We complete the proof using the following claims, which are straightforward to prove using the dual-mode properties (so we defer the proofs to the full version):

1. *(Statistical security for* $\mathbf{S}$ *in messy mode.)* When $\mathcal{A}$ corrupts the receiver $\mathbf{R}$,
$$\mathsf{IDEAL}_{\hat{\mathcal{F}}_{\text{OT}}, \mathcal{S}, \mathcal{Z}} \stackrel{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}}, \mathcal{A}, \mathcal{Z}}.$$
2. *(Statistical security for* $\mathbf{R}$ *in decryption mode.)* When $\mathcal{A}$ corrupts the sender $\mathbf{S}$,
$$\mathsf{IDEAL}_{\hat{\mathcal{F}}_{\text{OT}}, \mathcal{S}, \mathcal{Z}} \stackrel{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{dec}}, \mathcal{A}, \mathcal{Z}}.$$
3. *(Parameter switching.)* For any protocol π^{mode} in the $\mathcal{F}_{\text{CRS}}^{\mathsf{mode}}$-hybrid model, any adversary $\mathcal{A}$ and any environment $\mathcal{Z}$,
$$\mathsf{EXEC}_{\pi^{\mathsf{ext}}, \mathcal{A}, \mathcal{Z}} \stackrel{c}{\approx} \mathsf{EXEC}_{\pi^{\mathsf{dec}}, \mathcal{A}, \mathcal{Z}}.$$

We now complete the proof as follows. Consider the protocol $\mathsf{dm}^{\mathsf{ext}}$. When $\mathcal{A}$ corrupts $\mathbf{R}$, by item 1 above we have statistical security for $\mathbf{S}$ (whether or not $\mathbf{S}$ is corrupted). When $\mathcal{A}$ corrupts $\mathbf{S}$, by items 2 and 3 above we have

$$\mathsf{IDEAL}_{\hat{\mathcal{F}}_{\text{OT}}, \mathcal{S}, \mathcal{Z}} \stackrel{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{dec}}, \mathcal{A}, \mathcal{Z}} \stackrel{c}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}}, \mathcal{A}, \mathcal{Z}},$$

which implies computational security for $\mathbf{R}$.

It remains to show computational security when neither the sender nor the receiver is corrupted. Let $\mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}}, \mathcal{A}, \mathcal{Z}}(x_0, x_1, b)$ (resp, $\mathsf{EXEC}_{\mathsf{dm}^{\mathsf{dec}}, \mathcal{A}, \mathcal{Z}}(x_0, x_1, b)$) denote the output of an environment in the protocol $\mathsf{dm}^{\mathsf{ext}}$ (resp, $\mathsf{dm}^{\mathsf{dec}}$) that sets the inputs of the sender $\mathbf{S}$ to be (x_0, x_1) and the input of the receiver $\mathbf{R}$ to be the bit b. The following sequence of hybrids establishes what we want.

$$\mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}},\mathcal{A},\mathcal{Z}}(x_0, x_1, 1) \stackrel{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}},\mathcal{A},\mathcal{Z}}(0^\ell, x_1, 1) \stackrel{c}{\approx}$$
$$\mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}},\mathcal{A},\mathcal{Z}}(0^\ell, x_1, 0) \stackrel{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}},\mathcal{A},\mathcal{Z}}(0^\ell, 0^\ell, 0)$$

The first two and the last two experiments are statistically indistinguishable because of the messy property of encryption, and the second and third experiments are computationally indistinguishable because of the computational hiding of the receiver's selection bit. The first experiment corresponds to the real world execution, whereas the last experiment is what the simulator runs. Furthermore, by the completeness of the dual-mode cryptosystem, the first experiment is statistically indistinguishable from the ideal world exection with inputs (x_0, x_1, b).

The proof of security for protocol $\mathsf{dm}^{\mathsf{dec}}$ follows symmetrically, and we are done. $\square$

5 Realization from DDH

5.1 Background

Let $\mathcal{G}$ be a an algorithm that takes as input a security parameter 1^n and outputs a group description $\mathbb{G} = (G, p, g)$, where G is a cyclic group of prime order p and g is a generator of G.

Our construction will make use of groups for which the DDH problem is believed to be hard. The version of the DDH assumption we use is the following: for random *generators* $g, h \in G$ and for *distinct* but otherwise random $a, b \in \mathbb{Z}_p$, the tuples (g, h, g^a, h^a) and (g, h, g^a, h^b) are computationally indistinguishable.[4] This version of the DDH assumption is equivalent to another common form, namely, that $(g, g^a, g^b, g^{ab}) \stackrel{c}{\approx} (g, g^a, g^b, g^c)$ for independent $a, b, c \leftarrow Z_p$, because g^a is a generator and $c \neq ab$ with overwhelming probability.

5.2 Cryptosystem Based on DDH

We start by presenting a cryptosystem based on the hardness of the Decisional Diffie-Hellman problem, which slightly differs from the usual ElGamal cryptosystem in a few ways. The cryptosystem depends on a randomization procedure that we describe below. We note that the algorithm $\mathsf{Randomize}$ we describe below is implicit in the OT protocol of Naor and Pinkas [28].

Lemma 4 (Randomization). *Let G be an arbitrary multiplicative group of prime order p. For each $x \in \mathbb{Z}_p$, define $\text{DLOG}_G(x) = \{(g, g^x) : g \in G\}$. There is a probabilistic algorithm* $\mathsf{Randomize}$ *that takes generators $g, h \in G$ and elements $g', h' \in G$, and outputs a pair $(u, v) \in G^2$ such that:*

[4] To be completely formal, the respective ensembles of the two distributions, indexed by the security parameter n, are indistinguishable.

– *If $(g, g'), (h, h') \in \textsc{dlog}_G(x)$ for some x, then (u, v) is uniformly random in $\textsc{dlog}_G(x)$.*
– *If $(g, g') \in \textsc{dlog}_G(x)$ and $(h, h') \in \textsc{dlog}_G(y)$ for some $x \neq y$, then (u, v) is uniformly random in G^2.*

Proof. Define $\mathsf{Randomize}(g, h, g', h')$ to do the following: Choose $s, t \leftarrow \mathbb{Z}_p$ independently and let $u = g^s h^t$ and $v = (g')^s (h')^t$. Output (u, v).

Since g and h are generators of G, we can write $h = g^r$ for some nonzero $r \in \mathbb{Z}_p$. First suppose (g, g') and (h, h') belong to $\textsc{dlog}_G(x)$ for some x. Now, $u = g^s h^t = g^{s+rt}$ is uniformly random in G, since g is a generator of G and s is random in $\mathbb{Z}_p$. Furthermore, $v = (g')^s (h')^t = (g^s h^t)^x = u^x$ and thus, $(u, v) \in \textsc{dlog}_G(x)$.

Now suppose $(g, g') \in \textsc{dlog}_G(x)$ and $(h, h') \in \textsc{dlog}_G(y)$ for some $x \neq y$. Then $u = g^s h^t = g^{s+rt}$ and $v = g^{xs+ryt}$. Because $r(x - y) \neq 0 \in \mathbb{Z}_p$, the expressions $s + rt$ and $xs + ryt$ are linearly independent combinations of s and t. Therefore, over the choice of $s, t \in \mathbb{Z}_p$, u and v are uniform and independent in G.

We now describe the basic cryptosystem.

– $\mathsf{DDHKeyGen}(1^n)$: Choose $\mathbb{G} = (G, p, g) \leftarrow \mathcal{G}(1^n)$. The message space of the system is G.
  Choose another generator $h \leftarrow G$ and exponent $x \leftarrow \mathbb{Z}_p$. Let $pk = (g, h, g^x, h^x)$ and $sk = x$. Output (pk, sk).
– $\mathsf{DDHEnc}(pk, m)$: Parse pk as (g, h, g', h'). Let $(u, v) \leftarrow \mathsf{Randomize}(g, h, g', h')$. Output the ciphertext $(u, v \cdot m)$.
– $\mathsf{DDHDec}(sk, c)$: Parse c as (c_0, c_1). Output c_1 / c_0^{sk}.

Now consider a public key pk of the form (g, h, g^x, h^y) for distinct $x, y \in \mathbb{Z}_p$ (and where g, h are generators of G). It follows directly from Lemma 4 that $\mathsf{DDHEnc}(pk, \cdot)$ is messy. Namely, for every two messages $m_0, m_1 \in \mathbb{Z}_p$, $\mathsf{DDHEnc}(pk, m_0) \stackrel{s}{\approx} \mathsf{DDHEnc}(pk, m_1)$.

5.3 Dual-Mode Cryptosystem

We now construct a dual-mode encryption scheme based on the hardness of DDH.

– Both $\mathsf{SetupMessy}$ and $\mathsf{SetupDec}$ start by choosing $\mathbb{G} = (G, p, g) \leftarrow \mathcal{G}(1^n)$.
  $\mathsf{SetupMessy}(1^n)$: Choose random generators $g_0, g_1 \in G$. Choose *distinct nonzero* exponents $x_0, x_1 \leftarrow \mathbb{Z}_p$. Let $h_b = g_b^{x_b}$ for $b \in \{0, 1\}$. Let $\mathsf{crs} = (g_0, h_0, g_1, h_1)$ and $t = (x_0, x_1)$. Output (crs, t).
  $\mathsf{SetupDec}(1^n)$: Choose a random generator $g_0 \in G$, a random nonzero $y \in \mathbb{Z}_p$, and let $g_1 = g_0^y$. Choose a random nonzero exponent $x \in \mathbb{Z}_p$. Let $h_b = g_b^x$ for $b \in \{0, 1\}$, let $\mathsf{crs} = (g_0, h_0, g_1, h_1)$ and $t = y$. Output (crs, t).
  In the following, all algorithms are implicitly provided the crs and parse it as (g_0, h_0, g_1, h_1).

– $\mathsf{KeyGen}(\sigma)$: Choose $r \leftarrow \mathbb{Z}_p$. Let $g = g_\sigma^r$, $h = h_\sigma^r$ and $pk = (g, h)$. Let $sk = r$. Output (pk, sk).
– $\mathsf{Enc}(pk, b, m)$: Parse pk as (g, h). Let $pk_b = (g_b, h_b, g, h)$. Output $\mathsf{DDHEnc}(pk_b, m)$ as the encryption of m on branch b.
– $\mathsf{Dec}(sk, c)$: Output $\mathsf{DDHDec}(sk, c)$.
– $\mathsf{FindMessy}(t, pk)$: Parse the messy mode trapdoor t as (x_0, x_1) where $x_0 \neq x_1$. Parse the public key pk as (g, h). If $h \neq g^{x_0}$, then output $b = 0$ as a (candidate) messy branch. Otherwise, we have $h = g^{x_0} \neq g^{x_1}$ because $x_0 \neq x_1$, so output $b = 1$ as a (candidate) messy branch.
– $\mathsf{TrapKeyGen}(t)$: Parse the decryption mode trapdoor t as a nonzero $y \in \mathbb{Z}_p$. Pick a random $r \leftarrow \mathbb{Z}_p$ and compute $pk = (g_0^r, h_0^r)$ and output $(pk, r, r/y)$.

We remark that $\mathsf{SetupMessy}$ actually produces a crs that is statistically close to a common *random* (not reference) string, because it consists of four generators that do not comprise a DDH tuple.

Theorem 5. *The above scheme is a dual-mode cryptosystem, assuming that DDH is hard for $\mathcal{G}$.*

Proof. Completeness follows by inspection from the correctness of the basic DDH cryptosystem.

We now show indistinguishability of the two modes. In messy mode, $\mathsf{crs} = (g_0, h_0 = g_0^{x_0}, g_1, h_1 = g_1^{x_1})$, where g_0, g_1 are random generators of G and x_0, x_1 are distinct and nonzero in $\mathbb{Z}_p$. Let $a = \log_{g_0} g_1$, which is nonzero but otherwise uniform in $\mathbb{Z}_p$. Then $b = \log_{h_0}(h_1) = a \cdot x_1/x_0$ is nonzero and distinct from a, but otherwise uniform. Therefore crs is statistically close to a random DDH non-tuple (g_0, h_0, g_0^a, h_0^b), where $a, b \leftarrow \mathbb{Z}_p$ are distinct but otherwise uniform. Now in decryption mode, $\mathsf{crs} = (g_0, h_0 = g_0^x, g_1, h_1 = g_1^x)$, where x is nonzero and random in $\mathbb{Z}_p$. Since $\log_{h_0}(h_1) = \log_{g_0}(g_1) = y$ is nonzero and random in $\mathbb{Z}_p$, crs is statistically close to a random DDH tuple. Under the DDH assumption, the two modes are indistinguishable.

We now demonstrate identification of a messy branch. By inspection, $\mathsf{FindMessy}(t, pk)$ computes a branch b for which (g_b, h_b, g, h) (the key used when encrypting under pk on branch b) is not a DDH tuple. By Lemma 4, this b is therefore a messy branch.

We conclude with trapdoor key generation. Let $(\mathsf{crs}, y) \leftarrow \mathsf{SetupDec}(1^n)$. Note that crs is a DDH tuple of the form $(g_0, h_0 = g_0^x, g_1 = g_0^y, h_1 = g_1^x)$, where x and y are nonzero. $\mathsf{TrapKeyGen}(\mathsf{crs}, y)$ outputs $(pk, sk_0, sk_1) = ((g_0^r, h_0^r), r, r/y)$. The output of $\mathsf{KeyGen}(\sigma)$, on the other hand, is $((g_\sigma^r, h_\sigma^r), r)$. We will now show that (pk, sk_σ) and $\mathsf{KeyGen}(\sigma)$ are identically distributed.

Indeed, $(pk, sk_0) = (g_0^r, h_0^r, r)$ is identically distributed to $\mathsf{KeyGen}(0)$, by definition of KeyGen. By a renaming of variables letting $r = r'y$, we have that $(pk, sk_1) = (g_0^r, h_0^r, r/y)$ is identical to $(g_0^{r'y}, h_0^{r'y}, r') = (g_1^{r'}, h_1^{r'}, r')$, which is distributed identically to $\mathsf{KeyGen}(1)$, since $r' = r/y \in \mathbb{Z}_p$ is uniformly distributed.

Larger branch sets. We briefly outline how the dual-mode cryptosystem is modified for larger branch sets $\{0,1\}^k$. Essentially, the scheme involves k parallel and independent copies of the one above, but all using the same group $\mathbb{G}$. The encryption algorithm Enc computes a k-wise secret sharing of the message, and encrypts each share under the corresponding copy of the scheme. This ensures that decryption succeeds only for the one specific branch selected to be decryptable. The $\mathsf{FindMessy}$ algorithm includes a branch $b \in \{0,1\}^k$ in its output list of messy branches if *any* branch b_i is messy for its corresponding scheme.

6 Realization from QR

6.1 Cryptosystem Based on QR

We start by describing a (non-identity-based) variant of Cocks' cryptosystem [9], which is based on the conjectured hardness of the quadratic residuosity problem.

For $N \in \mathbb{N}$, let $\mathbb{J}_N$ denote the set of all $x \in \mathbb{Z}_N^*$ with Jacobi symbol 1. Let $\mathbb{QR}_N \subset \mathbb{J}_N$ denote the set of all quadratic residues (squares) in $\mathbb{Z}_N^*$. The message space is $\{\pm 1\}$. Let $\left(\frac{t}{N}\right)$ denote the Jacobi symbol of t in $\mathbb{Z}_N^*$.

- $\mathsf{CKeyGen}(1^n)$: Choose two random n-bit safe primes[5] p and q and let $N = pq$. Choose $r \leftarrow \mathbb{Z}_N^*$ and let $y \leftarrow r^2$. Let $pk = (N, y)$, and $sk = r$. Output (pk, sk).
- $\mathsf{CEnc}(pk, m)$: Parse pk as (N, y). Choose $s \leftarrow \mathbb{Z}_N^*$ at random such that $\left(\frac{s}{N}\right) = m$, and output $c = s + y/s$.
- $\mathsf{CDec}(sk, c)$: Output the Jacobi symbol of $c + 2 \cdot sk$.

The following lemma is implicit in the security proof from [9].

Lemma 6 (Messy Characterization). *Let N be a product of two random n-bit safe primes p and q, let $y \in \mathbb{Z}_N^*$ and let $pk = (N, y)$. If $y \notin \mathbb{QR}_N$, then $\mathsf{CEnc}(pk, \cdot)$ is messy. Namely, $\mathsf{CEnc}(pk, +1) \stackrel{s}{\approx} \mathsf{CEnc}(pk, -1)$.*

Proof. Consider the equation $c = s + y/s \bmod N$ in terms of s (for fixed c and y), and say $s = s_0$ is one of the solutions. Then we have $c = s_0 + y/s_0 \bmod p$ and $c = s_0 + y/s_0 \bmod q$. The other solutions are s_1, s_2, and s_3, where

$$\begin{array}{lll} s_1 = s_0 \bmod p & s_2 = y/s_0 \bmod p & s_3 = y/s_0 \bmod p \\ s_1 = y/s_0 \bmod q & s_2 = s_0 \bmod q & s_3 = y/s_0 \bmod q. \end{array}$$

If $y \notin \mathbb{QR}_N$, then at least one of $\alpha_1 = \left(\frac{y}{p}\right)$ or $\alpha_2 = \left(\frac{y}{q}\right)$ is -1. Then $\left(\frac{s_1}{N}\right) = \alpha_2\left(\frac{s_0}{N}\right)$, $\left(\frac{s_2}{N}\right) = \alpha_1\left(\frac{s_0}{N}\right)$ and $\left(\frac{s_3}{N}\right) = \alpha_1\alpha_2\left(\frac{s_0}{N}\right)$. Thus, two of $\left(\frac{s_i}{N}\right)$ are $+1$ and the other two are -1. It follows that c hides $\left(\frac{s}{N}\right)$ perfectly.

[5] Safe primes are primes p such that $\frac{p-1}{2}$ is also prime.

6.2 Dual-Mode Cryptosystem

We now describe a dual-mode cryptosystem that is based on the above cryptosystem.

- $\mathsf{SetupMessy}(1^n)$: Choose two random n-bit safe primes p and q and let $N = pq$. Choose $y \leftarrow \mathbb{J}_N \setminus \mathbb{QR}_N$. Let $\mathsf{crs} = (N, y)$, and $t = (p, q)$. Output (crs, t).
  $\mathsf{SetupDec}(1^n)$: Let $N = pq$ for random n-bit safe primes as above. Choose $s \leftarrow \mathbb{Z}_N^*$, and let $y = s^2 \bmod N$. Let $\mathsf{crs} = (N, y)$, amd $t = s$. Output (crs, t).
  In the following, all algorithms are implicitly provided the crs and parse it as (N, y), and all operations are performed in $\mathbb{Z}_N^*$.
- $\mathsf{KeyGen}(\sigma)$: Choose $r \leftarrow \mathbb{Z}_N^*$, and let $pk = r^2/y^\sigma$. Let $sk = r$. Output (pk, sk).
- $\mathsf{Enc}(pk, b, m)$: Let $pk_b = (N, pk \cdot y^b)$. Output $\mathsf{CEnc}(pk_b, m)$.
- $\mathsf{Dec}(sk, c)$: Output $\mathsf{CDec}(sk, c)$.
- $\mathsf{FindMessy}(t, pk)$: Parse the trapdoor t as (p, q) where $N = pq$. If $pk \in \mathbb{QR}_N$ (this can be checked efficiently using p and q), then output $b = 1$ as the (candidate) messy branch; otherwise, output $b = 0$.
- $\mathsf{TrapKeyGen}(t)$: Choose a random $r \leftarrow \mathbb{Z}_N^*$ and let $pk = r^2$ and $sk_b = r \cdot t^b$ for each $b \in \{0, 1\}$. Output (pk, sk_0, sk_1).

Theorem 7. *The above scheme is a dual-mode cryptosystem, assuming the hardness of the quadratic residuosity problem.*

Proof. We first show completeness. Say $(pk, sk) \leftarrow \mathsf{KeyGen}(\sigma)$. Thus, $pk = r^2 y^{-\sigma}$ for some r. $\mathsf{Enc}(pk, \sigma, m)$ runs $\mathsf{CEnc}(pk \cdot y^\sigma, m) = \mathsf{CEnc}(r^2, m)$. Thus, the public key used in the Cocks encryption algorithm is a quadratic residue. By the completeness of the Cocks cryptosystem, the decryption algorithm recovers m.

We now show indistinguishability of the two modes. In messy mode, $\mathsf{crs} = (N, y)$, where y is a uniform element in $\mathbb{J}_N \setminus \mathbb{QR}_N$. In decryption mode, $\mathsf{crs} = (N, y)$, where y is a uniform element in $\mathbb{QR}_N$. By the QR assumption, these are indistinguishable.

We now demonstrate identification of a messy branch. Let pk be the (possibly malformed) public key. Since $y \notin \mathbb{QR}_N$, either pk or $pk \cdot y$ is not a quadratic residue. Lemma 6 implies that one of the branches of pk is messy; it can be found using the factorization $t = (p, q)$ of N.

We conclude with trapdoor key generation. Let $y = t^2$. $\mathsf{TrapKeyGen}(\mathsf{crs}, t)$ outputs $(r^2, r, r \cdot t)$. The output of $\mathsf{KeyGen}(\sigma)$, on the other hand, is $(r^2 y^{-\sigma}, r)$. Now, $(pk, sk_0) = (r^2, r)$ is distributed identically to $\mathsf{KeyGen}(0)$, by definition of KeyGen. By a renaming of variables letting $r = r'/t$, we have $(pk, sk_1) = ((r')^2/t^2, r') = ((r')^2/y, r')$, which is distributed identically to $\mathsf{KeyGen}(1)$, since $r' = r/t \in \mathbb{Z}_N^*$ is uniformly distributed.

For larger branch sets $\{0, 1\}^k$, the scheme is modified in a manner similar to the one from Section 5, where all k parallel copies of the scheme use the same modulus N.

7 Acknowledgments

We thank Susan Hohenberger, Yuval Ishai, Oded Regev, and the anonymous reviewers for helpful comments on earlier drafts of this paper.

References

1. William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001.
2. Miklós Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.
3. Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293, 1997.
4. Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, pages 573–590, 2007.
5. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.
6. Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, pages 19–40, 2001.
7. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.
8. Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO*, pages 265–281, 2003.
9. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.
10. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.
11. Claude Crépeau. Equivalence between two flavours of oblivious transfers. In *CRYPTO*, pages 350–354, 1987.
12. Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO*, pages 247–264, 2003.
13. Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.
14. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Efficient and universally composable committed oblivious transfer and applications. In *TCC*, pages 297–316, 2004.
15. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008. To appear. Full version available at `http://eprint.iacr.org/2007/432`.
16. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
17. Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT*, pages 265–282, 2007.
18. Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. Cryptology ePrint Archive, Report 2008/163, 2008. `http://eprint.iacr.org/`.

19. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, pages 339–358, 2006.
20. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, pages 145–161, 2003.
21. Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, pages 97–114, 2007.
22. Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In *EUROCRYPT*, pages 78–95, 2005.
23. Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
24. Gillat Kol and Moni Naor. Cryptography and game theory: Designing protocols for exchanging information. In *TCC*, pages 320–339, 2008.
25. Yehuda Lindell. Efficient fully simulatable oblivious transfer. In *CT-RSA*, 2008.
26. Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.
27. Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO*, pages 573–590, 1999.
28. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
29. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, 2008. To appear. Full version available at `http://eprint.iacr.org/2007/279`.
30. Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Harvard University, 1981.
31. Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.
32. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.
33. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
34. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.