A Framework for Efficient and Composable Oblivious Transfer

Chris Peikert[*] SRI International

Vinod Vaikuntanathan MIT[†]

Brent Waters[‡] SRI International

February 18, 2008

Abstract

We propose a simple and general framework for constructing oblivious transfer (OT) protocols that are *efficient*, *universally composable*, and *generally realizable* from a variety of standard number-theoretic assumptions, including the decisional Diffie-Hellman assumption, the quadratic residuosity assumption, 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. Furthermore, the protocols can provide *statistical* security to either the sender or receiver, simply by changing the distribution of the common string. For certain instantiations of the protocol, even a common *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 version of Regev's lattice-based cryptosystem (STOC 2005) whose time and space efficiency are improved by a linear factor in the security parameter n. The amortized encryption and decryption time is 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.

1 Introduction

Oblivious transfer (OT), first proposed by Rabin [Rab81], is a fundamental primitive in cryptography, especially for secure two-party and multiparty computation [Yao86, GMW87]. 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

[*] 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.

sender is oblivious to which value was received. Since its introduction, OT has received considerable attention from the research community; see, e.g., [Kil88, Cré87, EGL85].

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 [GMW87]; however, this general approach is very inefficient and mainly of theoretical interest.

For practical use, it is desirable to have efficient OT protocols based on specific number-theoretic assumptions. Naor and Pinkas [NP01] and independently, Aiello, Ishai and Reingold [AIR01] 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 [CS02], Tauman Kalai [Kal05] presented analogous protocols based on the Nth residuosity assumption and the quadratic residuosity assumption (under the Extended Riemann Hypothesis). 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 [NP01], 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 [Lin08] used a cut-and-choose technique (in lieu of general zero knowledge) to construct a fully-simulatable OT protocol under the same set of assumptions as in [NP01, AIR01, Kal05]. 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 cost by a statistical security parameter (e.g., by a factor of 40 or so). There are also some 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 that constructs cryptographic primitives from *worst-case lattice assumptions* (e.g., [Ajt04, AD97, Reg04, MR07, Reg05, PW08, GPV08]), essentially nothing is known about obtaining efficient OT. In particular, it appears technically difficult to instantiate the projective hash framework [CS02] under such assumptions, so a different approach may be needed.

1.1 Our Approach

Our goal is to construct oblivious transfer protocols enjoying all three 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 illuminate 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 computing).

We present a simple and novel framework for reaching all three of the above goals, working within universal composability (UC) model of Canetti [Can01] (with static corruptions).

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.

The 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 value 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 consists of only two rounds (one message in each direction), and that it is essentially as efficient as the underlying cryptosystem. The security of the protocol comes directly from the properties of the dual-mode cryptosystem, 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* choice of the receiver's base public key, at least one of the sender's values is hidden *statistically* by the encryption. Moreover, the CRS is generated along with a "trapdoor" that makes it easy to determine (given the receiver's base public key) which of the values is so hidden. These properties imply *statistical* security (unconditionally) 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; note that 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 with this property). Computational security for the other party follows directly from statistical security in the other mode, and the indistinguishability of modes.[1]

The dual-mode abstraction has a number of nice properties. First, the definition is quite simple: for any candidate construction, we need only show three simple security 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 with this property).

[1] We note that Groth, Ostrovsky, and Sahai [GOS06] used a similar "parameter-switching" argument in the context of non-interactive zero knowledge.

Third, we can efficiently realize the dual-mode abstraction under several standard number-theoretic assumptions, including the DDH assumption, the quadratic residuosity assumption (without relying on the Extended Riemann Hypothesis), and standard 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 [CF01]. Second, as we have already mentioned, a single CRS suffices for any number of OT executions between the same two parties. Third, several of our instantiations require only a common *uniformly random* string, which may be obtainable without relying on a trusted party via, e.g., natural processes.

1.2 Concrete Constructions

To construct dual-mode systems from various assumptions, we build upon several prior public-key cryptosystems that admit what we call *messy* public keys (short for "message-lossy"). The defining property of a messy key is that a ciphertext 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. Prior cryptosytems based on lattices [AD97, Reg04, Reg05], Cocks' identity-based cryptosystem [Coc01], and the original OT protocols of [NP01, AIR01] all rely crucially on messy keys in their security proofs.

Messy keys play a similarly important role in our dual-mode constructions. As in prior OT protocols [NP01, AIR01, Kal05], our constructions guarantee that for any base 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-based construction, we obtain a dual-mode cryptosystem via relatively straightforward abstraction and modification of prior protocols. For quadratic residuosity, our techniques are different from those of [CS02, Kal05]; specifically, we use a modification of Cocks' identity-based cryptosystem [Coc01]. In both constructions, we have a precise characterization of messy keys and a trapdoor algorithm for identifying them.

Our lattice-based constructions are more technically involved. Our starting point is a cryptosytem of Regev [Reg05], along with some recent trapdoor techniques for lattices due to Gentry, Peikert, and Vaikuntanathan [GPV08]. We do not have an *exact* characterization of messy keys for the cryptosystem. However, [GPV08] gives a trapdoor algorithm that (correctly) identifies *almost all* public keys as messy. By a careful counting argument, we show that for almost all choices of the CRS, *every* base public key has at least one derived key that is correctly identified as messy by the trapdoor algorithm. See Section 7 for details.

Our DDH construction transfers strings, while the QR and lattice constructions essentially allow only for single-bit transfers. It is an interesting open question whether multi-bit OT can be achieved under the latter assumptions.

Finally, as an additional contribution of independent interest, we give a multi-bit version of Regev's lattice-based cryptosystem whose time and space efficiency are smaller by a linear factor in the security parameter n. The resulting system is very efficient (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 [Reg05].

1.3 Related Work

Using groups with efficient bilinear pairings, Camenisch, Neven, and shelat [CNS07] and Green and Hohenberger [GH07] recently proposed fully-simulatable protocols in the standard model for k-out-of-n OT with *adaptive selection*, as introduced by Naor and Pinkas [NP99]. Adaptive selection can be useful for applications such as oblivious database retrieval.

Jarecki and Shamtikov [JS07] constructed UC-secure *committed* OT for strings relying on the decisional composite residuosity assumption, in the common *reference* string model. Their protocol is four rounds (or two in the random oracle model). We also note that there are other techniques to achieving efficiency in oblivious transfer protocols that are complementary to ours. For example, Ishai *et al.* [IKNP03] show in the random oracle model how to extend k oblivious transfers (for some security parameter k) into many more, without much additional effort.

1.4 Organization

In Section 2 we give background that is relevant to the entire paper. In Section 3 we define our new dual-mode abstraction, and in Section 4 we present our OT protocol prove it secure in the UC framework. In Sections 5, 6, and 7 we instantiate the dual-mode abstraction under the DDH, QR, and lattice assumptions, respectively. We also present our more efficient lattice-based cryptosystem in Section 7.

2 Preliminaries

2.1 Notation

We let $\mathbb{N}$ denote the natural numbers. For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. We use standard O, Ω, o, and ω notation to classify the growth of functions. We say that $f = \tilde{O}(g)$ if $f = O(g \log^c n)$ for some fixed constant c. We let $\mathrm{poly}(n)$ 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 $1 - \mathsf{negl}(n)$.

2.2 The Universal Composability Framework (UC)

We work in the standard universal composability framework of Canetti [Can01] with static corruptions of parties. For consistency, we use the definition of computational indistinguishability, denoted by $\overset{c}{\approx}$, 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 [Can01] 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 2.1. Let $\mathcal{F}$ be a functionality. A protocol π is said to UC-realize $\mathcal{F}$ if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all environments $\mathcal{Z}$,

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

The common reference string functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ produces a string with a fixed distribution that can be sampled by a PPT algorithm $\mathcal{D}$. Its definition is given in Figure 1.

Functionality $\mathcal{F}_{\mathbf{CRS}}^{\mathcal{D}}$

$\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ runs with parties $P_1, \ldots, P_n$ and is parametrized by an algorithm $\mathcal{D}$.

- When receiving a message (sid, P_i, P_j) from P_i, let $\mathsf{crs} \leftarrow \mathcal{D}(1^n)$, send $(\mathsf{sid}, \mathsf{crs})$ to P_i and send (crs, P_i, P_j) to the adversary. Next, when receiving (sid, P_i, P_j) from P_j (and only P_j), send $(\mathsf{sid}, \mathsf{crs})$ to P_j and to the adversary, and halt.

Figure 1: The common reference string functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ [CR03].

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 $\mathcal{F}_{\mathrm{OT}}$ from [CLOS02], given in Figure 2

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

$\mathcal{F}_{\mathrm{OT}}$ 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).

Figure 2: The oblivious transfer functionality $\mathcal{F}_{\mathrm{OT}}$ [CLOS02].

Our OT protocols operate in the common reference string model, or, in the terminology of [Can01], the $\mathcal{F}_{\mathrm{CRS}}$-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 [CR03], this can be achieved by designing a protocol for the *multi-session extension* $\hat{\mathcal{F}}_{\mathrm{OT}}$ of the OT functionality $\mathcal{F}_{\mathrm{OT}}$. Intuitively, $\hat{\mathcal{F}}_{\mathrm{OT}}$ acts as a “wrapper” around any number of independent executions of $\mathcal{F}_{\mathrm{OT}}$, 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) [CR03] says that any protocol π operating in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model can be securely emulated in the real world by appropriately composing π with a *single* execution of a protocol ρ implementing $\hat{\mathcal{F}}_{\mathrm{OT}}$. 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 $\mathcal{F}_{\mathrm{OT}}$; 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 $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{FindMessy}, \mathsf{TrapKeyGen})$ 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 desired decryptable 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.

We now describe the required security properties.

Definition 3.1 (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 $\mu \in \{0,1\}$, every $(\mathsf{crs}, t) \leftarrow \mathsf{Setup}(1^n, \mu)$, every $\sigma \in \{0,1\}$, every $(pk, sk) \leftarrow \mathsf{KeyGen}(\sigma)$, and every $m \in \{0,1\}^\ell$, decryption is correct on branch σ, i.e., $\mathsf{Dec}(sk, \mathsf{Enc}(pk, \sigma, m)) = m$.

   It also suffices for decryption to be correct with overwhelming probability over the randomness of the entire experiment.

2. **Indistinguishability of modes:** the first outputs of $\mathsf{SetupMessy}$ and $\mathsf{SetupDec}$ are computationally indistinguishable, i.e., $\mathsf{SetupMessy}_1(1^n) \overset{c}{\approx} \mathsf{SetupDec}_1(1^n)$.

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 $m_0, m_1 \in \{0,1\}^\ell$, $\mathsf{Enc}(pk, b, m_0) \overset{s}{\approx} \mathsf{Enc}(pk, b, m_1)$.

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 $\sigma \in \{0,1\}$, $(pk, sk_\sigma) \overset{s}{\approx} \mathsf{KeyGen}(\sigma)$.

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, briefly noting inline how to generalize each construction.

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. Let $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{FindMessy}, \mathsf{TrapKeyGen})$ be a dual-mode cryptosystem. The dm protocol is given in Figure 3.

The protocol can actually operate in either mode of the dual-mode cryptosystem, which only affects 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 $\mathcal{D} = \mathsf{SetupDec}_1$.

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.

Figure 3: The protocol for realizing $\hat{\mathcal{F}}_{\mathrm{OT}}$.

Theorem 4.1. *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.*

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 and statistical security for the sender, and between decryption mode and statistical security for the receiver. The indinstinguishability of modes will establish 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 will 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 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 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 $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$, 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 $\hat{\mathcal{F}}_{\mathrm{OT}}$ 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 $\mathcal{A}$.

The proof will be completed using the following claims, which are proved below:

1. *(Claim 4.2, statistical security for* $\mathbf{S}$ *in messy mode.)* When $\mathcal{A}$ corrupts the receiver $\mathbf{R}$,
$$\mathsf{IDEAL}_{\hat{\mathcal{F}}_{\mathrm{OT}}, \mathcal{S}, \mathcal{Z}} \overset{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}}, \mathcal{A}, \mathcal{Z}}.$$

2. *(Claim 4.3, statistical security for* $\mathbf{R}$ *in decryption mode.)* When $\mathcal{A}$ corrupts the receiver $\mathbf{S}$,
$$\mathsf{IDEAL}_{\hat{\mathcal{F}}_{\mathrm{OT}}, \mathcal{S}, \mathcal{Z}} \overset{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{dec}}, \mathcal{A}, \mathcal{Z}}.$$

3. *(Claim 4.4, parameter switching.)* For any protocol π^{mode} in the $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{mode}}$-hybrid model, any adversary $\mathcal{A}$ and any environment $\mathcal{Z}$,
$$\mathsf{EXEC}_{\pi^{\mathsf{ext}}, \mathcal{A}, \mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\pi^{\mathsf{dec}}, \mathcal{A}, \mathcal{Z}}.$$

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

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

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.

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

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. □

It remains to prove the three claims made in the proof above.

Claim 4.2. *If the adversary $\mathcal{A}$ corrupts the receiver $\mathbf{R}$ in an execution of $\mathsf{dm}^{\mathsf{ext}}$, then we have*

$$\mathsf{IDEAL}_{\hat{\mathcal{F}}_{OT},\mathcal{S},\mathcal{Z}} \overset{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{ext}},\mathcal{A},\mathcal{Z}}.$$

Proof. The real world execution can be seen as a game that proceeds as follows, interacting with the environment $\mathcal{Z}(z)$: first, $\mathsf{crs} \leftarrow \mathsf{SetupMessy}_1(1^n)$. Then the environment arbitrarily schedules some number of subsessions, where in each subsession, $\mathcal{Z}$ chooses an arbitrary pk and arbitrary inputs (x_0, x_1) for the honest sender $\mathbf{S}$, who sends $y_b \leftarrow \mathsf{Enc}(\mathsf{crs}, pk, b, x_b)$ for each $b \in \{0, 1\}$ to $\mathcal{Z}$.

The ideal world execution proceeds similarly; however, first $(\mathsf{crs}, t) \leftarrow \mathsf{SetupMessy}(1^n)$ (but only crs is visible to $\mathcal{Z}$). Then the environment arbitrarily schedules subsessions, where in each subsession $\mathcal{Z}$ produces an arbitrary pk and arbitrary inputs (x_0, x_1) for the dummy $\mathbf{S}$. The simulator $\mathcal{S}$ runs $b \leftarrow \mathsf{FindMessy}(t, pk)$ and learns x_{1-b} from the functionality $\hat{\mathcal{F}}_{\mathrm{OT}}$. It then sends $y_b \leftarrow \mathsf{Enc}(\mathsf{crs}, pk, b, 0^\ell)$ and $y_{1-b} = \mathsf{Enc}(\mathsf{crs}, pk, 1 - b, x_{1-b})$ to $\mathcal{Z}$.

The only difference between the two games is therefore in y_b in each subsession. But by trapdoor identification of a messy branch, we have in the ideal game that $\mathsf{Enc}(\mathsf{crs}, pk, b, 0^\ell) \overset{s}{\approx} \mathsf{Enc}(\mathsf{crs}, pk, b, x_b)$. Therefore the two games are statistically indistinguishable. □

Claim 4.3. *If the adversary $\mathcal{A}$ corrupts the sender $\mathbf{S}$ in an execution of $\mathsf{dm}^{\mathsf{dec}}$, then we have*

$$\mathsf{IDEAL}_{\hat{\mathcal{F}}_{OT},\mathcal{S},\mathcal{Z}} \overset{s}{\approx} \mathsf{EXEC}_{\mathsf{dm}^{\mathsf{dec}},\mathcal{A},\mathcal{Z}}.$$

Proof. The real world execution can be seen as a game that proceeds as follows, interacting with the environment $\mathcal{Z}(z)$: first, $\mathsf{crs} \leftarrow \mathsf{SetupDec}_1(1^n)$. Then the environment arbitrarily schedules some number of subsessions. In each subsession, $\mathcal{Z}$ chooses an input σ for the honest $\mathbf{R}$, who generates $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, \sigma)$ and sends pk to $\mathcal{Z}$, then $\mathcal{Z}$ produces arbitrary (y_0, y_1) and the honest $\mathbf{R}$ outputs $\mathsf{Dec}(\mathsf{crs}, sk, y_\sigma)$.

The ideal world execution proceeds similarly; however, first $(\mathsf{crs}, t) \leftarrow \mathsf{SetupDec}(1^n)$ (but only crs is visible to $\mathcal{Z}$). Then the environment arbitrarily schedules subsessions, where in each subsession $\mathcal{Z}$ produces arbitrary σ (not known to $\mathcal{S}$), then $\mathcal{S}$ runs $(pk, sk_0, sk_1) \leftarrow \mathsf{TrapKeyGen}(t)$ and gives pk

to $\mathcal{Z}$, then $\mathcal{S}$ receives arbitrary (y_0, y_1) from $\mathcal{Z}$. The dummy $\mathbf{R}$ outputs the value $\mathsf{Dec}(\mathsf{crs}, sk_\sigma, y_\sigma)$, acquiring it from the functionality, which was provided the messages $x_b = \mathsf{Dec}(\mathsf{crs}, sk_b, y_b)$ by $\mathcal{S}$.

The only difference between the two games is therefore in the creation of the public and secret keys. However, by trapdoor key generation, $(pk, sk_\sigma) \overset{s}{\approx} \mathsf{KeyGen}(\mathsf{crs}, \sigma)$ for any value of crs generated by $\mathsf{SetupDec}$. Therefore the two games are statistically indistinguishable. □

Claim 4.4. *For any protocol π^{mode} in the $\mathcal{F}_{CRS}^{\mathsf{mode}}$-hybrid model, adversary $\mathcal{A}$ and environment $\mathcal{Z}$,*

$$\mathit{EXEC}_{\pi^{\mathsf{ext}}, \mathcal{A}, \mathcal{Z}} \overset{c}{\approx} \mathit{EXEC}_{\pi^{\mathsf{dec}}, \mathcal{A}, \mathcal{Z}}.$$

Proof. By the indistinguishability of modes in the dual-mode cryptosystem, the output of $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{ext}}$ and $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{dec}}$ are computationally indistinguishable. Environment $\mathcal{Z}$ running protocol π^{mode} can be seen as an efficient algorithm that receives a polynomial number of samples from either $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{ext}}$ or $\mathcal{F}_{\mathrm{CRS}}^{\mathsf{dec}}$. By a standard hybrid argument, the two executions are indistinguishable. □

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.[2] This version of the DDH assumption is equivalent to another common form, namely, that $(g, g^a, g^b, g^{ab}) \overset{c}{\approx} (g, g^a, g^b, g^c)$ 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 [NP01].

Lemma 5.1 (Randomization)**.** *Let G be an arbitrary multiplicative group of prime order p. For each $x \in \mathbb{Z}_p$, define $\textsc{dlog}_G(x) = \{(g, g^x) : g \in G\}$. 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:*

- *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.*

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

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 5.1 that $\mathsf{DDHEnc}(pk, \cdot)$ is messy. Namely, for every two messages $m_0, m_1 \in \mathbb{Z}_p$, $\mathsf{DDHEnc}(pk, m_0) \overset{s}{\approx} \mathsf{DDHEnc}(pk, m_1)$.

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 5.2. *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 5.1, 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 [Coc01], 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[3] 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 of the Cocks cryptosystem.

Lemma 6.1 (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 $\mathsf{CEnc}(pk, b, \cdot)$ is messy. Namely,*

$$\mathsf{CEnc}(pk, b, +1) \overset{s}{\approx} \mathsf{CEnc}(pk, b, -1).$$

Proof. 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. Consider the equation $c = s + y/s \bmod N$, and say 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

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

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 these are $+1$ and the other two are -1. It follows that c hides $\left(\frac{s}{N}\right)$ perfectly. □

6.2 Dual-Mode Cryptosystem

We now describe a dual-mode cryptosystem that is based on the Cocks 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 $b = 0$.

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

- $\mathsf{TrapKeyGen}(t)$: 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 6.2. *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 6.1 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 5.2, where all k parallel copies of the scheme use the same modulus N.

7 Realization from Lattices

Here we construct a dual-mode cryptosystem based on the hardness of the *learning with errors* (LWE) problem, which is implied by the *worst-case* hardness of standard lattice problems for *quantum* algorithms, as shown by Regev [Reg05]. We stress that although the underlying lattice assumption relates to quantum algorithms, our constructions are entirely classical.

As an independent contribution, we present a much more efficient version of Regev's CPA-secure cryptosystem [Reg05] based on LWE. Our cryptosystem encrypts an n-bit message at essentially the same cost (in both space and time) as a single-bit message in the system from [Reg05]. Specifically, the ciphertext expansion factor (the ratio of the ciphertext length to message length) can be made as small as $O(1)$, as opposed to $\tilde{O}(n)$. Our encryption and decryption algorithms require only $\tilde{O}(n)$ bit operations per encrypted bit, as opposed to $\tilde{O}(n^2)$. The overall public key size remains asymptotically the same at $\tilde{O}(n^2)$ bits; however, assuming a trusted source of public randomness, an optimization from [Reg05] allows the user-specific part of the public key to made only $\tilde{O}(n)$ bits, while ours remains $\tilde{O}(n^2)$ bits.

7.1 Background

We start by introducing the notation and computational problems that are relevant to this section, for the most part following [Reg05].

For any $x, y \in \mathbb{R}$ with $y > 0$ we define $x \bmod y$ to be $x - \lfloor x/y \rfloor y$. For $x \in \mathbb{R}$, $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$ denotes the nearest integer to x (with ties broken upward). We define $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, i.e., the group of reals $[0, 1)$ with modulo 1 addition.

Probability distributions. The *normal distribution* with mean 0 and variance σ^2 (or standard deviation σ) is the distribution on $\mathbb{R}$ having density function $\frac{1}{\sigma \cdot \sqrt{2\pi}} \exp(-x^2/2\sigma^2)$. It is a standard fact that the sum of two independent normal variables with mean 0 and variances σ_1^2 and σ_2^2 (respectively) is a normal variable with mean 0 and variance $\sigma_1^2 + \sigma_2^2$. We also need a standard tail inequality: a normal variable with variance σ^2 is within distance $t \cdot \sigma$ (i.e., t standard deviations) of its mean, except with probability at most $\frac{1}{t} \cdot \exp(-t^2/2)$. Finally, it is possible to efficiently sample from a normal variable to any desired level of accuracy.

For $\alpha \in \mathbb{R}^+$ we define Ψ_α to be the distribution on $\mathbb{T}$ of a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$, reduced modulo 1. For any probability distribution $\phi : \mathbb{T} \to \mathbb{R}^+$ and an integer $q \in \mathbb{Z}^+$ (often implicit) we define its *discretization* $\bar{\phi} : \mathbb{Z}_q \to \mathbb{R}^+$ to be the discrete distribution over $\mathbb{Z}_q$ of the random variable $\lfloor q \cdot X_\phi \rceil \bmod q$, where X_ϕ has distribution ϕ.

For an integer $q \geq 2$ and some probability distribution $\chi : \mathbb{Z}_q \to \mathbb{R}^+$, an integer dimension $n \in \mathbb{Z}^+$ and a vector $\mathbf{s} \in \mathbb{Z}_q^n$, define $A_{\mathbf{s},\chi}$ as the distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ of the variable $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ where $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ is uniform and $e \leftarrow \chi$ are independent, and all operations are performed in $\mathbb{Z}_q$.

Learning with errors (LWE). For an integer $q = q(n)$ and a distribution χ on $\mathbb{Z}_q$, the goal of the (average-case) *learning with errors* problem $\mathsf{LWE}_{q,\chi}$ is to distinguish (with nonnegligible probability) between an oracle that returns independent samples from $A_{\mathbf{s},\chi}$ for some uniform $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, and an oracle that returns independent samples from the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The conjectured hardness of LWE is parameterized chiefly by the dimension n. Therefore we let all other parameters (e.g., q, χ, and others) be functions of n, often omitting the explicit dependence for notational clarity.

Regev [Reg05] demonstrated that for certain moduli q and error distributions χ, $\mathsf{LWE}_{q,\chi}$ is as hard as solving several standard *worst-case* lattice problems *using a quantum algorithm.* We state a version of his result here:

Proposition 7.1 ([Reg05])**.** *Let $\alpha = \alpha(n) \in (0, 1)$ and let $q = q(n)$ be a* prime *such that $\alpha \cdot q > 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that solves $\mathsf{LWE}_{q,\bar{\Psi}_\alpha}$, then there exists an efficient* quantum *algorithm for solving the following worst-case lattice problems in the ℓ_2 norm:*

- *SIVP: In any lattice Λ of dimension n, find a set of n linearly independent lattice vectors of length within at most $\tilde{O}(n/\alpha)$ of optimal.*
- *GapSVP: In any lattice Λ of dimension n, approximate the length of a shortest nonzero lattice vector to within a $\tilde{O}(n/\alpha)$ factor.*

Peikert [Pei07] extended this result to hold for lattice problems in *any* ℓ_p norm, $p \geq 2$, for the same $\tilde{O}(n/\alpha)$ approximation factors.

We define our cryptosystems purely in relation to the LWE problem, without explicitly taking into account the connection to lattices (or their parameter restrictions). We then instantiate the parameters appropriately, invoking Proposition 7.1 to ensure security assuming the (quantum) hardness of lattice problems.

7.2 Efficient Cryptosystem Based on LWE

Our efficient cryptosystem closely resembles Regev's [Reg05]. In the original scheme, public keys consist of two components: a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ chosen uniformly at random, and a row vector $\mathbf{p} = \mathbf{s}^T\mathbf{A} + \mathbf{x} \in \mathbb{Z}_q^{1\times m}$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret key chosen uniformly at random and each x_i is chosen independently from the error distribution χ for $i \in [m]$. That is, the pairs $(\mathbf{a}_i, p_i)$ are independent samples from the LWE distribution $A_{\mathbf{s},\chi}$. A ciphertext encrypting a single bit $b \in \{0, 1\}$ is computed as $(\mathbf{u}, c) = (\mathbf{Ae}, \mathbf{pe} + b \cdot \lfloor q/2 \rfloor)$ for a vector $\mathbf{e} \in \{0, 1\}^m \subseteq \mathbb{Z}_q^m$ chosen uniformly at random. Decryption is performed by computing $c - \mathbf{s}^T\mathbf{u} \in \mathbb{Z}_q$, and testing whether it is closer to 0 or $\lfloor q/2 \rfloor$ modulo q. In the former case, the decrypted message is 0, otherwise it is 1.

A simple inspection of the algorithms reveals that a large fraction of the public key and ciphertext, as well as the encryption and decryption time, is devoted to the large matrix $\mathbf{A}$. Our main new idea is that the matrix $\mathbf{A}$ and the encryption randomness $\mathbf{e}$ can be *securely amortized* (reused) over as many as $\ell = O(n)$ different public key rows $\mathbf{p}_i$ (corresponding to independent secrets $\mathbf{s}_i$ and error vectors $\mathbf{x}_i$), without increasing the asymptotic complexity of the scheme. This yields essentially an n factor improvement in nearly all aspects of the system. In addition, we can save an additional logarithmic factor in the ciphertext expansion by taking the message symbols from a space $\mathbb{Z}_p$ slightly larger than $\{0, 1\}$ without affecting correctness, under a slightly stronger quantitative assumption on LWE (this idea was also used in [KTX07]).

We first describe the various parameters of the scheme and their roles, but defer giving concrete values until later. The message space is $\mathbb{Z}_p^\ell$ for some integers $p = \text{poly}(n) \geq 2$ and $\ell = \text{poly}(n) \geq 1$. Let $q = \text{poly}(n) > p$ be a prime modulus. Our public keys and ciphertexts consist of matrices and vectors over $\mathbb{Z}_q$. For every $v \in \mathbb{Z}_p$ (i.e., one entry of a message vector), define the "center" for v as $t(v) = \lfloor v \cdot \frac{q}{p} \rceil \in \mathbb{Z}_q$. Let χ denote an efficiently sampleable error distribution over $\mathbb{Z}_q$ (generally we take $\chi = \bar{\Psi}_\alpha$ for some α in order to apply Proposition 7.1, but other error distributions could potentially be used as well). All operations are performed over $\mathbb{Z}_q$.

- $\mathsf{LWEKeyGen}(1^n)$: Choose a secret key matrix $\mathbf{S} \leftarrow \mathbb{Z}_q^{n\times\ell}$ uniformly at random.

  To create the public key, choose a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ uniformly at random. Choose an error matrix $\mathbf{X} \in \mathbb{Z}_q^{\ell\times m}$ where each entry $x_{i,j} \leftarrow \chi$ is chosen independently from the error distribution χ. The public key consists of the matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{P} = \mathbf{S}^T\mathbf{A} + \mathbf{X} \in \mathbb{Z}_q^{\ell\times m}$.

  Note that the (i, j)th entry of $\mathbf{P}$ is $p_{i,j} = \langle \mathbf{a}_j, \mathbf{s}_i \rangle + x_{i,j}$, where $\mathbf{s}_i \in \mathbb{Z}_q^n$ is the (uniform and secret) ith column of $\mathbf{S}$, and $\mathbf{a}_j \in \mathbb{Z}_q^n$ is the (uniform and public) jth column of $\mathbf{A}$. That is, $(\mathbf{a}_j, p_{i,j})$ is a sample from the LWE distribution $A_{\mathbf{s}_i,\chi}$.

- $\mathsf{LWEEnc}(pk = (\mathbf{A}, \mathbf{P}), \mathbf{v})$: To encrypt a message $\mathbf{v} \in \mathbb{Z}_p^\ell$, define the "center" vector $\mathbf{t} = t(\mathbf{v}) \in \mathbb{Z}_q^\ell$ by applying $t(\cdot)$ coordinate-wise to $\mathbf{v}$. Choose a vector $\mathbf{e} \leftarrow \{0, 1\}^m \subset \mathbb{Z}_q^m$ uniformly at random. The ciphertext is the pair $(\mathbf{u}, \mathbf{c}) = (\mathbf{Ae}, \mathbf{Pe} + \mathbf{t}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$.

- $\mathsf{LWEDec}(sk = \mathbf{S}, (\mathbf{u}, \mathbf{c}))$: Compute $\mathbf{d} = \mathbf{c} - \mathbf{S}^T\mathbf{u} \in \mathbb{Z}_q^\ell$. Output the plaintext $\mathbf{v} \in \mathbb{Z}_p^\ell$, where each v_i is such that $d_i - t(v_i) \in \mathbb{Z}_q$ is closest to 0 mod q.

7.2.1 Some Supporting Lemmas

We now prove a few lemmas that will help establish correctness and security (for appropriate choices of parameters). At first, the reader may wish to just read the statements of the lemmas and then skip directly to Section 7.2.2, where we instantiate the parameters, analyze the efficiency, and prove security.

Lemma 7.2 (Completeness)**.** *Let $q \geq 4pm$, let $\alpha \leq 1/(p \cdot \sqrt{m} \cdot g(n))$ for any $g(n) = \omega(\sqrt{\log n})$, and let $\chi = \bar{\Psi}_\alpha$. Then* LWEDec *decrypts correctly with overwhelming probability, over the random choice of $\mathbf{X}$ by* LWEKeyGen *(and for any possible random choices of the* LWEEnc *and* LWEDec*).*

Proof. The proof essentially follows from the exponential tail bound on a sum of (rounded-off) Gaussians.

Consider some secret key $\mathbf{S}$ and associated public key $(\mathbf{A}, \mathbf{P} = \mathbf{S}^T\mathbf{A} + \mathbf{X})$ where $\mathbf{A}$ and $\mathbf{S}$ are arbitary, and $\mathbf{X}$ is chosen according to the prescribed distribution. Now consider a ciphertext $(\mathbf{u}, \mathbf{c}) = (\mathbf{Ae}, \mathbf{Pe} + \mathbf{t})$ generated by LWEEnc using some $\mathbf{e} \in \{0,1\}^m$, where $\mathbf{t} = t(\mathbf{v})$ is the offset vector for the message $\mathbf{v}$. The decryption algorithm LWEDec computes the vector

$$\mathbf{d} = \mathbf{c} - \mathbf{S}^T\mathbf{u} = (\mathbf{S}^T\mathbf{A} + \mathbf{X})\mathbf{e} + \mathbf{t} - \mathbf{S}^T\mathbf{Ae} = \mathbf{Xe} + \mathbf{t} \in \mathbb{Z}_q^\ell.$$

Now consider any coordinate $j \in [\ell]$. The distance from d_j to t_j (modulo q) is determined by the value $(\mathbf{Xe})_j$. Therefore it suffices to show that for every j, $(\mathbf{Xe})_j$ is at distance at most $q/4p$ from 0 (modulo q) with overwhelming probability, because this guarantees that d_j is closest to $t(v_j)$.

Now by definition of $\chi = \bar{\Psi}_\alpha$, $\mathbf{X} = \lfloor q \cdot \mathbf{Y} \rceil \bmod q$, where each $y_{i,j}$ is an independent normal variable with mean 0 and variance α^2. Then by the triangle inequality, $(\mathbf{Xe})_j$ is at most $m/2 \leq q/8p$ away from $q(\mathbf{Ye})_j$, modulo q. Therefore it suffices to show that $|(\mathbf{Ye})_j| \leq 1/8p$ with overwhelming probability.

Because the $y_{i,j}$ are independent, $(\mathbf{Ye})_j$ is distributed as a normal variable with mean 0 and standard deviation $\|\mathbf{e}\| \cdot \alpha \leq \sqrt{m} \cdot \alpha \leq 1/(p \cdot g(n))$. Therefore by the tail inequality on normal variables,

$$\Pr_{\mathbf{Y}}[|(\mathbf{Ye})_j| > 1/8p] \leq \tfrac{8}{g(n)} \cdot \exp(-g(n)^2/128).$$

Because $g(n) = \omega(\sqrt{\log n})$, this probability is negligible, and by a union bound over all $j \in [m]$, we are done. □

Lemma 7.3 (Pseudorandom public keys)**.** *The distribution of the public key $pk = (\mathbf{A}, \mathbf{P})$ generated by* LWEKeyGen *is computationally indistinguishable from uniform over $\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^{\ell \times m}$, assuming $LWE_{q,\chi}$ is hard.*

Proof. The proof is virtually identical to one from [PW08] for a similar statement. It amounts to showing that LWE remains hard even when reusing the same public $\mathbf{a}_j$ vectors with multiple independent secrets $\mathbf{s}_i$ (and independent error terms for every sample). Consider hybrid distributions $H_0, \ldots, H_\ell$ for the pair $(\mathbf{A}, \mathbf{P})$: in distribution H_k, the entire matrix $\mathbf{A}$ and the first k rows of $\mathbf{P}$ are all uniform, while the remaining rows of $\mathbf{P}$ are chosen exactly according to LWEKeyGen, using independent secrets $\mathbf{s}_i$ and error terms $x_{i,j}$ for all $i > k$ and all $j \in [m]$. Then H_0 is the distribution generated by LWEKeyGen, and H_ℓ is completely uniform.

We now show that distributions H_{k-1} and H_k are computationally indistinguishable (assuming $\mathsf{LWE}_{q,\chi}$ is hard), from which the lemma follows. Consider a simulator $\mathcal{S}^\cdot$, which is given an oracle $\mathcal{O}$ that returns samples either from $A_{\mathbf{s},\chi}$ (for some secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ chosen uniformly at random) or from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. First, $\mathcal{S}$ makes m queries to $\mathcal{O}$, yielding samples $(\mathbf{a}_j, b_j)$ for $j \in [m]$. Then for all $i > k$ and $j \in [m]$, $\mathcal{S}$ chooses independent $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ and error terms $x_{i,j} \leftarrow \chi$. $\mathcal{S}$ outputs the pair $(\mathbf{A}, \mathbf{P})$ constructed in the following way: the jth column $\mathbf{a}_j$ of

$\mathbf{A}$ is the vector $\mathbf{a}_j$, the first $k-1$ rows of $\mathbf{P}$ are uniform, the kth row of $\mathbf{P}$ consists of the entries $p_{k,j} = b_j$, and the remaining entries of $\mathbf{P}$ are $p_{i,j} = \mathbf{s}_i^T \mathbf{a}_j + x_{i,j}$ for $i > k$ and $j \in [m]$.

It is apparent that if $\mathcal{O}$ samples from $A_{\mathbf{s},\chi}$, then $\mathcal{S}$'s output is distributed according to H_{k-1}, whereas if $\mathcal{O}$ samples from the uniform distribution, $\mathcal{S}$'s output is distributed according to H_k. It follows that H_{k-1} and H_k are indistinguishable, and we are done. □

Messy public keys. For an arbitrary fixed public key $pk = (\mathbf{A}, \mathbf{P})$, define $\delta(pk)$ to be the statistical distance between the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$ and the distribution of $(\mathbf{Ae}, \mathbf{Pe})$, where $\mathbf{e} \leftarrow \{0,1\}^m$ is chosen uniformly at random. Then for any two messages $\mathbf{v}_0, \mathbf{v}_1 \in \mathbb{Z}_q^\ell$,

$$\Delta(\mathsf{LWEEnc}(pk, \mathbf{v}_0), \mathsf{LWEEnc}(pk, \mathbf{v}_1)) \leq 2\delta(pk),$$

because both distributions are within $\delta(pk)$ of uniform. (Of course, the correctness of LWEDec implies that the public keys pk generated by $\mathsf{LWEKeyGen}$ have large $\delta(pk)$.)

The following lemma (due to [Reg05], inheriting from ideas in [Ajt04, IZ89]), shows that for large enough m, a public key $(\mathbf{A}, \mathbf{P})$ chosen *uniformly* at random is very likely to be messy. We include a proof for completeness.

Lemma 7.4 (Most keys are messy)**.** *Let $m \geq 3(n+\ell)\lg q$. Then we have*

$$\Pr_{pk}\left[\delta(pk) > q^{-(n+\ell)/2}\right] \leq 1/q^{n+\ell},$$

where the probability is taken over $pk = (\mathbf{A}, \mathbf{P}) \leftarrow \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{\ell \times m}$ chosen uniformly at random. In particular, $\delta(pk)$ is exponentially small in n, except with exponentially-small probability.

Proof. Let $G = \mathbb{Z}_q^{n+\ell}$ be the finite abelian group of size $|G| = q^{n+\ell}$, from which each pair of columns $(\mathbf{a}_i, \mathbf{p}_i)$ is chosen uniformly at random. For some arbitrary $pk = (\mathbf{A}, \mathbf{P})$, let D_{pk} be the distribution of $(\mathbf{Ae}, \mathbf{Pe})$, where $\mathbf{e} \leftarrow \{0,1\}^m$ is chosen uniformly at random. More precisely, for each $g \in G$,

$$D_{pk}(g) = \frac{1}{2^m} \left|\{\mathbf{e} \in \{0,1\}^m \ : \ (\mathbf{Ae}, \mathbf{Pe}) = g\}\right|.$$

We now show that (over a random choice of pk) the squared ℓ_2 norm of this distributon is likely to be very close to $1/|G|$, from which it will follow that the distibution is close to uniform. The squared ℓ_2 norm $\|D_{pk}\|^2$ of D_{pk} is

$$\begin{aligned} \|D_{pk}\|^2 = \sum_{g\in G} D_{pk}(g)^2 &= \Pr_{\mathbf{e},\mathbf{e}'}\left[(\mathbf{Ae}, \mathbf{Pe}) = (\mathbf{Ae}', \mathbf{Pe}')\right] \\ &\leq \frac{1}{2^m} + \Pr_{\mathbf{e},\mathbf{e}'}\left[(\mathbf{Ae}, \mathbf{P}, \mathbf{e}) = (\mathbf{Ae}', \mathbf{Pe}') \mid \mathbf{e} \neq \mathbf{e}'\right] \end{aligned}$$

Taking the expectation over a uniformly random pk and using the fact that for any *fixed* $\mathbf{e} \neq \mathbf{e}'$, $\Pr_{(\mathbf{A},\mathbf{P})}[(\mathbf{Ae}, \mathbf{Pe}) = (\mathbf{Ae}', \mathbf{Pe}')] = 1/|G|$, we obtain

$$\mathop{\mathrm{E}}_{pk}\left[\|D_{pk}\|^2\right] \leq \frac{1}{2^m} + \frac{1}{|G|}.$$

Note that $\|D_{pk}\|^2 \geq 1/|G|$ because D_{pk} is a probability distribution. Then by Markov's inequality, for any $K > 0$ we have $\|D_{pk}\|^2 - 1/|G| \leq K/2^m$ except with probability at most $1/K$ over the choice of pk. For such pk, the statistical distance between D_{pk} and uniform is

$$\begin{aligned}\sum\nolimits_{g\in G} |D_{pk}(g) - 1/|G|| &\leq \sqrt{|G|}\left(\sum\nolimits_{g\in G}(D_{pk}(g) - 1/|G|)^2\right)^{1/2} \\ &= \sqrt{|G|}\left(\|D_{pk}\|^2 - 1/|G|\right)^{1/2} \\ &\leq \sqrt{|G|}\cdot(K/2^m)^{1/2}.\end{aligned}$$

Setting $K = q^{n+\ell}$, and using $m \geq 3(n+\ell)\lg q$, $|G| = q^{n+\ell}$, the claim follows. □

7.2.2 Instantiation and Analysis

We now instantiate all the parameters of our LWE cryptosystem to ensure correctness and security. Other instantiations are also possible, yielding tradeoffs between efficiency and the strength of the underlying lattice assumptions. We have not attempted to optimize constant factors.

Recall that $\mathbb{Z}_p$ is the message space. Let $p = n^c$ for some positive $c = c(n)$ that is bounded above by a constant (e.g., c may itself be a constant, or a decreasing function of n, such as $c(n) = 1/\lg n$). For concreteness, let the amortization factor $\ell = n$ (any $\ell = \Theta(n)$ would also work). Let $m = (12+6c)n\lg n = O(n\lg n)$. Let q be a prime in $[10, 20]\cdot pm\lg n = \tilde{O}(n^{c+1})$. Finally, let the error distribution $\chi = \bar{\Psi}_\alpha$ for $\alpha = 1/(p\cdot\sqrt{m}\cdot\lg n) = 1/\tilde{O}(n^{c+1/2})$.

We first analyze the efficiency of the system. For concreteness, say that $c > 0$ is a constant. Then with the parameters above, a public key contains $2mn$ elements of $\mathbb{Z}_q$, for a total size of $\tilde{O}(n^2)$ bits. The message space is $\mathbb{Z}_p^n$, so to encrypt $n\lg p = \Theta(n\lg n)$ bits requires $O(mn)$ operations in $\mathbb{Z}_q$, costing $O(n^2\lg^2 n)$ bit operations in total, i.e., $O(n\lg n)$ bit operations per message bit. The ciphertext contains $2n$ elements of $\mathbb{Z}_q$, so it is only an $O(1)$ factor larger than the message.

We now prove that the system is secure under appropriate lattice assumptions.

Theorem 7.5. *For the parameters described above, our LWE cryptosystem is secure under chosen plaintext attack, unless SIVP and GapSVP are easy for quantum algorithms to approximate to within some $\tilde{O}(n^{c+3/2})$ factor.*

Proof. We first show completeness. By Lemma 7.2, LWEDec decrypts correctly (with overwhelming probability) for our choice of α, as long as $q \geq 4pm$. Indeed, $4pm < 10pm\lg n \leq q$.

We now demonstrate security. By Lemma 7.3, a public key pk generated by $\mathsf{LWEKeyGen}$ is indistinguishable from a truly uniform public key, assuming $\mathsf{LWE}_{q,\chi}$ is hard. Now observe that for all sufficiently large n,

$$3(n+\ell)\lg q \leq 6n\lg(n^{c+2}) = (12+6c)n\lg n \leq m.$$

Then by Lemma 7.4, no adversary (even an unbounded one) has advantage more than $2q^{-n}+q^{-2n} = \mathsf{negl}(n)$ in a chosen-plaintext attack when pk is chosen uniformly. Therefore, any adversary having a non-negligible advantage when pk is chosen according to $\mathsf{LWEKeyGen}$ can also solve $\mathsf{LWE}_{q,\chi}$.

We conclude by justifying the hardness of $\mathsf{LWE}_{q,\chi}$. Using the fact that $q \geq 4pm\cdot\lg n$, we have $q\alpha \geq 4\sqrt{m} > 2\sqrt{n}$, as required by Proposition 7.1. Therefore $\mathsf{LWE}_{q,\chi}$ is hard unless approximating SIVP and GapSVP to within some $\tilde{O}(n/\alpha) = \tilde{O}(n^{c+3/2})$ factor is easy for quantum algorithms. □

7.3 Dual-Mode Cryptosystem

Here we construct a dual-mode cryptosystem based on the hardness of LWE. We do not know how to construct a scheme that *exactly* satisfies Definition 3.1; however, we can construct one that satisfies a slightly relaxed definition, which suffices for running a *bounded* number of oblivious transfers for a single common string. Details follow.

Remarks on decryption mode. We relax trapdoor key generation in decryption mode (Property 4 of Definition 3.1). In the relaxed definition, the keypairs generated by $\mathsf{TrapKeyGen}$ need only be *computationally* indistinguishable from those generated by KeyGen. Moreover, the indistinguishable distributions are defined over the *entire* experiment of creating a crs and several keypairs, rather than for *every* choice of crs generated by $\mathsf{SetupDec}$ (as in the original Definition 3.1). This is the reason why the number of executions per crs is bounded by a predetermined quantity.

More precisely, let $\ell = \text{poly}(n)$ be some fixed polynomial in the security parameter n. We add an additional parameter $i \in [\ell]$ to the inputs of all the algorithms (except the Setup algorithms), which is used to specify their ith executions with a certain crs. We then relax Property 4 of Definition 3.1 to the following:

4′. Let $(\mathsf{crs}, t) \leftarrow \mathsf{SetupDec}(1^n)$ and $(pk^{(i)}, sk_0^{(i)}, sk_1^{(i)}) \leftarrow \mathsf{TrapKeyGen}(t, i)$ for each $i \in [\ell]$. Then for every $\sigma_i \in \{0,1\}$ for $i \in [\ell]$, we require

$$(\mathsf{crs}, (pk^{(1)}, sk^{(1)}_{\sigma_1}), \ldots, (pk^{(\ell)}, sk^{(\ell)}_{\sigma_\ell})) \overset{c}{\approx} (\mathsf{SetupDec}_1(1^n), \mathsf{KeyGen}(\sigma_1, 1), \ldots, \mathsf{KeyGen}(\sigma_\ell, \ell)).$$

Our dm protocol (Figure 3) for emulating the multi-session OT functionality $\hat{\mathcal{F}}_{\text{OT}}$ and its proof of security from Section 4 can be easily modified to use this relaxed definition. The protocol simply limits itself to ℓ separate uses of a single crs, then it starts over with a new crs by invoking another copy of the $\mathcal{F}_{\text{CRS}}$ functionality. The proof of security follows similarly, though it provides only computational security for both sender and receiver in decryption mode. It is therefore preferable in the OT protocol to generate the crs in messy mode, which in our construction will also have the advantage of being a *uniformly random* (not reference) string.

Remarks on messy mode. We also slightly relax the identification of a messy branch in messy mode (Property 3 of Definition 3.1). Instead of quantifying over *all* $(\mathsf{crs}, t) \leftarrow \mathsf{SetupMessy}(1^n)$, we require only that $\mathsf{FindMessy}$ work for an overwhelming fraction of such (crs, t). Furthermore, we require only that $\mathsf{FindMessy}$ is correct with overwhelming probability over its own randomness, i.e., if a branch b is *not* messy, then $\mathsf{FindMessy}$ outputs b with negligible probability. These relaxations do not affect the security proof, because the failure events are negligible. We stress that $\mathsf{FindMessy}$ still must find a messy branch for *every* (possibly malformed) public key pk, because pk is generated by a possibly malicious receiver.

To implement $\mathsf{FindMessy}$ efficiently, our construction relies on recent techniques of Gentry, Peikert, and Vaikuntanathan [GPV08]. Among other things, they constructed an LWE-based cryptosystem in which the public matrix $\mathbf{A}$ is shared among all users, and showed how to securely embed a trapdoor in $\mathbf{A}$ so that an efficient algorithm (called $\mathsf{IsMessy}$) can identify messy public keys given the trapdoor.

The cryptosystem and $\mathsf{IsMessy}$ algorithm given in [GPV08, Section 6] are somewhat subtle, so we summarize their main features here (precise claims are given in Proposition 7.8). The cryptosystem is a slight variant of Regev's original system [Reg05]. First, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ is chosen under a distribution that is statistically close to uniform, together with a trapdoor (called $\mathbf{S}$ in that work). The matrix $\mathbf{A}$ is fixed for all users of the system. Just as in our construction, the key generator $\mathsf{LWEKeyGen}$ generates a public key $\mathbf{p} = \mathbf{s}^T\mathbf{A} + \mathbf{x}$, where $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret key chosen uniformly at random and $\mathbf{x} \leftarrow \chi^m$ is chosen from the error distribution $\chi = \bar{\Psi}_\alpha$ for some parameter α.

The encryption algorithm LWEEnc is slightly different: instead of choosing $\mathbf{e} \in \{0,1\}^m$, it chooses $\mathbf{e} \in \mathbb{Z}^m$ from a certain distribution D_r parameterized by a value r. Roughly speaking, D_r is a Gaussian-like distribution over $\mathbb{Z}^m$ having standard deviation approximately r. The decryption algorithm LWEDec is exactly the same as ours.

The cryptosystem is correct as long as the LWE noise parameter α is small enough (and q is large enough) to compensate for the size of r (recall that the value of α determines the approximation factor in the underlying worst-case lattice assumptions). A key property of LWEEnc is that a larger value of r means that a larger fraction of public keys $\mathbf{p}$ are messy. In our dual-mode construction, we need this fraction to be extremely close to 1, so we use a moderately large value of r.

The $\mathsf{IsMessy}$ algorithm that identifies messy public keys (using the trapdoor $\mathbf{S}$ for $\mathbf{A}$) is a probabilistic algorithm that has two main properties, which we describe here informally. First, if it declares some key $\mathbf{p}$ to be messy, then $\mathbf{p}$ is indeed messy. Second, $\mathsf{IsMessy}$ declares a very large fraction (close to 1) of keys $\mathbf{p}$ to be messy. (Both properties actually hold with overwhelming probability over the algorithm's randomness.) Note that $\mathsf{IsMessy}$ may make one-sided error, i.e., it might output "not sure" on a key that is in fact messy. Nevertheless, it identifies most messy public keys as such, and this will be good enough for our purposes. In particular, we can show that it correctly identifies a messy branch for *any* (adversarially-chosen) public key in our dual-mode cryptosystem (for almost all values of the crs). The proof of this fact (Lemma 7.9) uses a careful counting argument that depends crucially on the fraction of keys that $\mathsf{IsMessy}$ declares to be messy.

Construction. We now present our construction of a dual-mode cryptosystem. For clarity, we present a construction for $\ell = 1$ and omit the extra parameter $i \in [\ell]$, later noting the changes needed in the general case. We retain all the notation from Section 7.2, and for simplicity let $p = 2$, so the message space $\mathbb{Z}_p = \{0,1\}$. We stress that the dual-mode cryptosystem *itself* does *not* do any amortization. Instead, the amortization technique applies to the ℓ individual OT executions using the same crs, which each transfer a single message bit to the receiver.

- $\mathsf{SetupMessy}(1^n)$: choose a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ uniformly at random, together with a trapdoor $t = (\mathbf{S}, \mathbf{A})$ as described in [GPV08]. For each $b \in \{0,1\}$, choose an independent row vector $\mathbf{v}_b \leftarrow \mathbb{Z}_q^{1\times m}$ uniformly at random. Let $\mathsf{crs} = (\mathbf{A}, \mathbf{v}_0, \mathbf{v}_1)$, and output (crs, t).

  $\mathsf{SetupDec}(1^n)$: choose a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ uniformly at random. Choose a row vector $\mathbf{w} \leftarrow \mathbb{Z}_q^{1\times m}$ uniformly at random. For each $b \in \{0,1\}$, choose a secret $\mathbf{s}_b \leftarrow \mathbb{Z}_q^n$ uniformly at random and an error row vector $\mathbf{x}_b \leftarrow \chi^{1\times m}$ (i.e., the m entries are chosen independently from error distribution χ). Let $\mathbf{v}_b = \mathbf{s}_b^T\mathbf{A} + \mathbf{x}_b - \mathbf{w}$. Let $\mathsf{crs} = (\mathbf{A}, \mathbf{v}_0, \mathbf{v}_1)$, let $t = (\mathbf{w}, \mathbf{s}_0, \mathbf{s}_1)$, and output (crs, t).

  In the following, all algorithms are implicitly provided the crs and parse it as $(\mathbf{A}, \mathbf{v}_0, \mathbf{v}_1)$.

- $\mathsf{KeyGen}(\sigma)$: choose a secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly at random and a row vector $\mathbf{x} \leftarrow \chi^{1\times m}$. Let $pk = \mathbf{s}^T\mathbf{A} + \mathbf{x} - \mathbf{v}_\sigma$, let $sk = \mathbf{s}$, and output (pk, sk).

- $\mathsf{Enc}(pk, b, m)$: output $c \leftarrow \mathsf{LWEEnc}((\mathbf{A}, pk + \mathbf{v}_b), m)$.[4]

[4] The LWEEnc algorithm here is the one from [GPV08, Section 6], which chooses the randomness $\mathbf{e}$ from the

- $\mathsf{Dec}(sk, c)$: output $m \leftarrow \mathsf{LWEDec}(sk, c)$.
- $\mathsf{FindMessy}(t, pk)$: parse t as $(\mathbf{S}, \mathbf{A})$, run $\mathsf{IsMessy}(\mathbf{S}, \mathbf{A}, pk + \mathbf{v}_b)$ for each $b \in \{0, 1\}$, and output a b such that IsMessy outputs "messy." (Lemma 7.9 below shows that IsMessy outputs "messy" on at least one branch, and is correct with overwhelming probability.)

  Alternately, we may let FindMessy be an exponential-time algorithm that computes $\delta(\mathbf{A}, pk + \mathbf{u}_b)$ by brute-force enumeration for each $b \in \{0, 1\}$. In the OT protocol, this translates to an exponential-time statistically-close simulation for a cheating receiver. While not sufficient for UC security, this is good enough for (say) concurrent composition of many OT executions (and no other protocols) between the same sender and receiver.
- $\mathsf{TrapKeyGen}(t)$: Parse the trapdoor t as $(\mathbf{w}, \mathbf{s}_0, \mathbf{s}_1)$, and output $(pk, sk_0, sk_1) = (\mathbf{w}, \mathbf{s}_0, \mathbf{s}_1)$.

For general ℓ, we make the following simple modifications: the setup algorithms choose a single matrix $\mathbf{A}$, and independently choose $\mathbf{v}_b^{(i)}$ (in SetupMessy) or $\mathbf{w}^{(i)}, \mathbf{s}_b^{(i)}$ (in SetupDec) for each $i \in [\ell]$ and $b \in \{0, 1\}$ as above, placing the appropriate values in crs and t. On their ith executions, the algorithms use $\mathbf{w}^{(i)}$ instead of $\mathbf{w}$, etc., use independent randomness, but use the same matrix $\mathbf{A}$ throughout. In particular, we stress that Enc reuses only $\mathbf{A}$; it *does not* reuse the randomness of LWEEnc (as is done in the amortized cryptosystem of Section 7.2). The reason is that an *adversarial* receiver chooses the public keys that are used for encryption. If randomness is reused, this may allow the receiver to introduce correlations between the ciphertexts from different executions. Using fresh randomness in every execution avoids this issue and is provably secure.

We can also generalize the above system to a larger branch set $\{0, 1\}^k$ for a 1-out-of-2^k OT protocol, by including vectors $\mathbf{v}_b$ for every $b \in \{0, 1\}^k$ in the crs, chosen in an analogous way. The only difference in the proof of security is in Lemma 7.9, where the probability of the "bad" event increases by a polynomial factor (but remains negligible), due to a union bound over all pairs of branches.

7.3.1 Dual-Mode Properties

The proof that the above system comprises a dual-mode cryptosystem (according to our relaxed definition) is somewhat involved. Therefore we break it into separate lemmas relating to the indistinguishability of the two modes (Lemma 7.6), the trapdoor generation of keys in decryption mode (Lemma 7.7), and the guaranteed existence and identification of messy branches in messy mode (Lemma 7.9).

Lemma 7.6. *In the above dual-mode system construction, the messy and decryption modes are indistinguishable, i.e.,* $\mathsf{SetupMessy}_1(1^n) \overset{c}{\approx} \mathsf{SetupDec}_1(1^n)$, *assuming* $\mathit{LWE}_{q,\chi}$ *is hard.*

Proof. Consider the output of $\mathsf{SetupDec}_1$, which is of the form $(\mathbf{A}, (\mathbf{s}_0^T\mathbf{A} + \mathbf{x}_0) - \mathbf{w}, (\mathbf{s}_1^T\mathbf{A} + \mathbf{x}_1) - \mathbf{w})$. By the hardness of $\mathsf{LWE}_{q,\chi}$, we have $(\mathbf{A}, \mathbf{s}_1^T\mathbf{A} + \mathbf{x}_1) \overset{c}{\approx} (\mathbf{A}, \mathbf{w}')$, where row vector $\mathbf{w}' \leftarrow \mathbb{Z}_q^{1\times m}$ is uniformly random and independent. Therefore the output of $\mathsf{SetupDec}_1$ is indistinguishable from a tuple $(\mathbf{A}, (\mathbf{s}_0^T\mathbf{A} + \mathbf{x}_0) - \mathbf{w}, \mathbf{w}' - \mathbf{w})$. This tuple is totally uniform, because $\mathbf{w}$ and $\mathbf{w}'$ are uniform and independent. Because the output of $\mathsf{SetupMessy}_1$ is entirely uniform, the claim follows. □

distribution D_r, for a value of r we specify later.

Lemma 7.7. *The above dual-mode construction satisfies our relaxed Property $4'$, assuming $\mathsf{LWE}_{q,\chi}$ is hard.*

Proof. For simplicity, we will prove the lemma for $\ell = 1$; the general case follows by a hybrid argument over the ℓ independent pairs of variables in the crs and the ℓ calls to $\mathsf{KeyGen}/\mathsf{TrapKeyGen}$ that use these variables. Without loss of generality, we consider only the case $\sigma = 0$; the other case follows symmetrically.

Our goal is to prove that

$$(\mathsf{SetupDec}_1(1^n), \mathsf{KeyGen}(0)) \overset{c}{\approx} (\mathsf{crs}, (pk, sk_0)), \tag{1}$$

where in the right-hand side $(\mathsf{crs}, t) \leftarrow \mathsf{SetupDec}(1^n)$ and $(pk, sk_0, sk_1) \leftarrow \mathsf{TrapKeyGen}(t)$. We do so via a sequence of hybrid games.

First consider a hybrid game that outputs $(\mathsf{SetupMessy}_1(1^n), \mathsf{KeyGen}(0))$; this is indistinguishable from the left-hand side of (1) by indistinguishability of modes (proved above in Lemma 7.6). The output of this hybrid game expands as

$$(\mathbf{A}, \quad \mathbf{v}_0, \quad \mathbf{v}_1, \quad \mathbf{s}^T\mathbf{A} + \mathbf{x} - \mathbf{v}_0, \quad \mathbf{s}),$$

where $\mathbf{A}$, $\mathbf{v}_0$, $\mathbf{v}_1$, and $\mathbf{s}$ are uniform (in their respective domains) and $\mathbf{x} \leftarrow \chi^{1\times m}$.

Now rename variables in the above game, defining $\mathbf{w} = \mathbf{s}^T\mathbf{A} + \mathbf{x} - \mathbf{v}_0$ and renaming $\mathbf{s}$ and $\mathbf{x}$ as $\mathbf{s}_0$ and $\mathbf{x}_0$ (respectively). The above game is therefore equivalent to one that outputs

$$(\mathbf{A}, \quad \mathbf{s}_0^T\mathbf{A} + \mathbf{x}_0 - \mathbf{w}, \quad \mathbf{v}_1, \quad \mathbf{w}, \quad \mathbf{s}_0),$$

where $\mathbf{w}$ is uniform. Now because $\mathbf{v}_1$ is uniform and independent of the other variables, the preceding game is equivalent to one that outputs

$$(\mathbf{A}, \quad \mathbf{s}_0^T\mathbf{A} + \mathbf{x}_0 - \mathbf{w}, \quad \mathbf{v}_1 - \mathbf{w}, \quad \mathbf{w}, \quad \mathbf{s}_0).$$

Finally, the hardness of $\mathsf{LWE}_{q,\chi}$ implies that $(\mathbf{A}, \mathbf{v}_1)$ is indistinguishable from $(\mathbf{A}, \mathbf{s}_1^T\mathbf{A} + \mathbf{x}_1)$, where $\mathbf{s}_1 \leftarrow \mathbb{Z}_q^n$ and $\mathbf{x}_1 \leftarrow \chi^{1\times m}$. Therefore the prior game is indistinguishable from one that outputs

$$(\mathbf{A}, \quad \mathbf{s}_0^T\mathbf{A} + \mathbf{x}_0 - \mathbf{w}, \quad \mathbf{s}_1^T\mathbf{A} + \mathbf{x}_1 - \mathbf{w}, \quad \mathbf{w}, \quad \mathbf{s}_0).$$

This game is, by definition, equivalent to the right-hand side of (1), and we are done. □

In order to show that $\mathsf{FindMessy}$ works properly, we require the following facts which summarize the properties of the LWE-based cryptosystem and $\mathsf{IsMessy}$ algorithm constructed in [GPV08].[5]

Proposition 7.8 ([GPV08, Section 6 summary])**.** *Let $m \geq 2(n+1)\lg q$ and let $r \geq \sqrt{qm} \cdot \log^2 m$. There is a negligible function $\epsilon(m)$ such that with overwhelming probability over the choice of $\mathbf{A}, \mathbf{S}$, the following are true:*

- *For all but an at most $(1/2\sqrt{q})^m$ fraction of vectors $\mathbf{p} \in \mathbb{Z}_q^{1\times m}$, $\mathsf{IsMessy}(\mathbf{S}, \mathbf{A}, \mathbf{p})$ outputs "messy" with overwhelming probability (over its own randomness).*

[5] To be specific, Proposition 7.8 instantiates the lemmas from Section 6.2 of [GPV08] with the value $s = 10\sqrt{\log m}/\sqrt{q}$.

- *For any* $\mathbf{p} \in \mathbb{Z}_q^{1\times m}$ *such that* $\delta(\mathbf{p}) > 2\epsilon$*,* $\mathsf{IsMessy}(\mathbf{S}, \mathbf{A}, \mathbf{p})$ *outputs "not sure" with overwhelming probability (over its own randomness).*

Furthermore, if $q \geq 5rm$ *and* $\alpha \leq 1/(r\sqrt{m}\log m)$*, then* LWEDec *decrypts correctly with overwhelming probability.*

Lemma 7.9. *In the above dual-mode construction using the parameters from Proposition 7.8, the following is true for an overwhelming fraction of* $(\mathsf{crs}, t) \leftarrow \mathsf{SetupMessy}(1^n)$*: for* every *key pk, there exists at least one messy branch for pk, and* $\mathsf{FindMessy}(t, pk)$ *outputs such a branch with overwhelming probability (over its own randomness).*

Proof. Fix $\mathbf{A}$ (and $\mathbf{S}$) to be among the $1 - 2q^{-n}$ fraction of matrices for which the two items in Proposition 7.8 hold. Define $M \subseteq \mathbb{Z}_q^{1\times m}$ to be the set of vectors $\mathbf{p}$ described in the first item, for which $\mathsf{IsMessy}(\mathbf{S}, \mathbf{A}, \mathbf{p})$ outputs "messy" with overwhelming probability. Then we have

$$\Pr_{\mathbf{v}\in\mathbb{Z}_q^{1\times m}}[\mathbf{v} \notin M] \leq (1/2\sqrt{q})^m.$$

First we show that *every* $pk \in \mathbb{Z}_q^{1\times m}$ has some branch $pk + \mathbf{v}_b \in M$, with overwhelming probability over the remaining random choices of $\mathbf{v}_0, \mathbf{v}_1 \in \mathbb{Z}_q^{1\times m}$ in the crs. For any *fixed pk*, we have

$$\Pr_{\mathbf{v}_0,\mathbf{v}_1}[pk + \mathbf{v}_0 \notin M \text{ and } pk + \mathbf{v}_1 \notin M] = (\Pr_{\mathbf{v}}[\mathbf{v} \notin M])^2 \leq (1/4q)^m.$$

Applying the union bound over all q^m distinct values of *pk*, the probability over $\mathbf{v}_0, \mathbf{v}_1$ that there *exists* a *pk* for which both branches lie outside M is at most $(1/4)^m = \mathsf{negl}(n)$. In other words, for almost every choice of crs, *every* value of *pk* causes $\mathsf{IsMessy}$ to output "messy" with overwhelming probability on at least one branch of *pk*. Finally, the second item of Proposition 7.8 guarantees that if $\delta(pk + \mathbf{v}_b) > 2\epsilon$, then $\mathsf{IsMessy}$ outputs "messy" with negligible probability. In other words, with overwhelming probability the branch b chosen by $\mathsf{FindMessy}(t, pk)$ has $\delta(pk+\mathbf{v}_b) \leq 2\epsilon = \mathsf{negl}(n)$. □

7.3.2 Putting Everything Together

We now instantiate all the parameters of the dual-mode cryptosystem to satisfy the hypotheses of Proposition 7.8, and compute the underlying concrete approximation factors for lattice problems.

First we instantiate the parameters (we have made no effort to optimize the various constant or polylog(n) factors). Let $m = 8(n + 1)\lg n = O(n \lg n)$. Let q be a prime in $[25, 100] \cdot m^3 \lg^6 m = \tilde{O}(n^3)$. Let $r = \sqrt{qm} \cdot \log^2 m = \tilde{O}(m^2)$. Finally, let the error distribution $\chi = \bar{\Psi}_\alpha$ for $\alpha = 1/(r\sqrt{m}\log m) = 1/\tilde{O}(n^{5/2})$.

Theorem 7.10. *The above system is a dual-mode cryptosystem (according to the relaxed definition), unless both SIVP and GapSIVP are easy for quantum algorithms to approximate to within some* $\tilde{O}(n^{7/2})$ *factor.*

Proof. As long as $\mathsf{LWE}_{q,\chi}$ is hard (which we show below), indistinguishability of modes follows directly from Lemma 7.6, and trapdoor key generation follows directly from Lemma 7.7.

Completeness and trapdoor messy-branch identification follow from Proposition 7.8 and Lemma 7.9 (respectively), as long as the hypotheses of Proposition 7.8 are met. We now show that this is the case.

First, observe that

$$2(n+1)\lg q \leq 2(n+1)\lg n^4 \leq 8(n+1)\lg n = m$$

for all sufficiently large n. Next, we have $r \geq \sqrt{qm} \cdot \log^2 m$ by definition of r. Next, observe (by substituting for r) that

$$q \geq 5rm \iff \sqrt{q} \geq 5m^{3/2}\log^3 m \iff q \geq 25m^3 \log^6 m,$$

which is true by definition of q. Finally, $\alpha \leq 1/(r\sqrt{m}\log m)$ by definition of α.

We conclude by justifying the hardness of $\mathsf{LWE}_{q,\chi}$. Using the fact that $q \geq 5rm$ and the definition of α, we have have $q\alpha > 5\sqrt{m} \geq 2\sqrt{n}$, as required by Proposition 7.1. Therefore $\mathsf{LWE}_{q,\chi}$ is hard unless approximating SIVP and GapSVP to within some $\tilde{O}(n/\alpha) = \tilde{O}(n^{7/2})$ factor is easy for quantum algorithms. □

8 Acknowledgments

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

References

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293, 1997.

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001.

[Ajt04] Miklós Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, pages 19–40, 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *EUROCRYPT*, pages 573–590, 2007.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In *CRYPTO*, pages 265–281, 2003.

[Cré87] Claude Crépeau. Equivalence between two flavours of oblivious transfers. In *CRYPTO*, pages 350–354, 1987.

[CS02] 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.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[GH07] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In *ASIACRYPT*, pages 265–282, 2007.

[GMW87] 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.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, pages 339–358, 2006.

[GPV08] 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`.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, pages 145–161, 2003.

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *FOCS*, pages 248–253, 1989.

[JS07] Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In *EUROCRYPT*, pages 97–114, 2007.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In *EUROCRYPT*, pages 78–95, 2005.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[KTX07] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In *Public Key Cryptography*, pages 315–329, 2007.

[Lin08] Yehuda Lindell. Efficient fully simulatable oblivious transfer. In *CT-RSA*, 2008.

[MR07] 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.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO*, pages 573–590, 1999.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[Pei07] Chris Peikert. Limits on the hardness of lattice problems in ℓ_p norms. In *IEEE Conference on Computational Complexity*, pages 333–346, 2007.

[PW08] 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`.

[Rab81] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report, Harvard University, 1981.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.