More Constructions of Lossy and Correlation-Secure Trapdoor Functions[*]

David Mandell Freeman
Stanford University, USA
dfreeman@cs.stanford.edu

Oded Goldreich
Weizmann Institute of Science, Israel
oded.goldreich@weizmann.ac.il

Eike Kiltz
CWI, Netherlands
kiltz@cwi.nl

Alon Rosen
IDC Herzliya, Israel
alon.rosen@idc.ac.il

Gil Segev
Weizmann Institute of Science, Israel
gil.segev@weizmann.ac.il

May 13, 2010

Abstract

We propose new and improved instantiations of lossy trapdoor functions (Peikert and Waters, STOC '08), and correlation-secure trapdoor functions (Rosen and Segev, TCC '09). Our constructions widen the set of number-theoretic assumptions upon which these primitives can be based, and are summarized as follows:

- Lossy trapdoor functions based on the quadratic residuosity assumption. Our construction relies on modular squaring, and whereas previous such constructions were based on seemingly stronger assumptions, we present the first construction that is based solely on the quadratic residuosity assumption. We also present a generalization to higher order power residues.
- Lossy trapdoor functions based on the composite residuosity assumption. Our construction guarantees essentially any required amount of lossiness, where at the same time the functions are more efficient than the matrix-based approach of Peikert and Waters.
- Lossy trapdoor functions based on the d-Linear assumption. Our construction both simplifies the DDH-based construction of Peikert and Waters, and admits a generalization to the whole family of d-Linear assumptions without any loss of efficiency.
- Correlation-secure trapdoor functions related to the hardness of syndrome decoding.

Keywords: Public-key encryption, lossy trapdoor functions, correlation-secure trapdoor functions.

[*]An extended abstract of this work appears in *Public Key Cryptography — PKC 2010*, Springer LNCS **6056** (2010), pages 279–295.

1 Introduction

In this paper, we describe new constructions of lossy trapdoor functions and correlation-secure trapdoor functions. These primitives are strengthened variants of the classical notion of trapdoor functions, and were introduced with the main goal of enabling simple and black-box constructions of public-key encryption schemes that are secure against chosen-ciphertext attacks. At a high level, they are defined as follows:

Lossy trapdoor functions [33]: A collection of lossy trapdoor functions consists of two families of functions. Functions in one family are injective and can be efficiently inverted using a trapdoor. Functions in the other family are "lossy," which means that the size of their image is significantly smaller than the size of their domain. The only computational requirement is that a description of a randomly chosen function from the family of injective functions is computationally indistinguishable from a description of a randomly chosen function from the family of lossy functions.

Correlation-secure trapdoor functions [35]: The classical notion of a one-way function asks for a function that is efficiently computable but is hard to invert given the image of a uniformly chosen input. Correlation security generalizes the one-wayness requirement by considering k-wise products of functions and any specified input distribution, not necessarily the uniform distribution. Given a collection of functions $\mathcal{F}$ and a distribution $\mathcal{C}$ over k-tuples of inputs, we say that $\mathcal{F}$ is secure under $\mathcal{C}$-correlated inputs if the function $(f_1(x_1), \ldots, f_k(x_k))$ is one-way, where $f_1, \ldots, f_k$ are independently chosen from $\mathcal{F}$ and $(x_1, \ldots, x_k)$ are sampled from $\mathcal{C}$.

Lossy trapdoor functions were introduced by Peikert and Waters [33], who showed that they imply fundamental cryptographic primitives such as trapdoor functions, collision-resistant hash functions, oblivious transfer, and CCA-secure public-key encryption. In addition, lossy trapdoor functions have already found various other applications, including deterministic public-key encryption [4], OAEP-based public-key encryption [24], "hedged" public-key encryption for protecting against bad randomness [1], security against selective opening attacks [2], and efficient non-interactive string commitments [30].

The notion of correlation security was introduced by Rosen and Segev [35], who showed that any collection of injective trapdoor functions that is one-way under a natural input distribution can be used to construct a CCA-secure public-key encryption scheme.[1] They showed that any collection of lossy trapdoor functions that are sufficiently lossy is in fact also correlation-secure. This result was recently refined by Mol and Yilek [26] who showed that even lossiness of any polynomial fraction of a single bit suffices.

These applications motivate us to investigate new constructions of lossy and correlation-secure functions. Such constructions would enable us to widen the basis upon which one can achieve the above cryptographic tasks in a simple and modular way.

1.1 Our Contributions

We propose new and improved constructions of lossy and correlation-secure trapdoor functions based on well-established number-theoretic assumptions, some of which were previously not known

[1]Any distribution where $(x_1, \ldots, x_k)$ are $(1-\epsilon)k$-wise independent, for a constant $\epsilon < 1$, can be used in their framework. In particular, this includes the case where x_1 is uniformly distributed and $x_1 = \cdots = x_k$.

to imply either of the primitives. By directly applying the results of [33, 35, 26], we obtain new CCA-secure public-key encryption schemes based on these assumptions. Concretely, we present the following constructions:

1. *Lossy trapdoor permutations based on the quadratic residuosity assumption.* Our construction relies on Rabin's modular squaring function and is based solely on the quadratic residuosity assumption. More precisely, the function is defined as $f(x) = x^2 \cdot \delta_{r,s}(x) \bmod N$, where $N = PQ$ is an RSA modulus and $\delta_{r,s}(\cdot)$ is a function indexed by two public elements $r, s \in \mathbb{Z}_N$ serving two independent purposes. First, it extends the modular squaring function to a permutation over $\mathbb{Z}_N$. Second, it causes the function $f(x)$ to lose the information about the sign of x if and only if s is a quadratic residue. Therefore, under the quadratic residuosity assumption f has one bit of lossiness. We note that although a function with only one bit of lossiness (or, more generally, with only a non-negligible amount of lossiness) is not necessarily a (strong) one-way function, it nevertheless can be used as a building block for constructing even a CCA-secure public-key encryption scheme (see [26, 35]). In addition, we describe a generalization of the construction to higher order power residues that allows for more lossiness.

2. *Lossy trapdoor functions based on the composite residuosity assumption.* Our construction is based on the Damgård-Jurik encryption scheme [13] with additional insights by Damgård and Nielsen [14, 15]. The Damgård-Jurik scheme is based on computations in the group $\mathbb{Z}_{N^{s+1}}$, where $N = PQ$ is an RSA modulus and $s \geq 1$ is an integer (it contains Paillier's encryption scheme [31] as a special case by setting $s = 1$). At a high level, each function is described by a pair (pk, c), where pk is a public key for the encryption scheme, and c is either an encryption of 1 (injective mode) or an encryption of 0 (lossy mode). By using the homomorphic properties of the encryption scheme, given such a ciphertext c and an element x, it is possible to compute either an encryption of x in the injective mode, or an encryption of 0 in the lossy mode. We note that this construction was concurrently and independently proposed by Boldyreva et al. [4]. We also give an "all-but-one" version of the construction.

3. *Lossy trapdoor functions based on the d-Linear assumption.* Our construction both simplifies and generalizes the DDH-based construction of Peikert and Waters [33, Section 5]. (Recall that DDH is the 1-Linear assumption.) Let $\mathbb{G}$ be a finite group of order p and choose an $n \times n$ matrix M over $\mathbb{F}_p$ that has either rank d (lossy mode) or rank n (injective mode). We "encrypt" $M = (a_{ij})$ as the matrix $g^M = (g^{a_{ij}}) \in \mathbb{G}^{n \times n}$, where g is a generator of $\mathbb{G}$. If $\vec{x}$ is a binary vector of length n, then given g^M we can efficiently evaluate the function $f_M(\vec{x}) = g^{M\vec{x}} \in \mathbb{G}^n$. If M has rank n, then given M we can efficiently invert f_M on the image of $\{0,1\}^n$. On the other hand, if M has rank d and $p < 2^{n/d}$, then f is lossy. The d-Linear assumption implies that the lossy and injective modes cannot be efficiently distinguished. We also give an "all-but-one" version of the function f_M based on the DDH assumption.

4. *Correlation-secure trapdoor functions based on the hardness of syndrome decoding.* Our construction is based on Niederreiter's coding-based encryption system [29] which itself is the dual of the McEliece encryption system [25]. Our trapdoor function is defined as $f(x) = Hx$, where H is a binary $(n-k) \times n$ matrix (of a certain distribution that allows for embedding a trapdoor) and x is bit string of small Hamming weight. We show that the function's correlation security is directly implied by a result of Fischer and Stern [17] about the pseudorandomness of the function f. Interestingly, the related McEliece trapdoor function (which can be viewed

as the dual of the Niederreiter function) is not correlation-secure.[2] It is however possible to extend the framework of correlation security in a natural way to obtain a direct construction of a CCA-secure encryption scheme from the McEliece trapdoor function. This was recently demonstrated by Dowsley et al [16] (who proposed the first coding-based encryption scheme that is CCA-secure in the standard model) and, for the related lattice case, independently by Peikert [32] and Goldwasser and Vaikuntanathan [19]. Our contribution is to show that the Niederreiter function admits a simple construction of correlation-secure trapdoor functions based on the same security assumptions as [16].[3] The resulting CCA-secure encryption scheme is as efficient as the one from [16].

1.2 Related Work

Most of the known constructions and applications of lossy and correlation-secure trapdoor functions are already mentioned above; here we include a few more. Besides their construction based on DDH, Peikert and Waters [33] also present a construction of lossy trapdoor functions based on the worst-case hardness of lattice problems. The construction does not enjoy the same amount of lossiness as their DDH-based one, but it still suffices for their construction of a CCA-secure public-key encryption scheme. The worst-case hardness of lattice problems is also used by Peikert [32] and by Goldwasser and Vaikuntanathan [19] to construct CCA-secure encryption schemes using a natural generalization of correlation-secure trapdoor functions.

Kiltz et al. [24] show that the RSA trapdoor permutation is lossy under the Φ-Hiding assumption of Cachin et al. [9]. (Concretely, it has $\log_2(e)$ bits of lossiness, where e is the public RSA exponent.) Furthermore, they propose multi-prime hardness assumptions under which RSA has greater lossiness.

In concurrent and independent work, Mol and Yilek [26] propose a lossy trapdoor function based on the modular squaring function. Though this construction is related to ours, its security is based on the seemingly stronger assumption that a random two-prime RSA modulus is indistinguishable from a random three-prime RSA modulus. (See Appendix A for further discussion of this assumption.) In another concurrent and independent work, Hemenway and Ostrovsky [20] generalize the framework of Peikert and Waters [33] to rely on any homomorphic hash proof system, which is an extension of Cramer and Shoup's notion of hash proof systems [12]. Hemenway and Ostrovsky then show that homomorphic hash proof systems can be constructed based on either the quadratic residuosity assumption or the composite residuosity assumption. Their approach is significantly different than ours, and the resulting constructions seem incomparable when considering the trade-off between efficiency and lossiness.

1.3 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we review the definitions of lossy and correlation-secure trapdoor functions. In Section 3, we present our construction based on the quadratic residuosity assumption, and in Section 4 we generalize this construction to higher order

[2]The McEliece trapdoor function is defined as $f'_H(x, e) := Hx \oplus e$, where H is a binary $k \times n$ matrix, x is a k-bit string and e is a error vector of small Hamming weight. Given H_1, H_2 and two evaluations $y_1 = H_1x \oplus e$ and $y_2 = H_2x \oplus e$ one can reconstruct the unique x by solving $(H_1 \oplus H_2)x = y_1 \oplus y_2$ for x.

[3]We remark that our construction of a correlation-secure trapdoor function from coding theory does not carry over to the lattice case since the "dual" of the one-way function used in [32, 19] is not injective.

residues. In Sections 5, 6, and 7 we present our constructions based on the composite residuosity assumption, the d-Linear assumption, and the hardness of syndrome decoding, respectively.

In Appendix A, we revisit the folklore that relates the distinguishability of 2-prime and 3-prime composites to the quadratic residuosity assumption. In particular, we propose and prove a reasonable instantiation of this folklore.

2 Preliminaries

We assume familiarity with standard cryptographic objects and notions such as one-way functions, computational indistinguishability, trapdoor permutations, public-key encryption, and chosen ciphertext security. The reader is referred to [18] for definitions.

2.1 Lossy Trapdoor Functions

A *collection of lossy trapdoor functions* consists of two families of functions. Functions in one family are injective and can be efficiently inverted using a trapdoor. Functions in the other family are "lossy," which means that the size of their image is significantly smaller than the size of their domain. The only computational requirement is that a description of a randomly chosen function from the family of injective functions is computationally indistinguishable from a description of a randomly chosen function from the family of lossy functions.

Definition 2.1 (Lossy trapdoor functions)**.** Let $m : \mathbb{N} \to \mathbb{N}$ and $\ell : \mathbb{N} \to \mathbb{R}$ be two non-negative functions, and for any $n \in \mathbb{N}$, let $m = m(n)$ and $\ell = \ell(n)$. A collection of (m, ℓ)-lossy trapdoor functions is a 4-tuple of probabilistic polynomial-time algorithms $(\mathsf{G}_0, \mathsf{G}_1, \mathsf{F}, \mathsf{F}^{-1})$ such that:

1. **Sampling a lossy function:** $\mathsf{G}_0(1^n)$ outputs a function index $\sigma \in \{0,1\}^*$.
2. **Sampling an injective function:** $\mathsf{G}_1(1^n)$ outputs a pair $(\sigma, \tau) \in \{0,1\}^* \times \{0,1\}^*$. (Here σ is a function index and τ is a trapdoor.)
3. **Evaluation of lossy functions:** For every function index σ produced by G_0, the algorithm $\mathsf{F}(\sigma, \cdot)$ computes a function $f_\sigma : \{0,1\}^m \to \{0,1\}^*$, whose image is of size at most $2^{m-\ell}$.
4. **Evaluation of injective functions:** For every pair (σ, τ) produced by G_1, the algorithm $\mathsf{F}(\sigma, \cdot)$ computes an injective function $f_\sigma : \{0,1\}^m \to \{0,1\}^*$.
5. **Inversion of injective functions:** For every pair (σ, τ) produced by G_1 and every $x \in \{0,1\}^m$, we have $\mathsf{F}^{-1}(\tau, \mathsf{F}(\sigma, x)) = x$.
6. **Security:** The two ensembles $\{\sigma : \sigma \leftarrow \mathsf{G}_0(1^n)\}_{n \in \mathbb{N}}$ and $\{\sigma : (\sigma, \tau) \leftarrow \mathsf{G}_1(1^n)\}_{n \in \mathbb{N}}$ are computationally indistinguishable.

Note that the size m of the domain (of both types of functions) and the size $m - \ell$ of the image of lossy functions depend on the security parameter n. Note also that we do not specify the output of F^{-1} on inputs not in the image of f_σ. In the above definition we have assumed for simplicity that the domain is $\{0,1\}^m$. More generally, one may allow the functions to have arbitrary domains D_n (this would correspond to $D_n = \{0,1\}^m$ in the definition above). In such a case, one would have to add the requirement that D_n's size is bounded below by 2^{m-1}, and that the image of a lossy function is of size at most $|D_n| \cdot 2^{-\ell}$. (In Definition 2.3 below we generalize the definition so that the domain depends not only on n but also on the function index σ.)

A *collection of all-but-one lossy trapdoor functions* is a more general primitive. Such a collection is associated with a set B, whose members are referred to as *branches*. (If $B = \{0,1\}$ then we obtain the previous notion of lossy trapdoor functions.) The sampling algorithm of the collection receives an additional parameter $b^* \in B$, and outputs a description of a function $f(\cdot,\cdot)$ together with a trapdoor τ and a set of lossy branches β. The function f has the property that for any branch $b \notin \beta$ the function $f(b,\cdot)$ is injective (and can be inverted using τ), and the function $f(b^*,\cdot)$ is lossy. Moreover, the description of f hides (in a computational sense) the set of lossy branches β.

Our definition is slightly more general than that of Peikert and Waters [33, Section 3.2], which allows only one lossy branch (i.e., $\beta = \{b^*\}$). We allow possibly many lossy branches (other than b^*), and require that given a description of a function and b^* it is computationally infeasible to find another lossy branch. The proof of security of the Peikert-Waters CCA-secure public-key encryption scheme [33, Section 4.3] can easily be adapted to our more general context. (We are currently not aware of other applications of all-but-one lossy trapdoor functions).

Definition 2.2 (All-but-one lossy trapdoor functions)**.** Let $m : \mathbb{N} \to \mathbb{N}$ and $\ell : \mathbb{N} \to \mathbb{R}$ be two non-negative functions, and for any $n \in \mathbb{N}$, let $m = m(n)$ and $\ell = \ell(n)$. A collection of (m,ℓ)-all-but-one lossy trapdoor functions is a 4-tuple of probabilistic polynomial-time algorithms $(\mathsf{B},\mathsf{G},\mathsf{F},\mathsf{F}^{-1})$ such that:

1. **Sampling a branch:** $\mathsf{B}(1^n)$ outputs a value $b \in \{0,1\}^*$.

2. **Sampling a function:** For every value b produced by $\mathsf{B}(1^n)$, the algorithm $\mathsf{G}(1^n, b)$ outputs a triple $(\sigma,\tau,\beta) \in \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$ consisting of a function index σ, a trapdoor τ, and a set of lossy branches β with $b^* \in \beta$.

3. **Evaluation of lossy functions:** For every value b^* produced by $\mathsf{B}(1^n)$ and for every (σ,τ,β) produced by $\mathsf{G}(1^n, b^*)$, the algorithm $\mathsf{F}(\sigma, b^*, \cdot)$ computes a function $f_{\sigma,b^*} : \{0,1\}^m \to \{0,1\}^*$, whose image is of size at most $2^{m-\ell}$.

4. **Evaluation of injective functions:** For any b^* and b produced by $\mathsf{B}(1^n)$ and for every (σ,τ,β) produced by $\mathsf{G}(1^n, b^*)$, if $b \notin \beta$, then the algorithm $\mathsf{F}(\sigma, b, \cdot)$ computes an injective function $f_{\sigma,b} : \{0,1\}^m \to \{0,1\}^*$.

5. **Inversion of injective functions:** For any b^* and b produced by $\mathsf{B}(1^n)$ and for every (σ,τ,β) produced by $\mathsf{G}(1^n, b^*)$, if $b \notin \beta$ then we have
$$\mathsf{F}^{-1}(\tau, b, \mathsf{F}(\sigma, b, x)) = x.$$

6. **Security:** For any two sequences $\{(b_n^*, b_n)\}_{n\in\mathbb{N}}$ such that b_n^* and b_n are distinct values in the image of $\mathsf{B}(1^n)$, the two ensembles $\{\sigma : (\sigma,\tau,\beta) \leftarrow \mathsf{G}(1^n, b_n^*)\}_{n\in\mathbb{N}}$ and $\{\sigma : (\sigma,\tau,\beta) \leftarrow \mathsf{G}(1^n, b_n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable.

7. **Hiding lossy branches:** Any probabilistic polynomial-time algorithm $\mathcal{A}$ that receives as input (σ, b^*), where $b^* \leftarrow \mathsf{B}(1^n)$ and $(\sigma,\tau,\beta) \leftarrow \mathsf{G}(1^n, b^*)$, has only a negligible probability of outputting an element $b \in \beta \setminus \{b^*\}$ (where the probability is taken over the randomness of B, G, and $\mathcal{A}$).

We now introduce a useful generalization of lossy trapdoor functions which we call a *lossy trapdoor functions with index-dependent domains.* The only difference between these functions and those defined above is that the function's domain is no longer fixed to $\{0,1\}^m$; instead, it may depend on the function index σ.

Definition 2.3 (Lossy trapdoor functions with index-dependent domains)**.** Let $m : \mathbb{N} \to \mathbb{N}$ and $\ell : \mathbb{N} \to \mathbb{R}$ be two non-negative functions, and for any $n \in \mathbb{N}$, let $m = m(n)$ and $\ell = \ell(n)$. A collection of (m, ℓ)-lossy trapdoor functions with index-dependent domains is a 5-tuple of probabilistic polynomial-time algorithms $(\mathsf{G}_0, \mathsf{G}_1, \mathsf{S}, \mathsf{F}, \mathsf{F}^{-1})$ such that:

1. **Sampling a lossy function:** $\mathsf{G}_0(1^n)$ outputs a function index $\sigma \in \{0,1\}^*$.
2. **Sampling an injective function:** $\mathsf{G}_1(1^n)$ outputs a pair $(\sigma, \tau) \in \{0,1\}^* \times \{0,1\}^*$. (Here σ is a function index and τ is a trapdoor.)
3. **Sampling an input:** For every value σ produced by either G_0 or G_1, the algorithm $\mathsf{S}(\sigma)$ outputs an element sampled uniformly at random from the domain D_σ of the function f_σ. We require that $|D_\sigma| \geq 2^{m-1}$.
4. **Evaluation of lossy functions:** For every function index σ produced by G_0, the algorithm $\mathsf{F}(\sigma, \cdot)$ computes a function $f_\sigma : D_\sigma \to \{0,1\}^*$, whose image is of size at most $|D_\sigma| \cdot 2^{-\ell}$.
5. **Evaluation of injective functions:** For every pair (σ, τ) produced by G_1, the algorithm $\mathsf{F}(\sigma, \cdot)$ computes an injective function $f_\sigma : D_\sigma \to \{0,1\}^*$.
6. **Inversion of injective functions:** For every pair (σ, τ) produced by G_1 and every $x \in D_\sigma$, we have $\mathsf{F}^{-1}(\tau, \mathsf{F}(\sigma, x)) = x$.
7. **Security:** The two ensembles $\{\sigma : \sigma \leftarrow \mathsf{G}_0(1^n)\}_{n\in\mathbb{N}}$ and $\{\sigma : (\sigma, \tau) \leftarrow \mathsf{G}_1(1^n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable.

It is furthermore possible and straightforward to give an analogous generalization of all-but-one lossy trapdoor functions to handle index-dependent domains.

We remark that lossy trapdoor functions with index-dependent domains do not seem to be sufficient to construct correlated-product secure trapdoor functions or CCA-secure public-key encryption. The difficulty is that in the construction from [33, 35], a fixed value has to be evaluated on many independently generated instances of the trapdoor function (with respect to the same security parameter). It is therefore crucial that the domain stay the same for all these instances. However, lossy trapdoor functions with index-dependent domains are sufficient for many applications. These include deterministic public-key encryption [4], "hedged" public-key encryption for protecting against bad randomness [1], lossy encryption [2], security against selective opening attacks [2], and non-interactive string commitments [30]. (In some of these applications the lossy trapdoor function is required to have additional properties.)

2.2 Correlation-Secure Trapdoor Functions

A *collection of efficiently computable functions* is a pair of algorithms $\mathcal{F} = (\mathsf{G}, \mathsf{F})$, where G is a key generation algorithm used for sampling a description of a function, and F is an evaluation algorithm used for evaluating a function on a given input. The following definition formalizes the notion of a *k-wise product*, which is a collection $\mathcal{F}_k$ consisting of all k-tuples of functions from $\mathcal{F}$.

Definition 2.4 (k-wise product)**.** Let $\mathcal{F} = (\mathsf{G}, \mathsf{F})$ be a collection of efficiently computable functions. For any integer k, we define the *k-wise product* $\mathcal{F}_k = (\mathsf{G}_k, \mathsf{F}_k)$ as follows:

- The key-generation algorithm G_k on input 1^n invokes k independent instances of $\mathsf{G}(1^n)$ and outputs $(\sigma_1, \dots, \sigma_k)$. That is, a function is sampled from $\mathcal{F}_k$ by independently sampling k functions from $\mathcal{F}$.
- The evaluation algorithm F_k on input $(\sigma_1, \dots, \sigma_k, x_1, \dots, x_k)$ invokes F to evaluate each function σ_i on x_i. I.e., $\mathsf{F}_k(\sigma_1, \dots, \sigma_k, x_1, \dots, x_k) = (\mathsf{F}(\sigma_1, x_1), \dots, \mathsf{F}(\sigma_k, x_k))$.

A one-way function is a function that is efficiently computable but is hard to invert given the image of a uniformly chosen input. This notion extends naturally to one-wayness under any specified input distribution, not necessarily the uniform distribution. Specifically, we say that a function is one-way with respect to an input distribution $\mathcal{I}$ if it is efficiently computable but hard to invert given the image of a random input sampled according to $\mathcal{I}$.

In the context of k-wise products, a straightforward argument shows that for any collection $\mathcal{F}$ which is one-way with respect to some input distribution $\mathcal{I}$, the k-wise product $\mathcal{F}_k$ is one-way with respect to the input distribution that samples k independent inputs from $\mathcal{I}$. The following definition formalizes the notion of one-wayness under correlated inputs, where the inputs for $\mathcal{F}_k$ may be correlated.

Definition 2.5 (One-wayness under correlated inputs)**.** Let $\mathcal{F} = (\mathsf{G}, \mathsf{F})$ be a collection of efficiently computable functions with domain $\{D_n\}_{n \in \mathbb{N}}$, and let $\mathcal{C}$ be a distribution where $\mathcal{C}(1^n)$ is distributed over $D_n^k = D_n \times \cdots \times D_n$ for some integer $k = k(n)$. We say that $\mathcal{F}$ is *one-way under $\mathcal{C}$-correlated inputs* if $\mathcal{F}_k$ is one-way with respect to the input distribution $\mathcal{C}$.

For the special case that distribution $\mathcal{C}$ is the uniform k-repetition distribution (i.e., $\mathcal{C}$ samples a uniformly random input $x \in D_n$ and outputs k copies of x), we say that $\mathcal{F}$ is *one-way under k-correlated inputs.* Rosen and Segev [35, Theorem 3.3] show that a collection of (m, ℓ)-lossy trapdoor functions can be used to construct a collection $\mathcal{F}$ that is one-way under k-correlated inputs for any $k < \frac{m - \omega(\log m)}{m - \ell}$.

3 A Construction based on the Quadratic Residuosity Assumption

Our first construction is based on the modular squaring function $x \mapsto x^2 \bmod N$, where $N = PQ$ for prime numbers $P \equiv Q \equiv 3 \bmod 4$ (i.e., Blum integers). This is a 4-to-1 mapping on $\mathbb{Z}_N^*$, and the construction is obtained by embedding additional information in the output that reduces the number of preimages to either two (these are the lossy functions) or one (these are the injective functions) in a computationally indistinguishable manner. The injective trapdoor functions in our construction can be viewed as a permutation version of the Rabin trapdoor function [34].

In our initial construction (Section 3.1) the functions are defined over an index-dependent domain $\mathbb{Z}_N^*$ and have one bit of lossiness. However, lossy trapdoor functions in a collection are required to share the same domain; i.e., the domain should depend only on the security parameter. Our second construction (Section 3.2) overcomes this difficulty with a simple domain extension, which results in lossiness of $\log_2(4/3)$ bits.

We start with a definition. For any odd positive integer N, we denote by $\mathsf{JS}_N : \mathbb{Z} \to \{-1, 0, 1\}$ the Jacobi symbol mod N. We define functions $h, j : \mathbb{Z} \to \{0, 1\}$ as follows:

$$h(x) = \begin{cases} 1, & \text{if } x > N/2 \\ 0, & \text{if } x \le N/2 \end{cases}$$

$$j(x) = \begin{cases} 1, & \text{if } \mathsf{JS}_N(x) = -1 \\ 0, & \text{if } \mathsf{JS}_N(x) = 0 \text{ or } 1 \end{cases}$$

We define h and j on $\mathbb{Z}_N$ by representing elements of $\mathbb{Z}_N$ as integers between 0 and $N - 1$.

Fact 3.1. *Let $N = PQ$ where $P \equiv Q \equiv 3 \bmod 4$, and let $y \in \mathbb{Z}_N^*$ be a quadratic residue. Denote by $\{\pm x_0, \pm x_1\}$ the distinct solutions of the equation $x^2 = y \bmod N$. Then $\mathsf{JS}_P(-1) = \mathsf{JS}_Q(-1) = -1$, and therefore*

1. $\mathsf{JS}_N(x_0) = \mathsf{JS}_N(-x_0)$ *and* $\mathsf{JS}_N(x_1) = \mathsf{JS}_N(-x_1)$*,*

2. $\mathsf{JS}_N(x_0) = -\mathsf{JS}_N(x_1)$.

In particular, the four square roots of y take all four values of $(h(x), j(x))$.

3.1 A lossy trapdoor function with index-dependent domains

We define a 5-tuple $\mathcal{F} = (\mathsf{G}_0, \mathsf{G}_1, \mathsf{S}, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.3) as follows:

1. **Sampling a lossy function:** On input 1^n the algorithm G_0 chooses an n-bit modulus $N = PQ$, where $P \equiv Q \equiv 3 \bmod 4$ are random $n/2$-bit prime numbers. Then it chooses random $r \in \mathbb{Z}_N^*$ such that $\mathsf{JS}_N(r) = -1$, and a random $s \in \mathbb{Z}_N^*$ such that $\mathsf{JS}_N(s) = 1$ and s is a *quadratic residue*. The function index is $\sigma = (N, r, s)$, and the function f_σ is defined on the domain $D_\sigma = \{1, \ldots, N - 1\}$.

2. **Sampling an injective function:** On input 1^n the algorithm G_1 chooses an n-bit modulus $N = PQ$, where $P \equiv Q \equiv 3 \bmod 4$ are random $n/2$-bit prime numbers. Then it chooses random $r \in \mathbb{Z}_N^*$ such that $\mathsf{JS}_N(r) = -1$, and a random $s \in \mathbb{Z}_N^*$ such that $\mathsf{JS}_N(s) = 1$ and s is a *quadratic non-residue*. The function index is $\sigma = (N, r, s)$, the trapdoor is $\tau = (P, Q)$, and the function f_σ is defined on the domain $D_\sigma = \{1, \ldots, N - 1\}$.

3. **Sampling an input:** Given a function index $\sigma = (N, r, s)$, the algorithm S outputs a uniformly distributed $x \in D_\sigma = \{1, \ldots, N - 1\}$.

4. **Evaluation:** Given a function index $\sigma = (N, r, s)$ and $x \in D_\sigma = \{1, \ldots, N - 1\}$, the algorithm outputs
$$f_{N,r,s}(x) = x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N.$$

5. **Inversion:** Given a description of an injective function $\sigma = (N, r, s)$ together with its trapdoor $\tau = (P, Q)$ and $y = f_{N,r,s}(x)$, the algorithm F^{-1} retrieves x as follows.
   (a) Find $j(x)$ by computing $\mathsf{JS}_N(f_{N,r,s}(x))$ (note that $\mathsf{JS}_N(f_{N,r,s}(x)) = \mathsf{JS}_N(x)$). Let $y' = yr^{-j(x)}$.
   (b) Find $h(x)$ by checking whether y' is a quadratic residue mod N (note that $h(x) = 1$ if and only if y' is not a quadratic residue). Let $y'' = y's^{-h(x)}$.

(c) Find all square roots of y'' in $\mathbb{Z}_N$, and output the one that agrees with both $j(x)$ and $h(x)$. (We use Fact 3.1 if $y'' \in \mathbb{Z}_N^*$, and note that if $1 < \gcd(y'', N) < N$, then y'' has two square roots that are negatives of each other.)

We now prove that the above construction is indeed lossy based on the *quadratic residuosity assumption*. Let $\mathcal{J}_N = \{x \in \mathbb{Z}_N^* : \mathsf{JS}_N(x) = 1\}$, and let $\mathcal{Q}_N$ be the subgroup of squares in $\mathbb{Z}_N^*$. Then the quadratic residuosity assumption states that the two distributions obtained by sampling uniformly at random from $\mathcal{Q}_N$ and from $\mathcal{J}_N \setminus \mathcal{Q}_N$ are computationally indistinguishable.

Theorem 3.2. *Under the quadratic residuosity assumption, $\mathcal{F}$ is a collection of $(n, 1)$-lossy trapdoor functions with index-dependent domains.*

Proof. First, it follows from the correctness of the inversion algorithm that G_1 outputs permutations on the set $D_\sigma = \{1, \ldots, N - 1\}$. Next, we claim that G_0 outputs functions that are 2-to-1 on $\{1, \ldots, N - 1\}$. Suppose $y \in \mathcal{Q}_N$. Since s is a quadratic residue, Fact 3.1 implies that for each $(\eta, \iota) \in \{0, 1\}^2$ there is an $x_{\eta,\iota}$ satisfying

$$x_{\eta,\iota}^2 = ys^{-\eta}, \quad h(x_{\eta,\iota}) = \eta, \quad j(x_{\eta,\iota}) = \iota.$$

Then for each $\eta \in \{0, 1\}$ we have $f_{N,r,s}(x_{\eta,0}) = y$ and $f_{N,r,s}(x_{\eta,1}) = ry$. Thus each element in the set $\mathcal{Q}_N \cup r\mathcal{Q}_N$ has at least two preimages in $\mathbb{Z}_N^*$, and since this set has cardinality half that of $\mathbb{Z}_N^*$ we deduce that $f_{N,r,s}$ is 2-to-1 on $\mathbb{Z}_N^*$. A similar argument shows that every square in the ideal $P\mathbb{Z}_N$ has two preimages in $P\mathbb{Z}_N$, and the same for the ideal $Q\mathbb{Z}_N$. Since $\{1, \ldots, N - 1\} = \mathbb{Z}_N^* \cup P\mathbb{Z}_N \cup Q\mathbb{Z}_N$, the function $f_{N,r,s}$ is 2-to-1 on $D_\sigma = \{1, \ldots, N - 1\}$.

Descriptions of lossy functions and injective functions differ only in the element s, which is a random element of the subgroup of $\mathbb{Z}_N^*$ with Jacobi symbol 1 that is a quadratic residue in the lossy case and a quadratic non-residue in the injective case. Therefore, the quadratic residuosity assumption implies that lossy functions are computationally indistinguishable from injective functions. □

Note that since security does not depend on the distribution of r, the size of the function index σ can be reduced by choosing r to be the smallest positive integer such that $\mathsf{JS}_N(r) = -1$.

3.2 A lossy trapdoor function

We now show how to extend our previous construction to be defined over a common domain that only depends on the security parameter. We define a 4-tuple $\mathcal{F} = (\mathsf{G}_0, \mathsf{G}_1, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.1) as follows:

1. **Sampling a lossy function** and **sampling an injective function** are done as in Section 3.1, with the difference that now the function f_σ is defined on the domain $\{0, 1\}^n$.

2. **Evaluation:** Given a function index $\sigma = (N, r, s)$ and $x \in \{0, 1\}^n$, the algorithm F interprets x as an integer in the set $\{1, \ldots, 2^n\}$ and outputs

$$f_{N,r,s}(x) = \begin{cases} x^2 \cdot r^{j(x)} \cdot s^{h(x)} \bmod N, & \text{if } 1 \le x < N \\ x, & \text{if } N \le x \le 2^n \end{cases}$$

3. **Inversion:** Given a description of an injective function $\sigma = (N, r, s)$ together with its trapdoor $\tau = (P, Q)$ and $y = f_{N,r,s}(x)$, the algorithm F^{-1} retrieves x as follows. If $N \leq y \leq 2^n$, then the algorithm outputs y. Otherwise, it uses the method described in the inversion algorithm from Section 3.1.

Theorem 3.3. *Under the quadratic residuosity assumption, $\mathcal{F}$ is a collection of $(n, \log_2(4/3))$-lossy trapdoor functions.*

Proof. First, it follows from the correctness of the inversion algorithm that G_1 outputs permutations on the set $\{1, \ldots, 2^n\}$. Next, as already shown in the proof of Theorem 3.2, G_0 outputs functions that are 2-to-1 on the set $\{1, \ldots, N-1\}$. Since N is an n-bit modulus (i.e., $2^{n-1} < N < 2^n$), the lossy functions are 2-to-1 on at least half of their domain, which implies that their image is of size at most $3/4 \cdot 2^n = 2^{n-\log_2(4/3)}$. Finally, as in the proof of Theorem 3.2, the quadratic residuosity assumption implies that lossy functions are computationally indistinguishable from injective functions. $\square$

4 A Construction based on the *e*th Power Residuosity Assumption

In this section we generalize the construction of Section 3 to higher order power residues. Instead of using the squaring function mod N, we use the powering function $x \mapsto x^e \bmod N$, where N is a product of two primes congruent to 1 mod e. This is an e^2-to-1 mapping on $\mathbb{Z}_N^*$, and the construction is obtained by embedding additional information in the output that reduces the number of preimages to either e (these are the lossy functions) or 1 (these are the injective functions) in a computationally indistinguishable manner, resulting in $\log_2(e)$ bits of lossiness.

The security of our construction follows from the *eth power residuosity assumption*, which is a generalization of the standard quadratic residuosity assumption. This assumption, as well as our system, requires us to define the *eth power residue symbol*, a generalization of the Legendre and Jacobi symbols. We review the basic facts here; for further details see [23, Chapter 14] for the number theory context or [7, 22] for cryptographic applications.

4.1 Mathematical background

Let $e \geq 2$ be an integer, and let $N = pq$ be a product of two primes congruent to 1 mod e. We say that $x \in \mathbb{Z}_N^*$ is an *eth power residue mod N* if there is a $y \in \mathbb{Z}_N^*$ such that $y^e \equiv x \bmod N$. Let $\zeta_e \in \overline{\mathbb{Q}}$ be a primitive eth root of unity, let K be the number field $\mathbb{Q}(\zeta_e)$, and let $\mathcal{O}_K = \mathbb{Z}[\zeta_e]$ be the ring of integers in K. For an ideal $\mathfrak{a} \subset \mathcal{O}_K$, the *norm* of $\mathfrak{a}$ is $\mathcal{N}(\mathfrak{a}) = [\mathcal{O}_K : \mathfrak{a}]$. We define the eth power residue symbol as follows:

Definition 4.1. Let e and K be as above, and let $\mathfrak{p}$ be a prime ideal of $\mathcal{O}_K$ not containing e. For $x \in \mathcal{O}_K$, the *eth power residue symbol of x mod $\mathfrak{p}$*, denoted by $(\frac{x}{\mathfrak{p}})_e$, is defined to be

$$\left(\frac{x}{\mathfrak{p}}\right)_e := \begin{cases} 0 & \text{if } x \in \mathfrak{p} \\ \zeta_e^i & \text{if } x \notin \mathfrak{p}, \end{cases}$$

where i is the unique integer mod e such that $\zeta_e^i \equiv x^{(\mathcal{N}(\mathfrak{p})-1)/e} \pmod{\mathfrak{p}}$. (One can show that this definition is independent of the choice of the root of unity ζ_e [37, §III.1].)

We extend to non-prime ideals and single elements in the obvious way: if $\mathfrak{a} = \prod_i \mathfrak{p}_i$ is any ideal of $\mathcal{O}_K$ not containing any prime factor of e, and a is any element of $\mathcal{O}_K$, we define

$$\left(\frac{x}{\mathfrak{a}}\right)_e := \prod_i \left(\frac{x}{\mathfrak{p}_i}\right)_e \quad \text{and} \quad \left(\frac{x}{a}\right)_e := \left(\frac{x}{a\mathcal{O}_K}\right)_e.$$

The power residue symbol shares some important properties with the Jacobi symbol that it generalizes. First, if $\mathfrak{p}$ is prime, then $(\frac{x}{\mathfrak{p}})_e = 1$ if and only if x is an eth power mod $\mathfrak{p}$. Second, the symbol is multiplicative in both components: for $x, y \in \mathcal{O}_K$ and ideals $\mathfrak{a}, \mathfrak{b} \subset \mathcal{O}_K$, we have

$$\left(\frac{xy}{\mathfrak{a}}\right)_e = \left(\frac{x}{\mathfrak{a}}\right)_e \left(\frac{y}{\mathfrak{a}}\right)_e \quad \text{and} \quad \left(\frac{x}{\mathfrak{a}\mathfrak{b}}\right)_e = \left(\frac{x}{\mathfrak{a}}\right)_e \left(\frac{x}{\mathfrak{b}}\right)_e. \tag{4.1}$$

For any $x \in \mathcal{O}_K$ and ideal $\mathfrak{a}$ relatively prime to e, Squirrel [37] gives an algorithm for computing $\left(\frac{x}{\mathfrak{a}}\right)_e$ that runs in time polynomial in $\log(\mathcal{N}(\mathfrak{a}))$, $\log(\mathcal{N}(x))$, and e. Boneh and Horwitz [7, 22] give an alternative polynomial-time algorithm for the case where $\mathfrak{a}$ is principal. Both algorithms use *Eisenstein reciprocity* [23, p.207], a generalization of quadratic reciprocity.

To define our lossy trapdoor function, we generalize the functions $h(x)$ and $j(x)$ of Section 3 to higher residues. These functions will allow us to recover unique preimages of the eth powering map mod N.

When trying to generalize the function $j(x)$, we are immediately confronted with an obstacle: if x is an integer then $\left(\frac{x}{N}\right)_e$ is always 1 when e is odd, and is ± 1 when e is even, so if $e > 2$ the symbol does not contain enough information about x. To get around this problem, we find ideals $\mathfrak{a}_i \subset \mathcal{O}_K$ of norm N such that $N\mathcal{O}_K = \prod \mathfrak{a}_i$, and use the symbol $(\frac{x}{\mathfrak{a}_1})_e$. The following lemma shows that these ideals can be computed easily, given an element $\mu \in \mathbb{Z}_N^*$ that is a primitive eth root of unity mod p and mod q. Such a μ is said to be a *nondegenerate* primitive eth root of unity mod N. It is believed that revealing such a μ does not make factoring N any easier. This is clearly the case when $e = 2$, since the only such μ is -1. (See [7, 22, 8] for other contexts where this assumption is used.)

Lemma 4.2. *Let e be a positive integer, $N = pq$ be a product of two primes p, q with $p \equiv q \equiv 1 \bmod e$, and $\mathcal{O}_K = \mathbb{Z}[\zeta_e]$. Let $\mu \in \mathbb{Z}_N^*$ be a nondegenerate primitive eth root of unity. For each i in $1, \ldots, e$ with $\gcd(i, e) = 1$, let $\mathfrak{a}_i = N\mathcal{O}_K + (\zeta_e - \mu^i)\mathcal{O}_K$. Then $\mathcal{N}(\mathfrak{a}_i) = N$ for all i, and we have*

$$N\mathcal{O}_K = \prod_{(i,e)=1} \mathfrak{a}_i. \tag{4.2}$$

Proof. Since $p \equiv q \equiv 1 \pmod e$, the primes p and q split completely in $\mathcal{O}_K$ [28, Corollary I.10.4]. Specifically, if $\mathfrak{p}_i = p\mathcal{O}_K + (\zeta_e - \mu^i)\mathcal{O}_K$ and $\mathfrak{q}_i = q\mathcal{O}_K + (\zeta_e - \mu^i)\mathcal{O}_K$, then we have [28, Proposition I.8.3]

$$p\mathcal{O}_K = \prod_{(i,e)=1} \mathfrak{p}_i \quad \text{and} \quad q\mathcal{O}_K = \prod_{(i,e)=1} \mathfrak{q}_i. \tag{4.3}$$

Furthermore, we have $\mathcal{N}(\mathfrak{p}_i) = p$ and $\mathcal{N}(\mathfrak{q}_i) = q$ for all i. It follows immediately that

$$\mathfrak{p}_i\mathfrak{q}_i = N\mathcal{O}_K + (\zeta_e - \mu^i)(p + q + \zeta_e - \mu^i)\mathcal{O}_K \subset \mathfrak{a}_i$$

and that $\mathfrak{a}_i \subset \mathfrak{p}_i \cap \mathfrak{q}_i$. Since $\mathfrak{p}_i$ and $\mathfrak{q}_i$ are relatively prime, we have $\mathfrak{p}_i \cap \mathfrak{q}_i = \mathfrak{p}_i\mathfrak{q}_i$, and thus $\mathfrak{a}_i = \mathfrak{p}_i\mathfrak{q}_i$ for all i. The decomposition of N in (4.2) now follows from the decompositions of p and q in (4.3), and we have $\mathcal{N}(\mathfrak{a}_i) = \mathcal{N}(\mathfrak{p}_i)\mathcal{N}(\mathfrak{q}_i) = N$. $\square$

Note that when $e = 2$ we have $\mu = -1$, $K = \mathbb{Q}$, and $\mathfrak{a}_1 = N\mathbb{Z}$.

We now define a function $J(x) : \mathbb{Z} \to \mathbb{Z}_e$ that generalizes the function $j(x)$ of Section 3. Since the function will depend on our choice of a primitive eth root of unity μ, we make this dependence explicit in the notation. For a fixed μ, let $\mathfrak{a} = N\mathcal{O}_K + (\zeta_e - \mu)\mathcal{O}_K$ be the ideal $\mathfrak{a}_1$ from Lemma 4.2, and define

$$J_\mu(x) = \begin{cases} 0, & \text{if } \gcd(x, N) \neq 1, \\ i, & \text{if } \gcd(x, N) = 1 \text{ and } \left(\frac{x}{\mathfrak{a}}\right)_e = \zeta_e^i. \end{cases}$$

It follows from (4.1) that if $x, y \in \mathbb{Z}_N^*$, then $J_\mu(xy) = J_\mu(x) + J_\mu(y)$. If $\mathfrak{a}$ is principal, then a generator can be computed in time polynomial in $\log N$ and the discriminant of $K = \mathbb{Q}(\zeta_e)$ (using e.g. [10, Algorithm 6.5.10]), and thus the algorithm of Boneh and Horwitz ([7, Appendix B] or [22, Section 4.2.1]) can be used to compute $\left(\frac{x}{\mathfrak{a}}\right)_e$ in this case.

The function $H(x)$ generalizes the function $h(x)$ of Section 3 and is used to distinguish eth roots that have the same value of $J(x)$. Specifically, we define $H_\mu(x) : \mathbb{Z} \to \mathbb{Z}_e$ by

$$H_\mu(x) := (i \in \mathbb{Z}_e \text{ such that } x\mu^i \bmod N \text{ has minimal representative in } [0, N-1]).$$

If $e = 2$, then since $\mu = -1$ the function simply determines whether $x \bmod N$ is greater or less than $N/2$.

The fact that J_μ and H_μ can be used to distinguish preimages of the eth powering map is a consequence of the following proposition, which generalizes Fact 3.1.

Proposition 4.3. *Let e be a positive square free integer and $\mathcal{O}_K = \mathbb{Z}[\zeta_e]$. Let $N = pq$ where $p \equiv q \equiv 1 \bmod e$. Suppose that for every prime $f \mid e$ we have $p, q \not\equiv 1 \bmod f^2$. Then there is a $\mu \in \mathbb{Z}$ such that*

1. *μ is a nondegenerate primitive eth root of unity mod N,*
2. *$\left(\dfrac{\mu}{\mathfrak{a}_i}\right)_e = 1$ for every ideal $\mathfrak{a}_i \subset \mathcal{O}_K$ as in Lemma 4.2.*

Furthermore, if $y \in \mathbb{Z}_N^*$ is an eth power residue and μ has properties (1) and (2), then the e^2 solutions to $y = x^e \bmod N$ take on all e^2 values of $(H_\mu(x), J_\mu(x))$.

Proof. The assumption $p \equiv 1 \pmod{e}$ implies that there is a primitive eth root of unity $\mu_p \in \mathbb{F}_p$. For any prime $f \mid e$, the assumption $p \not\equiv 1 \pmod{f^2}$ implies that $x^f - \mu_p$ has no solutions in $\mathbb{F}_p$, for such a solution would be a primitive efth root of unity in $\mathbb{F}_p$, which doesn't exist. It follows that $(\frac{\mu_p}{\mathfrak{p}_1})_e$ is a primitive eth root of unity ζ_e^a (with $(a, e) = 1$). Similarly, there is a primitive eth root of unity $\mu_q \in \mathbb{F}_q$ such that $(\frac{\mu_q}{\mathfrak{q}_1})_e = \zeta_e^b$, with $(b, e) = 1$. Let a', b' be such that $aa' \equiv bb' \equiv 1 \pmod{e}$. Then by (4.1) we have

$$\left(\frac{\mu_p^{a'}}{\mathfrak{p}_1}\right)_e = \zeta_e, \quad \left(\frac{\mu_q^{-b'}}{\mathfrak{q}_1}\right)_e = \zeta_e^{-1}.$$

By the Chinese remainder theorem, there is an integer μ that is congruent to $\mu_p^{a'} \bmod p$ and $\mu_q^{-b'} \bmod q$, and it follows from the above argument that $(\frac{\mu}{\mathfrak{a}_1})_e = 1$. To show the same holds for all $\mathfrak{a}_i$, we note that for each i there is some automorphism σ of K fixing $\mathbb{Q}$ such that $\mathfrak{a}_i = \mathfrak{a}_1^\sigma$. Since $\mu \in \mathbb{Z}$, the result $(\frac{\mu}{\mathfrak{a}_i})_e = 1$ now follows from Galois-equivariance of the power residue symbol.

For the "furthermore" statement, let μ be as constructed above, and let $\alpha_1, \ldots, \alpha_e$ be integers such that $\{\alpha_i \mu^j\}_{i,j=1}^e$ is a complete set of solutions to $y = x^e \bmod N$. Then it is easy to see that for fixed i we have $J_\mu(\alpha_i \mu^j) = J_\mu(\alpha_i \mu^{j'})$ for all j, j', and $H_\mu(\alpha_i \mu^j) \neq H_\mu(\alpha_i \mu^{j'})$ for all $j \neq j'$. It thus suffices to show that $J_\mu(\alpha_i) \neq J_\mu(\alpha_{i'})$ for $i \neq i'$.

Suppose that $i \neq i'$, and let $\mathfrak{a} = \mathfrak{p}\mathfrak{q}$ be the prime factorization of $\mathfrak{a}$ in $\mathcal{O}_K$. Then there are unique $k, l \in \mathbb{Z}_e$ such that $\alpha_{i'} = \alpha_i \mu^k \bmod p$ and $\alpha_{i'} = \alpha_i \mu^l \bmod q$. We thus have

$$\left(\frac{\alpha_{i'}}{\mathfrak{a}}\right)_e = \left(\frac{\alpha_i}{\mathfrak{a}}\right)_e \left(\frac{\mu}{\mathfrak{p}}\right)_e^k \left(\frac{\mu}{\mathfrak{q}}\right)_e^l = \left(\frac{\alpha_i}{\mathfrak{a}}\right)_e \left(\frac{\mu}{\mathfrak{q}}\right)_e^{l-k}.$$

If $\left(\frac{\alpha_{i'}}{\mathfrak{a}}\right)_e = \left(\frac{\alpha_i}{\mathfrak{a}}\right)_e$ then $k = l \pmod{e}$ and thus $\alpha_{i'} = \alpha_i \mu^k \bmod N$, which contradicts our assertion that $\{\alpha_i \mu^j\}_{i,j=1}^e$ is a complete set of solutions to $y = x^e \bmod N$. We conclude that $J_\mu(\alpha_i) \neq J_\mu(\alpha_{i'})$ for $i \neq i'$. $\square$

4.2 A lossy trapdoor function with index-dependent domains

We assume that the square free integer e and a primitive eth root of unity $\zeta_e \in \overline{\mathbb{Q}}$ are fixed, and we let $K = \mathbb{Q}(\zeta_e)$ and $\mathcal{O}_K = \mathbb{Z}[\zeta_e]$. We define a 5-tuple $\mathcal{F} = (\mathsf{G}_0, \mathsf{G}_1, \mathsf{S}, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.3) as follows:

1. **Sampling a lossy function:** On input 1^n the algorithm G_0 chooses an n-bit modulus $N = pq$, where p and q are random $n/2$-bit prime numbers with $p, q \equiv 1 \pmod{e}$ and $p, q \not\equiv 1 \bmod f^2$ for all primes $f \mid e$. It chooses a random nondegenerate primitive eth root of unity $\mu \in \mathbb{Z}_N$ such that $(\frac{\mu}{\mathfrak{a}})_e = 1$, where $\mathfrak{a} \subset \mathcal{O}_K$ is the ideal $N\mathcal{O}_K + (\zeta_e - \mu)\mathcal{O}_K$. Then it chooses random $r \in \mathbb{Z}_N^*$ such that $J_\mu(r) = 1$, and a random $s \in \mathbb{Z}_N^*$ such that $J_\mu(s) = 0$ and s is an *eth power residue.* The function index is $\sigma = (N, \mu, r, s)$ and the function f_σ is defined on the domain $D_\sigma = \{1, \ldots, N-1\}$.

2. **Sampling an injective function:** On input 1^n the algorithm G_1 chooses an n-bit modulus $N = pq$ and a primitive eth root of unity $\mu \bmod N$ as above. Then it chooses random $r \in \mathbb{Z}_N^*$ such that $J_\mu(r) = 1$, and a random $s \in \mathbb{Z}_N^*$ such that $J_\mu(s) = 0$ and s is *not an eth power residue.* The function index is $\sigma = (N, \mu, r, s)$, the trapdoor is $\tau = (p, q)$, and the function f_σ is defined on the domain $D_\sigma = \{1, \ldots, N-1\}$.

3. **Sampling an input:** Given a function index $\sigma = (N, \mu, r, s)$, the algorithm S outputs a uniformly distributed $x \in D_\sigma = \{1, \ldots, N-1\}$.

4. **Evaluation:** Given a function index $\sigma = (N, \mu, r, s)$ and nonzero $x \in D_\sigma = \{1, \ldots, N-1\}$, the algorithm F outputs
$$f_{N,\mu,r,s}(x) = x^e \cdot r^{J_\mu(x)} \cdot s^{H_\mu(x)} \bmod N.$$

5. **Inversion:** Given a description of an injective function $\sigma = (N, \mu, r, s)$ together with its trapdoor $\tau = (p, q)$ and $y = f_{N,\mu,r,s}(x)$, the algorithm F^{-1} retrieves x as follows.

   (a) Find $J_\mu(x)$ by computing $J_\mu(f_{N,\mu,r,s}(x))$ (note that $J_\mu(f_{N,\mu,r,s}(x)) = J_\mu(x)$). Let $y' = yr^{-J_\mu(x)}$.

(b) Find $H_\mu(x)$ by computing the integer i such that $y's^{-i}$ is an eth power residue mod N (note that this i is equal to $H_\mu(x)$). Let $y'' = y's^{-H_\mu(x)}$.

(c) Find all solutions to $x^e = y''$ in $\mathbb{Z}_N$, and output the one that agrees with both $J_\mu(x)$ and $H_\mu(x)$. (We use Proposition 4.3 if $y'' \in \mathbb{Z}_N^*$, and note that if $1 < \gcd(y'', N) < N$, then there are e solutions, indexed by $H_\mu(x)$.)

We will show that this construction is indeed lossy based on the *eth power residuosity assumption*. We first state this assumption, and then give the theorem.

Definition 4.4. Let N, μ be as computed by G_0 or G_1 above. Let $\mathcal{J}_N = \{x \in \mathbb{Z}_N^* : J_\mu(x) = 0\}$, and let $\mathcal{E}_N$ be the subgroup of eth powers in $\mathbb{Z}_N^*$. We say that the *eth power residuosity assumption* holds for (N, μ) if the two distributions obtained by sampling uniformly at random from $\mathcal{E}_N$ and from $\mathcal{J}_N \setminus \mathcal{E}_N$ are computationally indistinguishable, where the adversary has access to both N and μ.

One can show that the set $\mathcal{J}_N$ is independent of the choice of ζ_e and of μ, given the constraint that μ satisfies the conditions of Proposition 4.3.

Theorem 4.5. *Under the eth power residuosity assumption, $\mathcal{F}$ is a collection of $(n, \log_2(e))$-lossy trapdoor functions with index-dependent domains.*

The proof of Theorem 4.5 is entirely analogous to the proof of Theorem 3.2, so we do not repeat the details.

4.3 Extending to large values of e

Our description of the functions J_μ and H_μ above suggests that computing these functions always takes time polynomial in e. If this is the case, then the lossy trapdoor function defined above can only be efficiently computed when e is logarithmic in N, and the lossiness is limited to being logarithmic in the security parameter n. However, if e is a product of many small primes, then we can modify the function to achieve lossiness that is a constant fraction of the security parameter n.

Suppose f is a prime dividing e. To compute the eth power residue symbol we use the following "compatibility" identity that holds for any ideal $\mathfrak{a} \subset \mathbb{Z}[\zeta_e]$ (see Appendix B for a proof):

$$\left(\frac{x}{\mathfrak{a} \cap \mathbb{Z}[\zeta_f]}\right)_f = \left(\frac{x}{\mathfrak{a}}\right)_e^{e/f}. \tag{4.4}$$

While the power residue symbol is independent of the choice of ζ_e used to compute it, the function J_μ depends on this choice. We thus create a system of compatible roots of unity by fixing ζ_e and setting $\zeta_f = \zeta_e^{e/f}$ for each $f \mid e$. If we use $J_\mu(x, r)$ to denote the function J_μ defined above relative to the rth power residue symbol, then the identity (4.4) and our system of compatible roots of unity implies that

$$J_\mu(x, e) \equiv J_\mu(x, f) \bmod f.$$

It follows that if e is square-free, then we can compute $J_\mu(x, e)$ by computing $J_\mu(x, f)$ for each prime $f \mid e$ and applying the Chinese remainder theorem.

We cannot use similar techniques to compute the function H_μ, but we can define a modified function $\hat{H}_\mu$ that has the necessary properties and can be computed via the Chinese remainder theorem. We first let $H_\mu(x, r)$ denote the function H_μ defined relative to the rth power residue symbol, and then define

$$\hat{H}_\mu(x, e) := (c \in \mathbb{Z}_e \text{ such that for all prime powers } f \mid e,\ c \equiv H_\mu(x, f) \bmod f)\,.$$

It is straightforward to show that the result of Proposition 4.3 still holds when we replace H_μ with $\hat{H}_\mu$.

We can thus carry out the construction of Section 4.2, replacing the function H_μ with $\hat{H}_\mu$, to obtain lossy trapdoor function with index-dependent domains and $\log_2(e)$ bits of lossiness. When e is a product of many small primes, the lossiness can be a constant fraction of the security parameter n. Note that since e is a publicly known factor of $\varphi(N)$, we require $e \le N^{1/4-\varepsilon}$ in order to ensure that Coppersmith's method [11] for finding small roots of a univariate polynomial modulo an unknown divisor of N cannot be used to efficiently factor N.

4.4 Lossy trapdoor functions

We can apply the technique of Section 3.2 to define functions with domain $\{1, \dots, 2^n\}$, i.e., depending only on the security parameter. The same analysis as in the case $e = 2$ shows that we obtain $\log_2(2e/(e+1))$ bits of lossiness, which is never greater than 1 even for large e.

We can do better by fixing some m and defining the functions over $\{1, \cdots, 2^{n+m}\}$. We apply the modified powering function $x \mapsto x^e \cdot r^{J_\mu(x)} \cdot s^{H_\mu(x)} \bmod N$ to each copy of $\mathbb{Z}_N$ in this domain, i.e., to the sets $\{aN + 1, \dots, (a+1)N - 1\}$ for $a = 0, \dots, \lfloor 2^{n+m}/N \rfloor - 1$. For the remainder of the domain we let the function be the identity. It is easy to see that for sufficiently large m these functions are e-to-1 on almost all of the domain, so we can obtain almost $\log_2(e)$ bits of lossiness. (In fact, one can show that if $m \ge \log_2(e) - 1$, then we obtain a collection of $(n, \log_2(e) - e \cdot 2^{-m})$-lossy trapdoor functions.)

5 A Construction based on the Composite Residuosity Assumption

Our construction is based on the Damgård-Jurik encryption scheme [13] with additional insights by Damgård and Nielsen [14, 15]. We begin with a brief description of the Damgård-Jurik scheme, and then present our constructions of lossy trapdoor functions and all-but-one lossy trapdoor functions.

5.1 The Damgård-Jurik encryption scheme

Damgård and Jurik [13] proposed an encryption scheme based on computations in the group $\mathbb{Z}_{N^{s+1}}$, where $N = PQ$ is an RSA modulus and $s \ge 1$ is an integer (it contains Paillier's encryption scheme [31] as a special case by setting $s = 1$). Consider a modulus $N = PQ$ where P and Q are odd primes and $\gcd(N, \phi(N)) = 1$ (when P and Q are sufficiently large and randomly chosen, this will be satisfied except with negligible probability). We call such a modulus N *admissible* in the following discussion. For such an N, the group $\mathbb{Z}^*_{N^{s+1}}$ as a multiplicative group is a direct product $G \times H$, where G is cyclic of order N^s and H is isomorphic to $\mathbb{Z}^*_N$.

Theorem 5.1 ([13]). *For any admissible N and $s < \min\{P, Q\}$, the map $\psi_s : \mathbb{Z}_{N^s} \times \mathbb{Z}^*_N \to \mathbb{Z}^*_{N^{s+1}}$ defined by $\psi_s(x, r) = (1+N)^x r^{N^s} \bmod N^{s+1}$ is an isomorphism of abelian groups. In particular,* *we have*

$$\psi_s(x_1 + x_2 \bmod N^s, r_1 r_2 \bmod N) = \psi_s(x_1, r_1) \cdot \psi_s(x_2, r_2) \bmod N^{s+1} \enspace .$$

Given $\lambda(N) = \mathrm{lcm}(P-1, Q-1)$*, there is a polynomial-time algorithm that inverts the function* ψ_s*.*

The following describes the Damgård-Jurik encryption scheme:

- **Key generation:** On input 1^n choose an admissible n-bit modulus $N = PQ$. The public key is (N, s) and the secret key is $\lambda = \mathrm{lcm}(P-1, Q-1)$.
- **Encryption:** Given a message $m \in \mathbb{Z}_{N^s}$ and the public key (N, s), choose a random $r \in \mathbb{Z}_N^*$ and output $\mathcal{E}(m) = (1+N)^m r^{N^s} \bmod N^{s+1}$.
- **Decryption:** Given a ciphertext $c \in \mathbb{Z}_{N^{s+1}}$ and the secret key λ, apply the inversion algorithm of Theorem 5.1 to compute $\psi_s^{-1}(c) = (m, r)$ and output m.

The semantic security of the scheme (for any $s \geq 1$) is based on the *decisional composite residuosity assumption*: namely, that any probabilistic polynomial-time algorithm that receives as input an n-bit RSA modulus N cannot distinguish a random element in $\mathbb{Z}_{N^2}^*$ from a random N-th power in $\mathbb{Z}_{N^2}^*$ with probability a non-negligible function of n. We refer the reader to [13] for a more formal statement of the decisional composite residuosity assumption and for the proof of security.

5.2 A lossy trapdoor function with index-dependent domains

Each function in our construction is described by a pair (N, c), where N is an n-bit modulus as above, and $c \in \mathbb{Z}_{N^{s+1}}$. For the injective functions c is a random Damgård-Jurik encryption of 1, and for the lossy functions c is a random encryption of 0. The semantic security of the encryption scheme guarantees that the two collections of functions are computationally indistinguishable. In order to evaluate a function $f_{(N,c)}$ on an input $(x, y) \in \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$ we compute $f_{(N,c)}(x) = c^x y^{N^s} \bmod \mathbb{Z}_{N^{s+1}}$. For an injective function $f_{(N,c)}$ it holds that $f_{(N,c)}(x, y)$ is an encryption of x (where the randomness of this ciphertext depends on x and y), and using the secret key it is possible to retrieve both x and y. For a lossy function $f_{(N,c)}$ it holds that $f_{(N,c)}(x, y)$ is an encryption of 0, and in this case most of the information in the input is lost.

Given any polynomial $s = s(n)$ we define a 5-tuple $\mathcal{F}_s = (\mathsf{G}_0, \mathsf{G}_1, \mathsf{S}, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.3) as follows:

1. **Sampling a lossy function:** On input 1^n the algorithm G_0 chooses an admissible n-bit modulus $N = PQ$. Then, it chooses a random $r \in \mathbb{Z}_N^*$ and lets $c = r^{N^s} \bmod N^{s+1}$. The description of the function is $\sigma = (N, c)$ and the function f_σ is defined on the domain $D_\sigma = \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$.
2. **Sampling an injective function:** On input 1^n the algorithm G_1 chooses an admissible n-bit modulus $N = PQ$. Then, it chooses a random $r \in \mathbb{Z}_N^*$ and lets $c = (1+N)r^{N^s} \bmod N^{s+1}$. The description of the function is $\sigma = (N, c)$, the trapdoor is $\tau = (\lambda, r)$, where $\lambda = \mathrm{lcm}(P-1, Q-1)$, and the function f_σ is defined on the domain $D_\sigma = \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$.
3. **Sampling an input:** Given a description of a function (N, c) the algorithm S outputs a uniformly distributed pair $(x, y) \in \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$.

4. **Evaluation:** Given a description of a function (N,c) and an input $(x,y) \in \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$, the algorithm F outputs $c^x y^{N^s} \bmod N^{s+1}$.

5. **Inversion:** Given a description of an injective function (N,c) together with its trapdoor (λ, r) and $z \in \mathbb{Z}_{N^{s+1}}$, the algorithm F^{-1} invokes the inversion algorithm provided by Theorem 5.1 to compute $\psi_s^{-1}(z) = (x, r^x y)$, and then recovers x and y.

Theorem 5.2. *Under the decisional composite residuosity assumption, for any polynomial $s = s(n)$ it holds that $\mathcal{F}_s$ is a collection of $((n-1)(s+1), (n-1)s-1)$-lossy trapdoor functions with index-dependent domains.*

Proof. Theorem 5.1 guarantees that the injective functions can be efficiently inverted using their trapdoor information. The semantic security of the Damgård-Jurik encryption scheme guarantees that the descriptions of injective and lossy functions are computationally indistinguishable. Thus it only remains to give an upper bound for the size of the lossy functions' images.

Let (N,c) be a description of a lossy function, where $c = r^{N^s} \bmod N^{s+1}$ for some $r \in \mathbb{Z}_N^*$. We have $|D_\sigma| = N^s(N-p-q+1) \geq \frac{1}{2}N^{s+1} \geq 2^{(n-1)(s+1)-1}$. Using the isomorphism ψ_s described in Theorem 5.1 we can express the image of the function as follows:

$$\begin{aligned}
|\mathrm{Image}(f_{N,c})| &\leq \left|\left\{c^x \cdot y^{N^s} \bmod N^{s+1} : x \in \mathbb{Z}_{N^s}, y \in \mathbb{Z}_N^*\right\}\right| \\
&= \left|\left\{(r^x \cdot y)^{N^s} \bmod N^{s+1} : x \in \mathbb{Z}_{N^s}, y \in \mathbb{Z}_N^*\right\}\right| \\
&= \left|\left\{\psi_s(0, r^x \cdot y \bmod N) : x \in \mathbb{Z}_{N^s}, y \in \mathbb{Z}_N^*\right\}\right| \\
&< N \ .
\end{aligned}$$

Therefore the amount of lossiness is at least $\ell(n) = \log_2(|D_\sigma|/|\mathrm{Image}(N,c)|) \geq \log_2(\frac{1}{2}N^{s-1}N) \geq (n-1)s-1$. ☐

The above construction can easily be extended to a collection of all-but-one lossy trapdoor functions. We describe the extension here; the proof of security is essentially identical to the proof of Theorem 5.4 and is therefore omitted.

Given an integer $s \geq 1$ we define a 4-tuple $\mathcal{F}_s^{\mathrm{ABO}} = (\mathsf{B}, \mathsf{G}, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.2, and here we consider only one lossy branch as defined in [33]) as follows:

- **Sampling a branch:** On input 1^n the algorithm B outputs a uniformly distributed $b \in \{0, \ldots, 2^{n/2-1}\}$.

- **Sampling a function:** On input 1^n and a lossy branch b^* the algorithm G chooses an admissible n-bit modulus $N = PQ$. Then, it chooses a random $r \in \mathbb{Z}_N^*$ and lets $c = (1+N)^{-b^*} r^{N^s} \bmod N^{s+1}$. The description of the function is (N,c) and the trapdoor consists of $\lambda = \mathrm{lcm}(P-1, Q-1)$, b^*, and r.

- **Evaluation:** Given a description of a function (N,c), a branch b, and an input $(x,y) \in \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$, and outputs $\left((1+N)^b c\right)^x \cdot y^{N^s} \bmod N^{s+1}$.

- **Inversion:** Given a description (N,c) of a function, its trapdoor (λ, b^*, r), a branch $b \neq b^*$ and $z \in \mathbb{Z}_{N^{s+1}}$, the algorithm F^{-1} applies the inversion algorithm provided by Theorem 5.1 to compute $\psi_s^{-1}(z) = ((b-b^*)x, r^x \cdot y)$. Note that the restriction $b, b^* \in \{0, \ldots, 2^{n/2}-1\}$ implies that $b - b^*$ is relatively prime to N (since $2^{n/2-1} < \min\{P, Q\}$), and therefore the algorithm F^{-1} can recover x by computing $(b-b^*)x \cdot (b-b^*)^{-1} \bmod N^s$, and then recover y.

Theorem 5.3. *Under the decisional composite residuosity assumption, for any polynomial $s = s(n)$ it holds that $\mathcal{F}_s^{\mathrm{ABO}}$ is a collection of $((n-1)(s+1), (n-1)s-1)$-all-but-one lossy trapdoor functions with index-dependent domains.*

5.3 A lossy trapdoor function

We now extend the above construction to a lossy trapdoor function. In order to guarantee that all the functions in the collection share the same domain, we define the functions over the domain $\{0,1\}^{(n-1)s} \times \{0,1\}^{n/2-1}$. That is, the domain is $\{0,1\}^m$, for $m = m(n) = (n-1)s + n/2 - 1$. First, the fact that N is an n-bit modulus implies that any $x \in \{0,1\}^{(n-1)s}$ can be interpreted as an element of $\mathbb{Z}_{N^s}$ since $2^{(n-1)s} < N$. Second, the fact that P and Q are $n/2$-bit prime numbers implies that if $y \in \{0,1\}^{n/2-1}$ is interpreted as an integer between 1 and $2^{n/2-1}$, then $y \in \mathbb{Z}_N^*$ (since $2^{n/2-1} < \min\{P, Q\}$ and thus $\gcd(N, y) = 1$).

Given any polynomial $s = s(n)$ we define a 4-tuple $\mathcal{F}_s = (\mathsf{G}_0, \mathsf{G}_1, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.1) as follows:

1. **Sampling a lossy function:** On input 1^n the algorithm G_0 chooses an admissible n-bit modulus $N = PQ$. Then, it chooses a random $r \in \mathbb{Z}_N^*$ and lets $c = r^{N^s} \bmod N^{s+1}$. The description of the function is $\sigma = (N, c)$.

2. **Sampling an injective function:** On input 1^n the algorithm G_1 chooses an admissible n-bit modulus $N = PQ$. Then, it chooses a random $r \in \mathbb{Z}_N^*$ and lets $c = (1+N)r^{N^s} \bmod N^{s+1}$. The description of the function is $\sigma = (N, c)$ and the trapdoor is $\tau = (\lambda, r)$, where $\lambda = \mathrm{lcm}(P-1, Q-1)$.

3. **Evaluation:** Given a description of a function (N, c) and an input $(x, y) \in \{0,1\}^{(n-1)s} \times \{0,1\}^{n/2-1}$, the algorithm F interprets the input as an element of $\mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$ and outputs $c^x y^{N^s} \bmod N^{s+1}$.

4. **Inversion:** Given a description of an injective function (N, c) together with its trapdoor (λ, r) and $z \in \mathbb{Z}_{N^{s+1}}$, the algorithm F^{-1} invokes the inversion algorithm provided by Theorem 5.1 to compute $\psi_s^{-1}(z) = (x, r^x y)$, and then recovers x and y.

Theorem 5.4. *Under the composite residuosity assumption, for any polynomial $s = s(n)$ it holds that $\mathcal{F}_s$ is a collection of $((n-1)s + n/2 - 1, (n-1)s - n/2 - 1)$-lossy trapdoor functions.*

Proof. Similar to the proof of Theorem 5.2, we can express the image of the function as follows:

$$\begin{aligned}
|\mathrm{Image}(N, c)| &\leq \left|\left\{c^x \cdot y^{N^s} \bmod N^{s+1} : x \in \mathbb{Z}_{N^s}, y \in \mathbb{Z}_N^*\right\}\right| \\
&= \left|\left\{(r^x \cdot y)^{N^s} \bmod N^{s+1} : x \in \mathbb{Z}_{N^s}, y \in \mathbb{Z}_N^*\right\}\right| \\
&= \left|\{\psi_s(0, r^x \cdot y \bmod N) : x \in \mathbb{Z}_{N^s}, y \in \mathbb{Z}_N^*\}\right| \\
&< N \\
&< 2^n \ .
\end{aligned}$$

Therefore the amount of lossiness is at least $\ell(n) = ((n-1)s + n/2 - 1) - n = (n-1)s - n/2 - 1$. $\square$

The above construction can easily be extended to a collection of all-but-one lossy trapdoor functions. We describe the extension here; the proof of security is essentially identical to the proof of Theorem 5.4 and is therefore omitted.

Given an integer $s \geq 1$ we define a 4-tuple $\mathcal{F}_s^{\mathrm{ABO}} = (\mathsf{B}, \mathsf{G}, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.2, and here we consider only one lossy branch as defined in [33]) as follows:

- **Sampling a branch:** On input 1^n the algorithm B outputs a uniformly distributed $b \in \{0, \dots, 2^{n/2-1}\}$.
- **Sampling a function:** On input 1^n and a lossy branch b^* the algorithm G chooses an admissible n-bit modulus $N = PQ$. Then, it chooses a random $r \in \mathbb{Z}_N^*$ and lets $c = (1+N)^{-b^*} r^{N^s} \bmod N^{s+1}$. The description of the function is (N, c) and the trapdoor consists of $\lambda = \mathrm{lcm}(P-1, Q-1)$, b^*, and r.
- **Evaluation:** Given a description of a function (N, c), a branch b, and an input $(x, y) \in \{0,1\}^{(n-1)s} \times \{0,1\}^{n/2-1}$, the algorithm F interprets (x, y) as an element of $\mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$, and outputs $\left((1+N)^b c\right)^x \cdot y^{N^s} \bmod N^{s+1}$.
- **Inversion:** Given a description (N, c) of a function, its trapdoor (λ, b^*, r), a branch $b \neq b^*$ and $z \in \mathbb{Z}_{N^{s+1}}$, the algorithm F^{-1} applies the inversion algorithm provided by Theorem 5.1 to compute $\psi_s^{-1}(z) = ((b - b^*)x, r^x \cdot y)$. Note that the restriction $b, b^* \in \{0, \dots, 2^{n/2} - 1\}$ implies that $b - b^*$ is relatively prime to N (since $2^{n/2-1} < \min\{P, Q\}$), and therefore the algorithm F^{-1} can recover x by computing $(b - b^*)x \cdot (b - b^*)^{-1} \bmod N^s$, and then recover y.

Theorem 5.5. *Under the decisional composite residuosity assumption, for any polynomial $s = s(n)$ it holds that $\mathcal{F}_s^{\mathrm{ABO}}$ is a collection of $((n-1)s + n/2 - 1, (n-1)s - n/2 - 1)$-all-but-one lossy trapdoor functions.*

6 A Construction based on the d-Linear Assumption

The *d-Linear assumption* [21, 36] is a generalization of the decision Diffie-Hellman assumption that may hold even in groups with an efficiently computable d-linear map. The 1-Linear assumption is DDH, while the 2-Linear assumption is also known as the *Decision Linear* assumption [5]. The assumption is as follows:

Definition 6.1. Let $d \geq 1$ be an integer, and let $\mathbb{G}$ be a finite cyclic group of order q. We say the *d-Linear assumption* holds in $\mathbb{G}$ if the distributions

$$\begin{array}{ll} \left\{(g_1, \dots, g_d, g_1^{r_1}, \dots, g_d^{r_d}, h, h^{r_1 + \cdots + r_d}) \right. & : g_1, \dots, g_d, h \xleftarrow{\mathrm{R}} \mathbb{G},\ r_1, \dots, r_d \xleftarrow{\mathrm{R}} \mathbb{Z}_q \left.\right\}, \\ \left\{(g_1, \dots, g_d, g_1^{r_1}, \dots, g_d^{r_d}, h, h^{s}) \right. & : g_1, \dots, g_d, h \xleftarrow{\mathrm{R}} \mathbb{G},\ r_1, \dots, r_d, s \xleftarrow{\mathrm{R}} \mathbb{Z}_q \left.\right\} \end{array}$$

are computationally indistinguishable.

For any $d \geq 1$, the d-linear assumption implies the $(d+1)$-linear assumption [21, Lemma 3].

Peikert and Waters [33, Section 5] give lossy and all-but-one lossy trapdoor functions based on the DDH assumption. In the Peikert-Waters construction, the function index is an ElGamal encryption of an $n \times n$ matrix M which is either the zero matrix (lossy mode) or the identity matrix (injective mode) using a finite cyclic group $\mathbb{G}$ of order p. The DDH assumption in $\mathbb{G}$ implies that these two encryptions cannot be distinguished. The construction can be generalized to d-linear assumptions using generalized ElGamal encryption, but such schemes are less efficient since ElGamal based on the d-Linear assumption produces $d+1$ group elements per ciphertext (see e.g. [36]).

Our construction is based on the following basic observation from linear algebra: if M is an $n \times n$ matrix over a finite field $\mathbb{F}_p$ and $\vec{x}$ is a length-n column vector, then the map $f_M : \vec{x} \mapsto M\vec{x}$ has image of size $p^{\mathrm{Rk}(M)}$. If we restrict the domain to only *binary* vectors (i.e., those with entries in $\{0,1\}$), then the function f_M is injective when $\mathrm{Rk}(M) = n$, and its inverse can be computed by $f_M^{-1} : \vec{y} \mapsto M^{-1}\vec{y}$. If on the other hand we have $\mathrm{Rk}(M) < n/\log_2(p)$, then f_M is not injective even when the domain is restricted to binary vectors, since the image is contained in a subgroup of size less than 2^n.

By performing the above linear algebra "in the exponent" of a group of order p, we can create lossy trapdoor functions based on DDH and the related d-Linear assumptions. In particular, for any n the size of the function index is the same for all d.

We will use the following notation: we let $\mathbb{F}_p$ denote a field of p elements and $\mathrm{Rk}_d(\mathbb{F}_p^{n\times n})$ the set of $n \times n$ matrices over $\mathbb{F}_p$ of rank d. If we have a group $\mathbb{G}$ of order p, an element $g \in \mathbb{G}$, and a vector $\vec{x} = (x_1, \dots, x_n) \in \mathbb{F}_p^n$, then we define $g^{\vec{x}}$ to be the column vector $(g^{x_1}, \dots, g^{x_n}) \in \mathbb{G}^n$. If $M = (a_{ij})$ is an $n \times n$ matrix over $\mathbb{F}_p$, we denote by g^M the $n \times n$ matrix over $\mathbb{G}$ given by $(g^{a_{ij}})$. Given a matrix $M = (a_{ij}) \in \mathbb{F}_p^{n\times n}$ and a column vector $\mathbf{g} = (g_1, \dots, g_n) \in \mathbb{G}^n$, we define $\mathbf{g}^M$ by

$$\mathbf{g}^M = \left(\prod_{j=1}^n g_j^{a_{1j}}, \dots, \prod_{j=1}^n g_j^{a_{nj}}\right).$$

Similarly, given a matrix $\mathbf{S} = (g_{ij}) \in \mathbb{G}^{n\times n}$ and a column vector $\vec{x} = (x_1, \dots, x_n) \in \mathbb{F}_p^n$, we define $\mathbf{S}^{\vec{x}}$ by

$$\mathbf{S}^{\vec{x}} = \left(\prod_{j=1}^n g_{1j}^{x_j}, \dots, \prod_{j=1}^n g_{nj}^{x_j}\right).$$

With these definitions, we have $(g^M)^{\vec{x}} = (g^{\vec{x}})^M = g^{(M\vec{x})}$.

The construction.

For any positive integer d and any real number $\epsilon \in (0,1)$, we define a 4-tuple $\mathcal{F} = (\mathsf{G}_0, \mathsf{G}_1, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.1) as follows:

1. **Sampling a lossy function:** On input 1^n, the algorithm G_0 chooses at random a $\lceil \epsilon n/d \rceil$-bit prime p, a group $\mathbb{G}$ of order p, and a generator g of $\mathbb{G}$. Then it chooses a matrix $M \xleftarrow{\mathrm{R}} \mathrm{Rk}_d(\mathbb{F}_p^{n\times n})$ and computes $\mathbf{S} = g^M \in \mathbb{G}^{n\times n}$. The function index is $\sigma = \mathbf{S}$.

2. **Sampling an injective function:** On input 1^n, the algorithm G_1 chooses at random a $\lceil \epsilon n/d \rceil$-bit prime p, a group $\mathbb{G}$ of order p, and a generator g of $\mathbb{G}$. Then it chooses a matrix $M \xleftarrow{\mathrm{R}} \mathrm{Rk}_n(\mathbb{F}_p^{n\times n})$ and computes $\mathbf{S} = g^M \in \mathbb{G}^{n\times n}$. The function index is $\sigma = \mathbf{S}$, and the trapdoor is $\tau = (g, M)$.

3. **Evaluation:** Given a function index $\mathbf{S}$ and $\vec{x} = (x_1, \dots, x_n) \in \{0,1\}^n$, the algorithm F computes the function $f_{\mathbf{S}}(x) = \mathbf{S}^{\vec{x}}$.

4. **Inversion:** Given a function index $\mathbf{S}$, a trapdoor $\tau = (g, M)$, and a vector $\mathbf{g} \in \mathbb{G}^n$, we define $\mathsf{F}^{-1}(\tau, \mathbf{g})$ as follows:

   (a) Compute $\mathbf{h} = (h_1, \dots, h_n) \leftarrow \mathbf{g}^{M^{-1}}$.

(b) Let $x_i = \log_g(h_i)$ for $i = 1, \dots, n$.

(c) Output $\vec{x} = (x_1, \dots, x_n)$.

Theorem 6.2. *Suppose $\epsilon n > d$. If the d-Linear assumption holds for $\mathbb{G}$, then the above family is a collection of $(n, (1-\epsilon)n)$-lossy trapdoor functions.*

Proof. We first note that in the lossy case, when M is of rank d, the image of $f_{\mathbf{S}}$ is contained in a subgroup of $\mathbb{G}^n$ of size $p^d < 2^{\epsilon n}$. The condition $\epsilon n > d$ guarantees $p \geq 3$, so when M is of rank n the function $f_{\mathbf{S}}$ is in fact injective. It is straightforward to verify that the inversion algorithm performs correctly for injective functions. Finally, by [27, Lemma A.1], the d-Linear assumption implies that the matrix $\mathbf{S}$ when M is of rank n is computationally indistinguishable from the matrix $\mathbf{S}$ when M is of rank d. □

Note that the system's security scales with the bit size of p, i.e., as $\epsilon n/d$. In addition, note that the discrete logarithms in the inversion step can be performed efficiently when $\vec{x}$ is a binary vector. (Here we take advantage of the fact that the output of F^{-1} is unspecified on inputs not in the image of F.)

We now describe the extension of the system to all-but-one lossy trapdoor functions, in the case where the parameter d in the above construction is equal to 1. Let I_n denote the $n \times n$ identity matrix. For any real number $\epsilon \in (0, 1)$, we define a 4-tuple $\mathcal{F} = (\mathsf{G}_0, \mathsf{G}_1, \mathsf{F}, \mathsf{F}^{-1})$ (recall Definition 2.2) as follows:

1. **Sampling a branch:** On input 1^n, the algorithm B outputs a uniformly distributed $b \in \{1, \dots, 2^{\lfloor \epsilon n \rfloor}\}$.

2. **Sampling a function:** On input 1^n and a lossy branch b^*, the algorithm G chooses at random a $\lceil \epsilon n \rceil$-bit prime p, a group $\mathbb{G}$ of order p, and a generator g of $\mathbb{G}$. Then it chooses a matrix $A \overset{\text{R}}{\leftarrow} \mathrm{Rk}_1(\mathbb{F}_p^{n \times n})$. Let $M = A - b^* I_n \in \mathbb{F}_p^{n \times n}$ and $\mathbf{S} = g^M \in \mathbb{G}^{n \times n}$. The function index is $\sigma = \mathbf{S}$, the trapdoor is $\tau = (g, M)$, and the set of lossy branches is $\beta = \{b^*, b^* - \mathrm{Tr}(A)\}$.

3. **Evaluation:** Given a function index $\mathbf{S}$, a branch b, and an input $x \in \{0, 1\}^n$, we interpret x as a binary column vector $\vec{x} = (x_1, \dots, x_n)$. The algorithm F computes the function $f_{\mathbf{S},b}(\vec{x}) = \mathbf{S}^{\vec{x}} * g^{b\vec{x}}$, where $*$ indicates the componentwise product of elements of $\mathbb{G}^n$.

4. **Inversion:** Given a function index $\mathbf{S}$, a trapdoor $\tau = (g, M)$, a branch b, and a vector $\mathbf{g} \in \mathbb{G}^n$, we define $\mathsf{F}^{-1}(\tau, b, \mathbf{g})$ as follows:

   (a) If $M + bI_n$ is not invertible, output $\perp$.

   (b) Compute $\mathbf{h} = (h_1, \dots, h_n) \leftarrow \mathbf{g}^{(M+bI_n)^{-1}}$.

   (c) Let $x_i = \log_g(h_i)$ for $i = 1, \dots, n$.

   (d) Output $\vec{x} = (x_1, \dots, x_n)$.

Theorem 6.3. *Suppose $\epsilon n > 1$. If the DDH assumption holds for $\mathbb{G}$, then the above family is a collection of $(n, (1-\epsilon)n)$-all-but-one lossy trapdoor functions.*

Proof. We first observe that if A is the rank 1 matrix computed by $\mathsf{G}(1^n, b^*)$, then

$$f_{\mathbf{S},b}(\vec{x}) = g^{(A-(b^*-b)I_n)\vec{x}}. \tag{6.1}$$

We now verify each property of Definition 2.2. Properties (1) and (2) are immediate. To verify property (3), note that (6.1) implies that $f_{\mathbf{S},b^*}(\vec{x}) = g^{A\vec{x}}$. Since A has rank 1, the image of $f_{\mathbf{S},b^*}$ is contained in a subgroup of $\mathbb{G}^n$ of size $p < 2^{\epsilon n}$.

To check property (4), we observe that the condition $\epsilon n > 1$ guarantees $p \geq 3$, so when $A - (b^* - b)I_n$ is invertible the function $f_{\mathbf{S},b}$ is injective. The condition $A - (b^* - b)I_n$ being not invertible is equivalent to $(b^* - b)$ being an eigenvalue of A. Since A has rank 1, its eigenvalues are 0 and $\mathrm{Tr}(A)$. Thus $(b^* - b)$ is an eigenvalue of A if and only if $b \in \beta$, and $f_{\mathbf{S},b}$ is injective for all $b \notin \beta$. It is straightforward to verify that the inversion algorithm performs correctly whenever $b \notin \beta$, so property (5) holds.

Properties (6) and (7) follow from the DDH assumption for $\mathbb{G}$. We show property (6) by constructing a sequence of games:

Game_0**:** This is the real security game. The adversary is given b_0, b_1, and $g^{A - b_\omega I_n}$ for $\omega \xleftarrow{\mathrm{R}} \{0,1\}$ and $A \xleftarrow{\mathrm{R}} \mathrm{Rk}_1(\mathbb{F}_p^{n\times n})$, and outputs a bit ω'. The adversary wins if $\omega' = \omega$.

Game_1**:** The same as Game_0, except the challenge is $g^{A' - b_\omega I_n}$ for some full rank matrix $A' \xleftarrow{\mathrm{R}} \mathrm{Rk}_n(\mathbb{F}_p^{n\times n})$.

Game_2**:** The same as Game_1, except the challenge is $g^{U - b_\omega I_n}$ for some uniform matrix $U \xleftarrow{\mathrm{R}} \mathbb{F}_p^{n\times n}$.

Game_3**:** The same as Game_2, except the challenge is g^U.

Since the Game_3 challenge is independent of ω, the advantage of any adversary playing Game_3 is zero. We now show that if the DDH assumption holds for $\mathbb{G}$, then for $i = 0, 1, 2$, no polynomial-time adversary $\mathcal{A}$ can distinguish Game_i from Game_{i+1} with non-negligible advantage.

$i = 0$**:** Any algorithm that distinguishes Game_0 from Game_1 can be used to distinguish the distributions $\{g^A : A \xleftarrow{\mathrm{R}} \mathrm{Rk}_1(\mathbb{F}_p^{n\times n})\}$ and $\{g^{A'} : A' \xleftarrow{\mathrm{R}} \mathrm{Rk}_n(\mathbb{F}_p^{n\times n})\}$. By [6, Lemma 1], any algorithm that distinguishes these distributions can solve the DDH problem in $\mathbb{G}$.

$i = 1$**:** Since the proportion of full-rank matrices to all matrices in $\mathbb{F}_p^{n\times n}$ is $(p-1)/p$, even an unbounded adversary can distinguish Game_1 from Game_2 with probability at most $1/p$.

$i = 2$**:** Since the matrix U is uniform in $\mathbb{F}_p^{n\times n}$, the matrix $U - b_\omega I_n$ is also uniform in $\mathbb{F}_p^{n\times n}$, so Game_2 and Game_3 are identical.

We conclude that for any b_0, b_1, no polynomial-time adversary can win Game_0 with non-negligible advantage.

Finally, to demonstrate property (7) we show that any adversary $\mathcal{A}$ that produces an element of β given $\mathbf{S}$ and b^* can be used to compute discrete logarithms in $\mathbb{G}$, contradicting the DDH assumption. Choose a matrix $A \xleftarrow{\mathrm{R}} \mathrm{Rk}_1(\mathbb{F}_p^{n\times n})$, and let $A'(X)$ be the $n \times n$ matrix over $\mathbb{F}_p[X]$ that is the matrix A with the first row multiplied by X. For any value $X = t \neq 0$, the matrix $A'(t)$ is uniformly distributed in $\mathrm{Rk}_1(\mathbb{F}_p^{n\times n})$.

Let (g, g^t) be a discrete logarithm challenge for $\mathbb{G}$. For any b^* we compute the matrix $\mathbf{S} = g^{A'(t) - b^* I_n}$ and give $(\mathbf{S}, b^*)$ to the adversary $\mathcal{A}$. If the adversary outputs $b \in \beta$ with $b \neq b^*$, then we can compute $\mathrm{Tr}(A'(t))$ since this is the only nonzero eigenvalue of $A'(t)$. If a_{ii} is the ith diagonal entry of A, this gives us an equation

$$a_{11}t + a_{22} + \cdots + a_{nn} = \lambda. \tag{6.2}$$

Since $a_{11} = 0$ with probability $1/p$, we can solve for t with all but negligible probability. $\square$

If we choose any integer $d \geq 1$ and repeat the above construction with p a $\lceil \epsilon n/d \rceil$-bit prime and A a rank d matrix, then we expect to obtain an all-but-one lossy trapdoor function under the d-Linear assumption. Indeed, the proofs of properties (1)–(6) carry through in a straightforward way. However, the above proof of property (7) does not seem to generalize. In particular, the generalization of (6.2) is the equation $\det(A'(t) - \lambda I_n) = 0$, which can be written as $ut + v = 0$ for some (known) $u, v \in \mathbb{F}_p$. When $d = 1$ the element $u = a_{11}$ is independent of λ, so we can conclude that it is nonzero with high probability; however when $d \geq 2$ this is not necessarily the case. We thus leave as an open problem the completion of the proof for $d \geq 2$.

7 Correlated Input Security from Syndrome Decoding

Our construction is based on Niederreiter's coding-based encryption system [29] which itself is the dual of the McEliece encryption system [25].

Let $0 < \rho = \rho(n) < 1$ and $0 < \delta = \delta(n) < 1/2$ be two functions in the security parameter n. We set the domain $D_{n,\delta}$ to be the set of all n-bit strings with Hamming weight δn. Note that D_n is efficiently samplable (see e.g. [17]). The Niederreiter trapdoor function $\mathcal{F} = (\mathsf{G}, \mathsf{F}, \mathsf{F}^{-1})$ is defined as follows.

- **Key generation:** On input 1^n the algorithm G chooses at random a non-singular binary $\rho n \times \rho n$ matrix S, a $(n, n - \rho n, \delta n)$-linear binary Goppa code capable of correcting up to δn errors (given by its $\rho n \times n$ binary parity check matrix G), and a $n \times n$ permutation matrix P. It sets $H := SGP$, which is a binary $\rho n \times n$ matrix. The description of the function is $\sigma = H$, the trapdoor is $\tau = (S, G, P)$.

- **Evaluation:** Given a description H of a function and $x \in \{0,1\}^n$ with Hamming weight δn, the algorithm F computes the function $f_H(x) = Hx \in \{0,1\}^{\rho n}$.

- **Inversion:** Given the trapdoor (S, G, P) and $y = Hx$, the algorithm F^{-1} computes $S^{-1}y = GPx$, applies a syndrome decoding algorithm for G to recover $\hat{y} = Px$, and computes $x = P^{-1}\hat{y}$.

The Niederreiter trapdoor function can be proved one-way under the indistinguishability and syndrome decoding assumptions which are indexed by the parameters $0 < \rho < 1$ and $0 < \delta < 1/2$.

Indistinguishability assumption. The binary $\rho n \times n$ matrix H output by $\mathsf{G}(1^n)$ is computationally indistinguishable from a uniform matrix of the same dimensions.

Syndrome decoding assumption. The collection of functions which is defined as $f_U(x) := Ux$ for a uniform $\rho n \times n$ binary matrix U is one-way on domain $D_{n,\delta}$.

Choosing the weight δ to be close to the Gilbert-Warshamov bound is commonly believed to give hard instances for the syndrome decoding problem. The Gilbert-Warshamov bound for a $(n, k, \delta n)$ linear code with $\delta < 1/2$ is given by the equation $k/n \leq 1 - H_2(\delta)$, where $H_2(\delta) := -\delta \log_2 \delta - (1 - \delta)\log_2(1 - \delta)$. It is therefore assumed that the syndrome decoding assumption holds for all $0 < \delta < 1/2$ satisfying $H_2(\delta) < \rho$ [17]. Note that one-wayness also implies that the cardinality of $D_{n,\delta}$ is super-polynomial in n.

The following theorem was proved in [17].

Theorem 7.1 ([17])**.** *If the syndrome decoding assumption holds for $\tilde{\rho}$ and δ then the ensembles* $\{(M, Mx) : M \xleftarrow{\text{R}} \{0,1\}^{\tilde{\rho}n \times n};\ x \xleftarrow{\text{R}} D_{n,\delta})\}_{n \in \mathbb{N}}$ *and* $\{(M, y) : M \xleftarrow{\text{R}} \{0,1\}^{\tilde{\rho}n \times n};\ y \xleftarrow{\text{R}} \{0,1\}^{\tilde{\rho}n}\}_{n \in \mathbb{N}}$ *are computationally indistinguishable.*

This theorem implies that the Niederreiter trapdoor function is one-way under k-correlated inputs.

Theorem 7.2. *Suppose ρ, δ, and k are chosen such that $\tilde{\rho} := \rho k < 1$, and the indistinguishability and the syndrome decoding assumptions hold for parameters $\tilde{\rho}$ and δ. Then the Niederreiter trapdoor function is one-way under k-correlated inputs.*

Proof. Fix a probabilistic polynomial-time adversary $\mathcal{A}$ that plays the security game for one-wayness under k-correlated inputs. Define

$$\varepsilon = \Pr[\mathcal{A}(H_1, \ldots, H_k, H_1(x), \ldots, H_k(x)) = x],$$

where $H_i \xleftarrow{\text{R}} \mathsf{G}(1^n)$ and $x \xleftarrow{\text{R}} D_{n,\delta}$. We now exchange all the matrices H_i for uniform matrices U_i of the same dimension. By the indistinguishability assumption and a hybrid argument, we have that

$$\Big| \Pr[\mathcal{A}(H_1, \ldots, H_k, H_1(x), \ldots, H_k(x)) = x] \\ - \Pr[\mathcal{A}(U_1, \ldots, U_k, U_1(x), \ldots, U_k(x)) = x] \Big| \in \text{negl}(n).$$

For $\tilde{\rho} := \rho k$, define the $\tilde{\rho}n \times n$ matrix U by concatenating the columns of the matrices U_i. Then the distributions $(U_1, \ldots, U_k, U_1(x), \ldots, U_k(x))$ and (U, Ux) are identical. Since $H_2(\delta) \leq \rho/k = \tilde{\rho}$ we can apply Theorem 7.1 to obtain

$$|\Pr[\mathcal{A}(U, Ux) = x] - \Pr[\mathcal{A}(M, u_{\tilde{\rho}n}) = x]| \in \text{negl}(n),$$

where $u_{\tilde{\rho}n}$ is a uniform bit-string in $\{0,1\}^{\tilde{\rho}n}$. Observing that $\Pr[\mathcal{A}(U, u_{\tilde{\rho}n}) = x] = 1/|D_{n,\delta}| \in \text{negl}(n)$ (since the Niederreiter function is assumed to be one-way) implies that ε is negligible. $\square$

We remark that the above proof implies that the Niederreiter trapdoor function has linearly many hard-core bits, which greatly improves efficiency of the CCA-secure encryption scheme obtained by using the construction from [35].

Acknowledgements

We thank Ivan Damgård and Chris Peikert for useful discussions. We thank Dan Boneh for showing us references [7] and [37].

Part of David Mandell Freeman's research was conducted at CWI and Universiteit Leiden, Netherlands, and supported by a National Science Foundation International Research Fellowship, with additional support from the Office of Multidisciplinary Activities in the NSF Directorate for Mathematical and Physical Sciences. The remainder was supported by an NSF Mathematical Sciences Postdoctoral Fellowship.

Oded Goldreich is partially supported by the Israel Science Foundation (grant No. 1041/08).

Eike Kiltz is supported by the research program Sentinels. Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

Alon Rosen is partially supported by the Israel Science Foundation (grant No. 334/08).

Gil Segev is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

References

[1] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *Advances in Cryptology — ASIACRYPT 2009*, volume 5912 of *Springer LNCS*, pages 232–249, 2009.

[2] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of *Springer LNCS*, pages 1–35, 2009.

[3] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112, 1988.

[4] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Advances in Cryptology — CRYPTO 2008*, volume 5157 of *Springer LNCS*, pages 335–359, 2008.

[5] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Springer LNCS*, pages 41–55, 2004.

[6] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *Advances in Cryptology — CRYPTO 2008*, volume 5157 of *Springer LNCS*, pages 108–125, 2008.

[7] D. Boneh and J. Horwitz. Weak trapdoors from the rth-power-residue symbol. Unpublished manuscript, 2002.

[8] D. Boneh, K. Rubin, and A. Silverberg. Finding composite order ordinary elliptic curves using the cocks-pinch method. Cryptology ePrint Archive, Report 2009/533, 2009. `http://eprint.iacr.org/`.

[9] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology — EUROCRYPT 1999*, volume 1592 of *Springer LNCS*, pages 402–414, 1999.

[10] H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[11] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.

[12] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology – EUROCRYPT 2002*, pages 45–64, 2002.

[13] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography — PKC 2001*, volume 1992 of *Springer LNCS*, pages 119–136, 2001. Full version (with additional co-author J. B. Nielsen) available at `htttp://www.daimi.au.dk/~ivan/GenPaillier_finaljour.ps`.

[14] I. Damgård and J. B. Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Springer LNCS*, pages 581–596, 2002.

[15] I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Springer LNCS*, pages 247–264, 2003.

[16] R. Dowsley, J. Müller-Quade, and A. C. A. Nascimento. A CCA2 secure public key encryption scheme based on the McEliece assumptions in the standard model. In *Topics in Cryptology — CT-RSA 2009*, volume 5473 of *Springer LNCS*, pages 240–251, 2009.

[17] J.-B. Fischer and J. Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *Advances in Cryptology — EUROCRYPT 1996*, volume 1070 of *Springer LNCS*, pages 245–255, 1996.

[18] O. Goldreich. Foundations of cryptography – basic applications. Cambridge University Press, 2004.

[19] S. Goldwasser and V. Vaikuntanathan. New constructions of correlation-secure trapdoor functions and CCA-secure encryption schemes. Manuscript, 2008.

[20] B. Hemenway and R. Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. Electronic Colloquium on Computational Complexity, Report TR09-127, 2009.

[21] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *Advances in Cryptology — CRYPTO 2007*, volume 4622 of *Springer LNCS*, pages 553–571, 2007.

[22] J. Horwitz. *Applications of Cayley graphs, bilinearity, and higher-order residues to cryptology.* PhD thesis, Stanford University, 2004. Available at `http://math.scu.edu/~jhorwitz/pubs/horwitz-phd.pdf`.

[23] K. Ireland and M. Rosen. *A classical introduction to modern number theory*, volume 84 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1990.

[24] E. Kiltz, A. O'Neill, and A. Smith. Lossiness of RSA and the instantiability of OAEP. Manuscript, 2009.

[25] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN Prog. Rep., Jet Prop. Lab.*, pages 114–116, Jan 1978.

[26] P. Mol and S. Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In *Public Key Cryptography — PKC 2010*, volume ???? of *Springer LNCS*, pages ???–???, 2010. Full version available at `http://eprint.iacr.org/2009/524`.

[27] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology — CRYPTO 2009*, volume 5677 of *Springer LNCS*, pages 18–35, 2009. Full version available at `http://eprint.iacr.org/2009/105`.

[28] J. Neukirch. *Algebraic Number Theory*, volume 322 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1999. Translated from the German by N. Schappacher.

[29] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory. [Problemy Upravlenija i Teorii Informacii]*, 15:159–166, 1986.

[30] R. Nishimaki, E. Fujisaki, and K. Tanaka. Efficient non-interactive universally composable string-commitment schemes. In *ProvSec*, pages 3–18, 2009.

[31] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT 1999*, volume 1592 of *Springer LNCS*, pages 223–238, 1999.

[32] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *41st ACM Symposium on Theory of Computing*, pages 333–342, 2009.

[33] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *40th ACM Symposium on Theory of Computing*, pages 187–196, 2008. Full version available at `http://eprint.iacr.org/2007/279`.

[34] M. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[35] A. Rosen and G. Segev. Chosen-ciphertext security via correlated products. In *Theory of Cryptography Conference — TCC 2009*, volume 5444 of *Springer LNCS*, pages 419–436, 2009.

[36] H. Shacham. A Cramer-Shoup encryption scheme from the Linear assumption and from progressively weaker Linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. Available at `http://eprint.iacr.org/2007/074`.

[37] D. Squirrel. Computing reciprocity symbols in number fields, 1997. Undergraduate thesis, Reed College.

A A Note on the Relationship between the 2vs3Primes and QR Assumptions

Folklore dating to the 1980s (see, e.g., [3]) asserts that if it is hard to distinguish 2-prime composites from 3-prime composites, then it is hard to distinguish quadratic residues from quadratic non-residues (modulo a composite). As is often the case with folklore, it is not clear what *exactly* is meant by this assertion. Specifically, the folklore observation is that a quadratic residuosity oracle allows one to distinguish a 2-prime composites N from 3-prime composites N (by sampling random integers modulo N and counting the fraction of quadratic residues modulo N), but this observation presumes that the quadratic residuosity oracle works well for both 2-prime and 3-prime composites.[4] Note, however, that the existence of such an oracle does *not* follow from the negation of the *standard* quadratic residuosity assumption, which only refers to 2-prime composites. Indeed, the (standard) quadratic residuosity assumption asserts that for random 2-prime composites $N = PQ$ such that $|\log_2 P - \log_2 Q| \leq 1$ (and $P \equiv Q \equiv 3 \pmod 4$) it is infeasible to distinguish random quadratic residues modulo N from random quadratic nonresidues modulo N that have Jacobi symbol 1, where in both cases the potential distinguisher is also given the composite N. The negation of this assumption is that a corresponding distinguisher does exist, but it is unclear how such a distinguisher behaves when given pairs (x, N) when N is a 3-prime composite.

Let $\mathsf{Gen}_2(\cdot)$ be a probabilistic polynomial-time algorithm that on input 1^n samples two $n/2$-bit prime numbers P and Q such that $N = PQ$ is an n-bit number, and outputs N. Similarly, let $\mathsf{Gen}_3(\cdot)$ be a probabilistic polynomial-time algorithm that on input 1^n samples three $n/3$-bit prime numbers P, Q, and R such that $N = PQR$ is an n-bit number, and outputs N. In both cases the primes may be subject to additional constraints, in the form of congruence relations. For example, for most applications the algorithm $\mathsf{Gen}_2(\cdot)$ is required to choose P and Q such that $P \equiv Q \equiv 3 \bmod 4$ (i.e., Blum integers).

For any odd positive integer N we denote by $\mathsf{JS}_N : \mathbb{Z} \rightarrow \{-1, 0, 1\}$ the Jacobi symbol modulo N, and define $\mathcal{J}_N = \{x \in \mathbb{Z}_N^* : \mathsf{JS}_N(x) = 1\}$, and $\mathcal{Q}_N = \{x^2 : x \in \mathbb{Z}_N^*\}$. Using this notation, the QR assumption asserts that, for N generated by $\mathsf{Gen}_2(\cdot)$, the uniform distribution over $\mathcal{Q}_N$ is computationally indistinguishable from the uniform distribution over $\mathcal{J}_N \setminus \mathcal{Q}_N$. (In both cases, the sample of $\mathbb{Z}_N$ is accompanied by N itself.) As for the 2vs3Primes assumption, it asserts that the output distributions of $\mathsf{Gen}_2(\cdot)$ and $\mathsf{Gen}_3(\cdot)$ are computationally indistinguishable.

In what follows we prove that under a reasonable restriction on the primes that are sampled by $\mathsf{Gen}_2(\cdot)$ and $\mathsf{Gen}_3(\cdot)$, the 2vs3Primes assumption implies the QR assumption. The restriction that we require is that given an integer N that is the output of either $\mathsf{Gen}_2(\cdot)$ or $\mathsf{Gen}_3(\cdot)$, it is possible to efficiently sample from the uniform distribution over the set $\mathcal{J}_N \setminus \mathcal{Q}_N$. That is, we need to be able to efficiently produce a uniformly distributed element that has Jacobi symbol 1 (modulo N), and is not a square modulo N. We denote this restriction by (R1), and below we demonstrate that it it is implied by enforcing simple congruence relations on the primes that compose N.

Theorem A.1. *Subject to restriction* (R1)*, the 2vs3Primes assumption implies the QR assumption.*

Proof. We show how to reduce distinguishing between 2-prime and 3-prime composites to predicting the quadratic residuosity character of residues modulo 2-prime composites. Let $\mathcal{A}$ be an arbitrary algorithm, and let $\epsilon(n)$ denote the advantage of $\mathcal{A}$ in guessing the quadratic character of a random $x \in \mathcal{J}_N$, where $N \leftarrow \mathsf{Gen}_2(1^n)$. Denoting the quadratic character of x modulo N by $\mathsf{QC}_N(x)$ (i.e., $\mathsf{QC}_N(x) = 1$ if and only if x is a square mod N), we have

$$\begin{aligned}
\epsilon(n) &\overset{\text{def}}{=} 2 \cdot \left(\Pr\left[\mathcal{A}(x, N) = \mathsf{QC}_N(x) : N \leftarrow \mathsf{Gen}_2(1^n), x \leftarrow \mathcal{J}_N\right] - \frac{1}{2} \right) \\
&= \Pr\left[\mathcal{A}(x, N) = 1 : N \leftarrow \mathsf{Gen}_2(1^n), x \leftarrow \mathcal{Q}_N\right] \\
&\qquad - \Pr\left[\mathcal{A}(x, N) = 1 : N \leftarrow \mathsf{Gen}_2(1^n), x \leftarrow \mathcal{J}_N \setminus \mathcal{Q}_N\right]
\end{aligned}$$

[4] Furthermore, the straightforward argument seems to presume that, in the case of 3-prime composites, the oracle is correct with probability greater than 3/4.

Assuming that $\mathcal{A}$ is efficient and that $\epsilon(n)$ is non-negligible, we derive a distinguisher $\mathcal{A}'$ between 2-prime and 3-prime composites. For any $n \in \mathbb{N}$ and $i \in \{2, 3\}$ define

$$\begin{aligned}
\alpha_i(n) &= \Pr\left[\mathcal{A}(x, N) = 1 : N \leftarrow \mathsf{Gen}_i(1^n), x \leftarrow \mathcal{J}_N\right] , \\
\beta_i(n) &= \Pr\left[\mathcal{A}(x, N) = 1 : N \leftarrow \mathsf{Gen}_i(1^n), x \leftarrow \mathcal{Q}_N\right] , \\
\gamma_i(n) &= \Pr\left[\mathcal{A}(x, N) = 1 : N \leftarrow \mathsf{Gen}_i(1^n), x \leftarrow \mathcal{J}_N \setminus \mathcal{Q}_N\right] .
\end{aligned}$$

Using the fact that for $N = PQ$ the quotient group $\mathcal{J}_N/\mathcal{Q}_N$ consists of two cosets whereas for $N = PQR$ it consists of four cosets, we infer that

$$\begin{aligned}
\alpha_2(n) &= \frac{1}{2} \cdot \beta_2(n) + \frac{1}{2} \cdot \gamma_2(n) , \\
\alpha_3(n) &= \frac{1}{4} \cdot \beta_3(n) + \frac{3}{4} \cdot \gamma_3(n) ,
\end{aligned}$$

and therefore

$$\alpha_2(n) - \alpha_3(n) = \frac{1}{4} \cdot (\beta_2(n) - \beta_3(n)) + \frac{3}{4} \cdot (\gamma_2(n) - \gamma_3(n)) + \frac{1}{4} \cdot (\beta_2(n) - \gamma_2(n)) . \tag{A.1}$$

The term $\beta_2(n) - \gamma_2(n)$ is equal to $\epsilon(n)$, which is non-negligible by our hypothesis. Thus at least one of the three other terms in equation (A.1) (i.e., $\alpha_2(n) - \alpha_3(n)$, $\beta_2(n) - \beta_3(n)$, and $\gamma_2(n) - \gamma_3(n)$) must also be non-negligible. In all three of these cases we can construct an algorithm $\mathcal{A}'$ that has a non-negligible advantage in distinguishing between the output distributions of $\mathsf{Gen}_2(\cdot)$ and $\mathsf{Gen}_3(\cdot)$:

- If $\alpha_2(n) - \alpha_3(n)$ is non-negligible, then let $\mathcal{A}'$ be the algorithm that on input N samples $x \leftarrow \mathcal{J}_N$, and invokes $\mathcal{A}(x, N)$. Such an x can be sampled, for example, by sampling a uniform $x \in \mathbb{Z}_N^*$, computing its Jacobi symbol, and repeating the process until we find an element with Jacobi symbol 1.

- If $\beta_2(n) - \beta_3(n)$ is non-negligible, then let $\mathcal{A}'$ be the algorithm that on input N samples $x \leftarrow \mathcal{Q}_N$, and invokes $\mathcal{A}(x, N)$. Such an x can be sampled by sampling a uniform $x \in \mathbb{Z}_N^*$ and outputting $x^2 \bmod N$.

- If $\gamma_2(n) - \gamma_3(n)$ is non-negligible, then let $\mathcal{A}'$ be the algorithm that on input N samples $x \leftarrow \mathcal{J}_N \setminus \mathcal{Q}_N$, and invokes $\mathcal{A}(x, N)$. Such an x can be sampled due to restriction (R1).

□

We conclude by presenting another restriction (R2) which is easy to satisfy and implies restriction (R1). As pointed out in the proof of Claim A.1, the quotient group $\mathcal{J}_N/\mathcal{Q}_N$ consists of two cosets for $N = PQ$ and of four cosets for $N = PQR$. More specifically, for $N = PQ$ it holds that $\mathcal{J}_N/\mathcal{Q}_N = \{\mathcal{Q}_N, \mathcal{Q}_N \cdot e\}$ for any $e \in \mathcal{J}_N \setminus \mathcal{Q}_N$. In addition, for $N = PQR$ it holds that $\mathcal{J}_N/\mathcal{Q}_N = \{\mathcal{Q}_N, \mathcal{Q}_N \cdot e_1, \mathcal{Q}_N \cdot e_2, \mathcal{Q}_N \cdot e_3\}$ for any $e_1, e_2, e_3 \in \mathcal{J}_N$ such that e_1 is a square modulo P and not modulo Q or R, e_2 is a square modulo Q and not modulo P or R, and e_3 is a square modulo R and not modulo P or Q.

In restriction (R2) we require that there exist three fixed elements $e_1, e_2, e_3 \in \mathbb{Z}_N$ such that for $N = PQ$ it holds that $e_1, e_2, e_3 \in \mathcal{J}_N \setminus \mathcal{Q}_N$, and for $N = PQR$ it holds that $\mathcal{J}_N/\mathcal{Q}_N = \{\mathcal{Q}_N, \mathcal{Q}_N \cdot e_1, \mathcal{Q}_N \cdot e_2, \mathcal{Q}_N \cdot e_3\}$. Given this restriction, the algorithm that samples x uniformly in

Z_N^* and e uniformly in $\{e_1, e_2, e_3\}$, and outputs $x^2 \cdot e \bmod N$, produces, in both cases, a uniformly distributed element in the set $\mathcal{J}_N \setminus \mathcal{Q}_N$.

Using the fact that for any odd prime $P \neq 3$, it holds that

$$\mathsf{JS}_{\mathsf{P}}(-1) = \begin{cases} 1, & \text{if } P \equiv 1 \bmod 4 \\ -1, & \text{if } P \equiv 3 \bmod 4 \end{cases}$$

$$\mathsf{JS}_P(2) = \begin{cases} 1, & \text{if } P \equiv 1 \text{ or } 7 \bmod 8 \\ -1, & \text{if } P \equiv 3 \text{ or } 5 \bmod 8 \end{cases}$$

$$\mathsf{JS}_P(-3) = \begin{cases} 1, & \text{if } P \equiv 1 \bmod 3 \\ -1, & \text{if } P \equiv 2 \bmod 3, \end{cases}$$

we see that restriction (R2) is satisfied with $e_1 = -1$, $e_2 = 2$, and $e_3 = -3$, provided that $\mathsf{Gen}_2(\cdot)$ chooses P and Q such that $P \equiv Q \equiv 11 \bmod 24$, and that $\mathsf{Gen}_3(\cdot)$ chooses P, Q and R such that $P \equiv 5 \bmod 24$, $Q \equiv 23 \bmod 24$, and $R \equiv 19 \bmod 24$.

Note that these congruence restrictions may make the problem of distinguishing 2-prime composites from 3-primes composites easier than in the general case. In particular, the distinguishing task given the above choice of e_i would be extremely easy if it were the case that $11 \cdot 11 \not\equiv 5 \cdot 19 \cdot 23 \bmod 24$, but "luckily" this is not the case.[5]

For larger values of e_i there are in fact choices of congruences for P, Q in $\mathsf{Gen}_2(\cdot)$ and P, Q, R in $\mathsf{Gen}_3(\cdot)$ such that the e_i always have the desired Jacobi symbols, but $N \bmod e_1 e_2 e_3$ is different in the 2-primes case and the 3-primes case.

B Compatibility of Power Residue Symbols

Proposition B.1. *Let e, f be integers with $f \mid e$. Let $x \in \mathbb{Z}[\zeta_f]$, let $\mathfrak{A}$ be an ideal of $\mathbb{Z}[\zeta_e]$ relatively prime to e, and let $\mathfrak{a} = \mathfrak{A} \cap \mathbb{Z}[\zeta_f]$. Then*

$$\left(\frac{x}{\mathfrak{a}}\right)_f = \left(\frac{x}{\mathfrak{A}}\right)_e^{e/f}.$$

Proof. By multiplicativity of the power residue symbol, it suffices to prove the statement when $\mathfrak{A}$ is a prime $\mathfrak{P}$ of $\mathbb{Z}[\zeta_e]$. Let $\mathfrak{p} = \mathfrak{P} \cap \mathbb{Z}[\zeta_f]$. If we let α be an eth root of x and β be an fth root of x, then we have (see [28, §V.3] and [37, §III])

$$\left(\frac{x}{\mathfrak{P}}\right)_e = \frac{\alpha^\sigma}{\alpha} \quad \text{and} \quad \left(\frac{x}{\mathfrak{p}}\right)^f = \frac{\beta^\tau}{\beta},$$

where $\sigma \in \mathrm{Gal}(\mathbb{Q}(\zeta_e)/\mathbb{Q})$ is the Frobenius automorphism of $\mathbb{Q}(\zeta_e)$ at $\mathfrak{P}$ and $\tau \in \mathrm{Gal}(\mathbb{Q}(\zeta_f)/\mathbb{Q})$ is the Frobenius automorphism of $\mathbb{Q}(\zeta_f)$ at $\mathfrak{p}$. Since these power residue symbols are independent of the choice of α and β, we can without loss of generality take $\beta = \alpha^{e/f}$. Furthermore, since $\mathfrak{P}$ is a prime over $\mathfrak{p}$, the restriction of σ to $\mathbb{Q}(\zeta_f)$ is the automorphism τ. It follows that

$$\left(\frac{x}{\mathfrak{p}}\right)^f = \frac{\beta^\tau}{\beta} = \frac{(\alpha^\sigma)^{e/f}}{\alpha^{e/f}} = \left(\frac{x}{\mathfrak{P}}\right)_e^{e/f}.$$

□

[5]Actually, this is no coincidence, since in both cases it holds that $\mathsf{JS}_N(-1) = \mathsf{JS}_N(2) = \mathsf{JS}_N(-3) = 1$ if and only if $N \equiv 1 \bmod 24$.