Dual Projective Hashing and its Applications — Lossy Trapdoor Functions and More

Hoeteck Wee*

George Washington University
hoeteck@gwu.edu

Abstract. We introduce the notion of dual projective hashing. This is similar to Cramer-Shoup projective hashing, except that instead of smoothness, which stipulates that the output of the hash function looks random on NO instances, we require invertibility, which stipulates that the output of the hash function on NO instances uniquely determine the hashing key, and moreover, that there is a trapdoor which allows us to efficiently recover the hashing key.

- We show a simple construction of lossy trapdoor functions via dual projective hashing. Our construction encompasses almost all known constructions of lossy trapdoor functions, as given in the works of Peikert and Waters (STOC '08) and Freeman et al. (PKC '10).
- We also provide a simple construction of deterministic encryption schemes secure with respect to hard-to-invert auxiliary input, under an additional assumption about the projection map. Our construction clarifies and encompasses all of the constructions given in the recent work of Brakerski and Segev (Crypto '11). In addition, we obtain a new deterministic encryption scheme based on LWE.

* Supported by NSF CAREER Award CNS-0953626.

1 Introduction

In [14], Cramer and Shoup introduced a new primitive called smooth projective hashing as an abstraction of their earlier chosen-ciphertext (CCA) secure encryption scheme [13]. This primitive has since found numerous applications beyond CCA security, notably password-authenticated key exchange, two-message oblivious transfer, and leakage-resilient encryption [20, 22, 31]. In each of these cases, the connection to smooth projective hashing provided two important benefits: first, a more intuitive description and analysis of previous (sometimes seemingly ad-hoc) schemes given respectively in [26], [30, 1] and [2]; second, new instantiations based on different cryptographic assumptions, such as quadratic residuocity (QR) and decisional composite residuocity (DCR) [33].

Informally, smooth projective hashing refers to a family of hash functions $\{\mathsf{H}_k\}$ indexed by a hashing key k and whose input u comes from some "hard" language. The *projective* property stipulates that there is a projective map α defined on hashing keys such that for all YES instances u, the hash value $\mathsf{H}_k(u)$ is completely determined by u and $\alpha(k)$. In contrast, the *smoothness* property stipulates that on NO instances, $\mathsf{H}_k(\cdot)$ should be completely undetermined. Typically in applications, the hash value $\mathsf{H}_k(\cdot)$ is used to "mask" and hide an input (e.g. the plaintext in encryption, or sender's input in oblivious transfer).

1.1 Our contributions

We introduce the notion of *dual projective hashing*. As with smooth projective hashing, we consider a family of projective hash functions $\{\mathsf{H}_k\}$ indexed by a hashing key k and whose input u comes from some "hard" language. As before, we require that on YES instances u, the hash value $\mathsf{H}_k(u)$ is completely determined by u and $\alpha(k)$. On the other hand, for NO instances u, we require *invertibility* — that $\alpha(k)$ and $\mathsf{H}_k(u)$ jointly determine k; moreover, there is some inversion trapdoor that allows us to efficiently recover k given $(\alpha(k), \mathsf{H}_k(u))$ along with u. We proceed to describe two applications of dual projective hashing. In both of these applications, we will think of u as an index and k as an input to some hash function. As such, we will henceforth use $\Lambda^*_u(k)$ to denote $\mathsf{H}_k(u)$ whenever we refer to dual projective hashing.

Lossy trapdoor functions. Lossy trapdoor functions (TDF) [34] is a strengthened variant of the classical notion of trapdoor functions and were introduced with the main goal of enabling simple and black-box constructions of CCA-secure encryption. 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 signicantly 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.

Lossy trapdoor functions were introduced by Peikert and Waters [34], 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 [7], OAEP-based public-key encryption [27], "hedged" public-key encryption for protecting against bad randomness [5], security against selective opening attacks [6], and efficient non-interactive string commitments [32].

Starting from dual projective hashing, we may derive a family of lossy trapdoor functions indexed by u and given by:

$$\mathsf{F}_u : x \mapsto \alpha(x) \| \Lambda^*_u(x)$$

The injective mode is given by uniformly sampling u from NO instances, and the lossy mode is given by uniformly sampling u from YES instances. The *injective* property guarantees that if u is a NO instance, then we can efficiently recover x from the output of the function. On the other hand, the *projective* property guarantees that if u is a YES instance, then the output is fully determined by $\alpha(x)$, and therefore reveals at most $\log |\alpha(x)|$ bits of information about x.

Deterministic Encryption. Deterministic public-key encryption (where the encryption algorihtm is deterministic) was introduced by Bellare, Boldyreva and O'Neill [3], with additional constructions given in [7, 4, 11] and in concurrent works [19, 28]. Deterministic encryption has a number of practical applications, such as efficient search on encrypted data and securing legacy protocols. Our framework further clarify the constructions of deterministic encryption schemes of Boldyreva, Fehr and O'Neill [7] for high-entropy inputs and of Brakerski and Segev [11] for hard-to-invert auxiliary input (which in particular, generalize high-entropy inputs). The former combines lossy trapdoor functions and extractors, whereas the latter rely on algebraic properties of specific instantiations of lossy trapdoor functions. Specifically, the latter presented two seemingly different schemes, one based on DDH/DLIN and the other based on QR and DCR.

We consider the deterministic encryption scheme that follows from our lossy trapdoor function (following the approach used in [7]):

- the public key is a random NO instance u and the secret key is the inversion trapdoor;
- to encrypt a message M, we output $\alpha(M) \| \Lambda^*_u(M)$.

We show that:

- if $\alpha(\cdot)$ is a strong extractor (where the seed is provided by the public parameter) for high min-entropy sources, then we obtain a deterministic encryption for high min-entropy message distributions;
- if $\alpha(\cdot)$ is a "reconstructive" extractor (which is similar to a hard-core function), then we obtain a deterministic encryption secure with respect to hard-to-invert auxiliary input.

In particular, a reconstructive extractor is also a strong extractor [35], and random linear functions are both good strong extractors and good reconstructive extractors (via the left-over hash lemma [23], the Goldreich-Levin theorem [21] and generalizations there-of). These results provide a unifying framework for deterministic encryption and clarify the relation between the previous schemes and the connections to the literature on pseudorandomness. It is also interesting to contrast this construction with leakage-resilient public-key encryption derived from smooth projective hashing [31], where the extractor comes from the hash function instead of the projection map (and the seed comes from the instance instead of the public parameter).

Instantiations. We present instantiations of dual projective hashing from all three major classes of cryptographic assumptions: (1) Diffie-Hellman assumptions like DDH and DLIN, (2) number-theoretic assumptions like QR and DCR, and (3) lattice-based assumptions like LWE. Most of these instantiations are already implicit in recent works. In fact, the early constructions of hash proof systems based on DDH

and QR in [14] already satisfy the *invertibility* property, albeit inefficiently. However, since the hashing key k is encoded in the exponent, efficiently recovering the key seems as hard as computing discrete log. Instead, we will rely on hashing keys that are vectors and/or matrices over $\{0, 1\}$. (Similar constructions have been used in KDM-security [9, 10] for different technical issues.) On the other hand, the DCR-based hash proof system in [14, 12] does yield a dual projective hash function, since we can efficiently solve discrete log for base $1 + N$ over $\mathbb{Z}_{N^2}^*$, given the factorization of N.

Combining these instantiations with our generic transformations, we obtain:

- a unified treatment of almost all known constructions of lossy trapdoor functions, as given in [34, 18] (the exceptions being the QR-based constructions in [18, 29] and the one based on the Φ-hiding assumption in [27]);
- a unified treatment of both of the deterministic encryption schemes secure with respect to hard-to-invert auxiliary input given in [11];
- the first lattice-based deterministic encryption scheme that is secure with respect to hard-to-invert auxiliary input.

Relation to smooth projective hashing. Having presented the applications, we would like to highlight several conceptual differences between smooth projective hashing and dual projective hashing. In smooth projective hashing, we are interested in quantifying what the projected key and the instance tells us about the hash value, whereas in dual projective hashing, we want to know what the projected key and hash value tells us about the hashing key. Moreover, in essentially all applications of smooth projective hashing, YES instances are used for functionality/correctness and NO are used to establish security; it is the other way around for dual projective hashing. Finally, in smooth projective hashing, we use the hash value to hide information; in dual projective hashing, we publish the hash value as part of the output.

1.2 Previous work

Comparison with previous constructions. Peikert and Waters constructed lossy trapdoor functions from the DDH and LWE assumptions, and more generally, from any homomorphic encryption schemes with reusable randomness. The description of the trapdoor functions in their constructions are a matrix of ciphertexts, and evaluation corresponds to matrix multiplication. Hemenway and Ostrovsky [24] constructed lossy trapdoor functions from smooth projective hashing where the hash function is homomorphic, which may in turn be instantiated from the QR, DDH and DCR assumptions. The construction is syntactically very similar to the matrix-based construction in [34] (although the analysis is somewhat different): the description of the trapdoor functions are a matrix of hash values, and evaluation corresponds to matrix multiplication.

Freeman et al. [18] gave direct constructions of lossy trapdoor functions from the QR, DCR and DLIN assumptions. Each of these constructions are fairly different and there is no evident unifying theme to these constructions. Specifically, the DLIN construction is a variant of the matrix-based scheme in [34]. Mol and Yilek [29] and Bellare et al. [4] independently constructed lossy trapdoor functions from the QR and the DCR assumptions respectively. We note that the QR-based schemes in these two papers only handle bounded lossiness.

In contrast to the Hemenway-Ostrovsky construction, our construction does not rely on smoothness nor any algebraic structure on the underlying hash proof system; we also have a more direct transformation from

the hash function to the lossy trapdoor function, which are syntactically and conceptually quite different from that in [24]. (For instance, we use NO instances for injective functions and YES instances for lossy functions, and it is the other way around in [24].) On the other hand, in order to instantiate the hash functions, we do rely on a vector/matrix of values, similar to the constructions developed in the different context of key dependent message security and leakage resilience [9, 31, 10].

Additional related work. A lossy encryption scheme is a standard public-key encryption scheme where the public key may be generated in one of two modes: in the *injective* mode, the ciphertext uniquely determines the plaintext and there is an associated secret key which allows correct decryption, and in the *lossy* mode, the ciphertext reveals no information about the plaintext [6]. Given a lossy trapdoor function, it is easy to construct a lossy encryption scheme [6]. Hemenway et al. [25] gave a direct construction of a lossy encryption scheme from any hash proof system. In the construction, the public key is also an instance from the language. However, the usage is reversed: for lossy encryption, the injective mode uses a YES instance, and the lossy mode uses a NO instance.

Organization. We formalize dual projective hashing in Section 2. We present both the definition and our results on lossy trapdoor functions in Section 3, and those for deterministic encryption in Section 4. We present the instantiations of dual projective hashing in Sections 5 through 8.

Notation. We denote by $s \leftarrow_{\mathrm{R}} S$ the fact that s is picked uniformly at random from a finite set S and by $x, y, z \leftarrow_{\mathrm{R}} S$ that all x, y, z are picked independently and uniformly at random from S. We denote by $\mathrm{negl}(\cdot)$ a negligible function. By PPT, we denote a probabilistic polynomial-time algorithm. Throughout, we use 1^λ as the security parameter. We use $\cdot$ to denote multiplication (or group operation) as well as component-wise multiplication. We use boldface to denote vectors (always column vectors) and matrices.

2 Dual Projective Hashing

In this section, we describe dual projective hashing more formally. We warn the reader that we use slightly different notation from the outline given in the introduction (in particular, we denote the input to $\Lambda^*_u(\cdot)$ by x instead of k).

Setup. There is a setup algorithm Setup that given the security parameter 1^λ, outputs the public parameters HP for the hash function. All algorithms are given HP as part of its input; we omit HP henceforth whenever the context is clear. Associated with each HP are a pair of disjoint sets Π_{Y} and Π_{N} corresponding to YES and NO instances respectively. We require that the uniform distributions over each of Π_{Y} and Π_{N} be efficiently samplable. Specifically, there exist a pair of sampling algorithms: $\mathsf{SampYes}(\mathrm{HP})$ outputs a random pair of values (u, w) where the first output u is uniformly distributed over Π_{Y} and w is the corresponding witness; $\mathsf{SampNo}(\mathrm{HP})$ outputs a random pair of values (u, τ) where the first output u is uniformly distributed over Π_{N} and τ is the corresponding trapdoor. We discuss the roles of the witness and the trapdoor below.

Subset membership assumption. The *subset membership assumption* states that the uniform distributions over Π_Y and Π_N are computationally indistinguishable, even given HP. More formally, for an adversary $\mathcal{A}$, we consider the advantage function $\mathsf{AdvSubset}^{\mathcal{A}}(\lambda)$ given by

$$\Pr\left[\mathcal{A}(\text{HP}, u) = 1 : \text{HP} \leftarrow \mathsf{Setup}(1^\lambda), u \leftarrow_R \Pi_Y\right] - \Pr\left[\mathcal{A}(\text{HP}, u) = 1 : \text{HP} \leftarrow \mathsf{Setup}(1^\lambda), u \leftarrow_R \Pi_N\right]$$

The subset membership assumption states that for all PPT $\mathcal{A}$, the advantage $\mathsf{AdvSubset}^{\mathcal{A}}(\lambda)$ is a negligible function in λ.

Projective hashing. Fix a public parameter HP. We consider a family of hash functions $\{\Lambda^*_u(\cdot)\}$ indexed by an instance $u \in \Pi_Y \cup \Pi_N$. We also require that the hash function be efficiently computable; we call the algorithm for computing $\Lambda^*_u(\cdot)$ the private evaluation algorithm. We say that $\Lambda^*_u(\cdot)$ is *projective* if there exists a projection map $\alpha(\cdot)$ such that for all $u \in \Pi_Y$ and for all inputs x, $\alpha(x)$ completely determines $\Lambda^*_u(x)$. Specifically, we require that there exists an efficient public evaluation algorithm Pub that on input $\alpha(x)$ and for all $(u, w) \leftarrow \mathsf{SampYes}(\text{HP})$, outputs $\Lambda^*_u(x)$. That is,

$$\mathsf{Pub}(\alpha(x), u, w) = \Lambda^*_u(x)$$

Invertibility. We say that $\Lambda^*_u(\cdot)$ is *invertible* if there is an efficient trapdoor inversion algorithm TdInv that for all $(u, \tau) \leftarrow \mathsf{SampNo}(\text{HP})$ and for all x, recovers x given $(\alpha(x), \Lambda^*_u(x))$ and the trapdoor τ. That is,

$$\mathsf{TdInv}(\tau, \alpha(x), \Lambda^*_u(x)) = x$$

We note here that for two of our factoring-related instantiations, $\mathsf{SampNo}(\text{HP})$ also requires as input the coin tosses used to sample HP in order to compute the inversion trapdoor (there, HP is a public RSA modulus N and τ is the factorization of N). For these instantiations, we cannot treat HP as a global system parameter; instead, it will be part of the public key in the case of deterministic encryption and part of the function index in the case of lossy trapdoor functions. We suppress this subtlety in our main constructions since SampNo is only used for functionality and not in the proof of security.

3 Lossy Trapdoor Functions

In this section, we present our results on lossy trapdoor functions. We first describe the definition of lossy TDFs given in [34].

Definition 1 (Lossy Trapdoor Functions). A collection of (m, k)-*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.) $\mathsf{G}_0(1^\lambda)$ outputs a function index u.
2. (SAMPLING AN INJECTIVE FUNCTION.) $\mathsf{G}_1(1^\lambda)$ outputs a pair (u, τ) where u is a function index and τ is a trapdoor.
3. (EVALUATION OF LOSSY FUNCTIONS.) For every function index u produced by G_0, the algorithm $\mathsf{F}(u, \cdot)$ computes a function $f_u : \{0,1\}^m \to \{0,1\}^*$, whose image is of size at most 2^{m-k}.
4. (EVALUATION OF INJECTIVE FUNCTIONS.) For every pair (u, τ) produced by G_1, the algorithm $\mathsf{F}(u, \cdot)$ computes a injective function $f_u : \{0,1\}^m \to \{0,1\}^*$.
5. (INVERSION OF INJECTIVE FUNCTIONS.) For every pair (u, 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 first outputs of $\mathsf{G}_0(1^\lambda)$ and $\mathsf{G}_1(1^\lambda)$ are computationally indistinguishable.

Here λ is the security parameter, and the value k is called the *lossiness*.

Our construction. Given a dual projective hash function, we may construct a family of lossy trapdoor functions, as shown in Fig 1.

Lossy TDF

(SAMPLING A LOSSY FUNCTION.) $\mathsf{G}_0(1^\lambda)$: Run $(u, w) \leftarrow \mathsf{SampYes}(\text{HP})$. Output $\text{HP}\|u$.

(SAMPLING AN INJECTIVE FUNCTION.) $\mathsf{G}_1(1^\lambda)$: Run $(u, \tau) \leftarrow \mathsf{SampNo}(\text{HP})$. Output $(\text{HP}\|u, \tau)$.

(EVALUATION.) $\mathsf{F}(u, x)$: Output $\alpha(x)\|\Lambda^*_u(x)$.

(INVERSION.) $\mathsf{F}^{-1}(\tau, y_0\|y_1)$: Output $\mathsf{TdInv}(\tau, y_0, y_1)$.

Note: We assume here all algorithms receive as input $\text{HP} \leftarrow \mathsf{Setup}(1^\lambda)$.

Fig. 1. Lossy TDF from dual projective hashing

Theorem 1. *Under the subset membership assumption, the above construction yields a collection of* $(m, m - \log|\operatorname{Im}\alpha|)$*-lossy trapdoor functions.*

Proof. Correctness for injective functions follows readily from the invertibility property. Lossiness for lossy functions follows readily from the projective property, which implies that for $u \in \Pi_Y$, $|\operatorname{Im} f_u| \leq |\operatorname{Im}\alpha|$. Security is equivalent to the subset membership assumption. □

4 Deterministic Encryption

In this section, we present our results for deterministic encryption. We begin with the definition, then some results about extractors, and finally our construction.

4.1 Deterministic encryption

A deterministic encryption scheme is a triplet of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where Gen is randomized and $\mathsf{Enc}, \mathsf{Dec}$ are deterministic. Via $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$, the randomized key-generation algorithm, produces public/secret keys for security parameter 1^λ. Enc on input a public key PK and a message M, produces a ciphertext. $\mathsf{Dec}(\text{SK}, \psi)$ on input a secret key SK and a ciphertext ψ, outputs a message. We require correctness, namely that with overwhelming probability over (PK, SK), for all M, $\mathsf{Dec}(\mathsf{Enc}(M)) = M$.

Hard-to-invert auxiliary inputs. Following [11, 16, 17], we consider auxiliary input $f(x)$ from which it is hard to recover x. The source of hardness may be any combination of information-theoretic hardness (where the function is many-to-one) and computational hardness (e.g. if f is a one-way permutation). An efficiently computable function $\mathcal{F} = \{f_\lambda\}$ is *δ-hard-to-invert w.r.t. an efficiently samplable distribution $\mathcal{D}$* if for every PPT algorithm $\mathcal{A}$, it holds that $\Pr[\mathcal{A}(1^\lambda, f_\lambda(x)) = x] \leq \delta$ where the probability is taken over $x \leftarrow_{\mathrm{R}} \mathcal{D}$ and over the internal coin tosses of $\mathcal{A}$.

Security with auxiliary input. We follow the definition of security for deterministic encryption with auxiliary input from [11, 3, 4, 7].[1] For simplicity, we will only consider security while encrypting a single message, although our proofs of security extend to multiple messages and block-wise hard-to-invert auxiliary inputs. For an adversary $\mathcal{A}$, auxiliary input function $\mathcal{F}$ and message distribution $\mathcal{M}$ over $\{0,1\}^m$, we define the advantage function

$$\mathsf{AdvPrivSInd}^{\mathcal{A},\mathcal{F},\mathcal{M}}(\lambda) := \Pr\left[b = b' : \begin{array}{l} (M_0, M_1) \leftarrow \mathcal{M}; \\ (\mathrm{PK}, \mathrm{SK}) \leftarrow \mathsf{Gen}(1^\lambda); \\ b \leftarrow_{\mathrm{R}} \{0,1\}; \\ \psi \leftarrow \mathsf{Enc}(\mathrm{PK}, M_b); \\ b' \leftarrow \mathcal{A}(\mathrm{PK}, \psi, f(M_0), f(M_1)) \end{array}\right] - \frac{1}{2}$$

A deterministic encryption scheme is $(\mathcal{F}, \mathcal{M})$-PrivSInd secure if for all PPT $\mathcal{A}$, the advantage $\mathsf{AdvPrivSInd}^{\mathcal{A},\mathcal{F},\mathcal{M}}(\lambda)$ is a negligible function in λ.

4.2 Extractors

Reconstructive extractors. A (ϵ, δ)-*reconstructive extractor* is a pair of functions $(\mathsf{Ext}, \mathsf{Rec})$:

- an extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \Sigma$
- a (uniform) oracle machine Rec that on input $(1^n, 1/\epsilon)$ runs in time $\mathrm{poly}(n, 1/\epsilon, \log|\Sigma|)$.

that satisfy the following property: for every $x \in \{0,1\}^n$ and every function D such that

$$\left| \Pr_{r \leftarrow_{\mathrm{R}} \{0,1\}^d}[D(r, \mathsf{Ext}(x,r)) = 1] - \Pr_{r \leftarrow_{\mathrm{R}} \{0,1\}^d, \sigma \leftarrow_{\mathrm{R}} \Sigma}[D(r,\sigma) = 1] \right| \geq \epsilon$$

we have:

$$\Pr[\mathsf{Rec}^D(1^n, 1/\epsilon) = x] \geq \delta$$

where the probability is over the coin tosses of Rec.

It was shown in [35] that any (ϵ, δ)-reconstructive extractor is a (strong) extractor for sources of min-entropy roughly $\log 1/\delta$. It is also easy to show that the output of any (ϵ, δ)-reconstructive extractor is pseudorandom for $\delta \cdot \mathrm{negl}(\cdot)$-hard-to-invert auxiliary inputs.

Extractors from linear functions. It turns out that random linear functions are not only good randomness extractors (a fact commonly referred to as the left-over hash lemma), but also good reconstructive extractors.

[1] Specifically, we use the notion of *strong indistinguishability* (PRIV-sIND) [11, Definition 4.4] restricted to single messages.

Lemma 1 ([21, 17, 10]). *Let q be a prime. Then, the function* $\mathsf{Ext} : \{0,1\}^n \times \mathbb{Z}_q^n \to \mathbb{Z}_q$ *given by* $(\mathbf{x}, \mathbf{a}) \mapsto \mathbf{x}^\top \mathbf{a}$ *is a* $(\epsilon, \frac{\epsilon^3}{512 \cdot n \cdot q^2})$*-reconstructive extractor.*

That is, Ext maps $(x_1, \ldots, x_n), (a_1, \ldots, a_n)$ to $a_1 x_1 + \cdots a_n x_n \pmod q$. Moreover, the lemma extends to the following settings:

- q is a random RSA modulus, assuming that factoring is hard on average.
- $\mathbb{G}$ is a group of prime order q with generator g, and we consider the extractor $\mathsf{Ext} : \{0,1\}^n \times \mathbb{G}^n \to \mathbb{G}$ given by $(\mathbf{x}, g^{\mathbf{a}}) \mapsto g^{\mathbf{x}^\top \mathbf{a}}$.

4.3 Our construction

Given a dual projective hash function, we may construct a deterministic encryption scheme, as shown in Fig 2. For this construction, it is important that we state explicitly that the projection map $\alpha(\cdot)$ takes the public parameter HP as its first input.

Deterministic Encryption Scheme

(KEY GENERATION.) $\mathsf{Gen}(1^\lambda)$: Run $\text{HP} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(u, \tau) \leftarrow \mathsf{SampNo}(\text{HP})$. Output

$$\text{PK} := \text{HP} \| u \quad \text{and} \quad \text{SK} := \tau$$

(ENCRYPTION.) $\mathsf{Enc}(\text{PK}, M)$: On input $\text{PK} = \text{HP} \| u$ and message M, output the ciphertext

$$\alpha(\text{HP}, M) \| \Lambda_u^*(M)$$

(DECRYPTION.) $\mathsf{Dec}(\text{SK}, \psi)$: On input $\text{SK} = \tau$ and ciphertext $\psi = y_0 \| y_1$, output

$$\mathsf{TdInv}(\tau, y_0, y_1)$$

Fig. 2. Deterministic encryption scheme from dual projective hashing

---

Theorem 2. *If* $(x, \text{HP}) \mapsto \alpha(\text{HP}, x)$ *is a* (ϵ, δ)*-reconstructive extractor and the subset membership assumption holds, then the encryption scheme as shown above is* PrivSInd*-secure with respect to hard-to-invert auxiliary input.*

Correctness of the encryption scheme follows readily the invertibility property of dual projective hashing. IND-PRIV security follows from the next technical claim.

Lemma 2. *Let $\mathcal{A}$ be an adversary against* $(\mathcal{F}, \mathcal{M})$-PrivSInd *security of the above encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. *Then, we can construct adversaries $\mathcal{A}_0$ and $\mathcal{A}_1$ such that for any ϵ:*

$$\begin{aligned} \textit{either} \qquad & \mathsf{AdvPrivSInd}^{\mathcal{A},\mathcal{F},\mathcal{M}}(\lambda) \leq \mathsf{AdvSubset}^{\mathcal{A}_0}(\lambda) + 2\epsilon \\ \textit{or} \qquad & \Pr_{M \leftarrow \mathcal{M}}\left[\mathcal{A}_1(f(M)) = M\right] \geq \delta\epsilon \end{aligned}$$

The running time of $\mathcal{A}_0$ is roughly that of $\mathcal{A}$ and the running time of $\mathcal{A}_1$ is $\mathrm{poly}(n, 1/\epsilon, \log|\Sigma|)$ times that of $\mathcal{A}$.

Proof. We proceed via a sequence of games. We start with Game 0 as in the PrivSInd experiment and end up with a game where the view of $\mathcal{A}$ is statistically independent of the challenge bit b. We write $u \in \Pi_{\mathrm{N}}$ to denote the public key PK in Game 0. This means that the view of the adversary $\mathcal{A}$ is given by:

$$\left\langle \mathrm{HP} \| u, \alpha(\mathrm{HP}, M_b) \,\|\, \Lambda^*_u(M_b), f(M_0), f(M_1) \right\rangle$$

GAME 1: SWITCHING TO $u \leftarrow_{\mathrm{R}} \Pi_{\mathrm{Y}}$. We replace $u \leftarrow_{\mathrm{R}} \Pi_{\mathrm{N}}$ with sampling $(u, w) \leftarrow \mathsf{SampYes}(\mathrm{HP})$. Clearly, Game 0 and 1 are computationally indistinguishable by hardness of subset membership, and the advantage of the adversary changes by at most $\mathsf{AdvSubset}(\lambda)$.

GAME 2: ENCRYPTING USING Pub. In the challenge ciphertext, we replace $\Lambda^*_u(M_b)$ with $\mathsf{Pub}(\alpha(\mathrm{HP}, M_b), u, w)$. By the projective property, Games 1 and 2 are identically distributed.

GAME 3: SWITCHING THE OUTPUT OF $\alpha(\cdot)$ TO RANDOM. We replace $\alpha(\mathrm{HP}, M_b)$ in the challenge ciphertext with a random $\sigma \leftarrow_{\mathrm{R}} \Sigma$. That is, we change the ciphertext from

$$\alpha(\mathrm{HP}, M_b) \| \mathsf{Pub}(\alpha(\mathrm{HP}, M_b), u, w) \quad \text{to} \quad \sigma \| \mathsf{Pub}(\sigma, u, w)$$

If the advantage of the adversary from Game 2 to Game 3 changes by at most 2ϵ, then we are done. Otherwise, we may use $\mathcal{A}$ to construct a distinguisher D such that

$$\left| \Pr\big[D\big(\mathrm{HP}, \alpha(\mathrm{HP}, m), f(M)\big) = 1\big] - \Pr\big[D\big(\mathrm{HP}, \sigma, f(M)\big) = 1\big] \right| > 2\epsilon$$

where $\mathrm{HP} \leftarrow \mathsf{Setup}(1^\lambda), M \leftarrow \mathcal{M}, \sigma \leftarrow_{\mathrm{R}} \Sigma$. ($D$ simply chooses $b \leftarrow_{\mathrm{R}} \{0,1\}$, uses its input as M_b, chooses $M_{1-b} \leftarrow_{\mathrm{R}} \mathcal{M}$, simulates the view of $\mathcal{A}$ using $\mathsf{Pub}(\cdot, u, w)$ to obtain an output b' and outputs 1 if $b' = b$.) By an averaging argument, with probability ϵ over $M \leftarrow \mathcal{M}$, D achieves distinguishing probability ϵ, upon which we can use Rec^D to compute M from $f(M)$ with probability δ. This means that we can invert f on the distribution $\mathcal{M}$ with probability $\epsilon \cdot \delta$.

We conclude by observing that in Game 3, the view of the adversary is statistically independent of the challenge bit b. Hence, the probability that $b' = b$ is exactly $1/2$. □

Remark 1. It follows fairly readily from the analysis that if $(x, \mathrm{HP}) \mapsto \alpha(\mathrm{HP}, x)$ is a strong extractor (which is a weaker guarantee than a reconstructive extractor), then the above encryption scheme is PrivSInd-secure with respective to high min-entropy inputs. We defer the details and a more precise statement to the full version of this paper. We also point out here that the distribution for HP must be independent of the message distribution $\mathcal{M}$ (for the same reason the seed to an extractor must be chosen independently of the weaker random source). For this reason, all known constructions of deterministic encryption only achieve security for message distributions that do not depend on the public key.

5 Instantiations from DDH and DLIN

Let $\mathbb{G}$ be a group of prime order q specified using a generator g. The DDH assumption asserts that g^{ab} is pseudorandom given g, g^a, g^b where $g \leftarrow_{\mathrm{R}} \mathbb{G}; a, b \leftarrow_{\mathrm{R}} \mathbb{Z}_q$. The d-LIN assumption asserts that $g_{d+1}^{r_1+\cdots+r_d}$ is pseudorandom given $g_1, \ldots, g_{d+1}, g_1^{r_1}, \ldots, g_d^{r_d}$ where $g_1, \ldots, g_{d+1} \leftarrow_{\mathrm{R}} \mathbb{G}; r_1, \ldots, r_d \leftarrow_{\mathrm{R}} \mathbb{Z}_q$. DDH is

equivalent to 1-LIN. We present the DLIN-based hash proof system in [9, 31], also used in [18, 11]. When instantiated with our generic transformations, this yields the DLIN-based $(m, m - d \log q)$-lossy trapdoor functions given in [18] and the DLIN-based deterministic encryption scheme in [11].

Setup. $\text{HP} := (\mathbb{G}, g^{\mathbf{P}}), \mathbf{P} \leftarrow_{\text{R}} \mathbb{Z}_q^{d \times m}$. The language is given by

$$\Pi_{\text{Y}} := \left\{ g^{\mathbf{WP}} : \mathbf{W} \in \mathbb{Z}_q^{m \times d} \right\} \qquad \text{and} \qquad \Pi_{\text{N}} := \left\{ g^{\mathbf{A}} : \mathbf{A} \in \mathbb{Z}_q^{m \times m} \text{ with full rank} \right\}$$

A uniformly chosen matrix $\mathbf{A} \leftarrow_{\text{R}} \mathbb{Z}_q^{m \times m}$ has full rank with overwhelming probability, so Π_{N} is efficiently samplable via rejection sampling. The uniform distributions over Π_{Y} and Π_{N} are computationally distinguishable under the d-LIN assumption as shown in [31, 9].

Hashing. The hashing input is given by $\mathbf{x} \in \{0,1\}^m$, with

$$\alpha(g^{\mathbf{P}}, \mathbf{x}) := g^{\mathbf{Px}}$$

Private and public evaluation are given by:

$$\Lambda^*_{\mathbf{U}}(\mathbf{x}) := \mathbf{U}^{\mathbf{x}} \in \mathbb{G}^m \qquad \text{and} \qquad \mathsf{Pub}(g^{\mathbf{Px}}, \mathbf{U}, \mathbf{W}) := g^{\mathbf{W} \cdot \mathbf{Px}}$$

where $(\mathbf{U}^{\mathbf{x}})_i := \sum_{j=1}^m \mathbf{U}_{ij}^{\mathbf{x}_j}$. Observe that for $\mathbf{U} = g^{\mathbf{WP}} \in \Pi_{\text{Y}}$, we have

$$\Lambda^*_{\mathbf{U}}(\mathbf{x}) = g^{\mathbf{WPx}} = \mathsf{Pub}(g^{\mathbf{Px}}, \mathbf{U}, \mathbf{W})$$

Inversion. The inversion trapdoor is $\mathbf{A}^{-1}$. Observe that for $\mathbf{U} = g^{\mathbf{A}} \in \Pi_{\text{N}}$, we have

$$\Lambda^*_{\mathbf{U}}(\mathbf{x}) = g^{\mathbf{Ax}}$$

Given the inversion trapdoor $\mathbf{A}^{-1}$ and $\Lambda^*_{\mathbf{U}}(\mathbf{x})$, we can compute $g^{\mathbf{x}}$ and thus $\mathbf{x}$.

6 Instantiations from QR

Fix a Blum integer $N = PQ$ for safe primes $P, Q \equiv 3 \pmod 4$ (such that $P = 2p + 1$ and $Q = 2q + 1$ for primes p, q). Let $\mathbb{J}_N$ denote the subgroup of $\mathbb{Z}_N^*$ with Jacobi symbol $+1$, and let $\mathbb{QR}_N$ denote the cyclic subgroup of quadratic residues. Observe that $|\mathbb{J}_N| = 2pq = 2|\mathbb{QR}_N|$. The QR assumption states that the uniform distributions over $\mathbb{QR}_N$ and $\mathbb{J}_N \setminus \mathbb{QR}_N$ are computationally indistinguishable.

First construction. We present a QR-based hash proof system based on the IBE scheme of Boneh et. al [8]. When instantiated with our generic transformations, this yields a new family of QR-based $(\log \phi(N) - 1, 1)$-lossy trapdoor functions; however, it is less efficient than that given in [18].

Setup. $\text{HP} := (N)$. The language is given by

$$\Pi_{\text{Y}} := \mathbb{QR}_N \qquad \text{and} \qquad \Pi_{\text{N}} := \mathbb{J}_N \setminus \mathbb{QR}_N$$

The uniform distributions over Π_{Y} and Π_{N} are computationally indistinguishable under the QR assumption.

Hashing. The hashing input is given by $x \in \mathbb{Z}_N^*/\{\pm 1\}$, with

$$\alpha(x) := x^2$$

Private and public evaluation are given by:

$$\Lambda^*_u(x) := J(f(x)) \qquad \text{and} \qquad \mathsf{Pub}(N, u, w) := J(g(w))$$

where f, g are the polynomials obtained by running the "IBE compatible algorithm" [8, Section 4] on inputs x^2, u. For $u = w^2 \in \Pi_{\text{Y}}$, we have $J(f(x)) = J(g(w))$ by correctness of the IBE compatible algorithm.

Inversion. The inversion trapdoor (which depends on HP) is the factorization of N. For $u = -w^2 \in \Pi_{\text{N}}$, we have $J(f(x))$ is equally likely to be 1 and -1 given x^2. Given the inversion trapdoor (i.e. the factorization of N), we can compute all four square roots $\pm x_0, \pm x_1$ of x^2 along with both $J(f(x_0))$ and $J(f(x_1))$; we can then recover x.

Second construction. We present a QR-based hash proof system implicit in [11, 24], which is a matrix analogue of original Cramer-Shoup construction [14]. When instantiated with our generic transformations, this yields the QR-based $(m, m - \log|\phi(N)|)$-lossy trapdoor functions in [24]. and the QR-based deterministic encryption scheme in [11].

Setup. $\text{HP} := (N, g^{\mathbf{p}}), \mathbf{p} \leftarrow_{\text{R}} \mathbb{Z}^m_{N/2}, g \leftarrow_{\text{R}} \mathbb{QR}_N$. The language is given by

$$\Pi_{\text{Y}} := \left\{ g^{\mathbf{w}\mathbf{p}^\top} : \mathbf{w} \in \mathbb{Z}^m_{N/2} \right\} \qquad \text{and} \qquad \Pi_{\text{N}} := \left\{ (-1)^{\mathbf{I}_m} \cdot g^{\mathbf{w}\mathbf{p}^\top} : \mathbf{w} \in \mathbb{Z}^m_{N/2} \right\}$$

where in the expression for Π_{N}, the matrix dot product refers to element-wise multiplication. The uniform distributions over Π_{Y} and Π_{N} are computationally indistinguishable under the QR assumption as shown in [11, 24, 10].

Hashing. The hashing input is given by $\mathbf{x} \in \{0,1\}^m$, with

$$\alpha(g^{\mathbf{p}}, \mathbf{x}) := g^{\mathbf{p}^\top \mathbf{x}} \in \mathbb{Z}^*_N$$

Here, $\Lambda^*_{\mathbf{U}} : \{0,1\}^m \to (\mathbb{Z}^*_N)^m$, with private and public evaluation given by:

$$\Lambda^*_{\mathbf{U}}(\mathbf{x}) := \mathbf{U}^{\mathbf{x}} \qquad \text{and} \qquad \mathsf{Pub}(\text{PK}, \mathbf{U}, \mathbf{w}) := \text{PK}^{\mathbf{w}}$$

where $(\mathbf{U}^{\mathbf{x}})_i := \sum_{j=1}^m \mathbf{U}_{ij}^{\mathbf{x}_j}$. Observe that for $\mathbf{U} = g^{\mathbf{w}\mathbf{p}^\top} \in \Pi_{\text{Y}}$, we have

$$\Lambda^*_{\mathbf{U}}(\mathbf{x}) = g^{\mathbf{w}\mathbf{p}^\top \mathbf{x}} = (g^{\mathbf{p}^\top \mathbf{x}})^{\mathbf{w}} = \mathsf{Pub}(\text{PK}, \mathbf{U}, \mathbf{w})$$

Inversion. The inversion trapdoor is the vector $\mathbf{w}$. Observe that for $\mathbf{U} = (-1)^{\mathbf{I}_m} \cdot g^{\mathbf{w}\mathbf{p}^\top} \in \Pi_{\text{N}}$, we have

$$\Lambda^*_{\mathbf{U}}(\mathbf{x}) = (-1)^{\mathbf{x}} \cdot g^{\mathbf{w}\mathbf{p}^\top \mathbf{x}} = (-1)^{\mathbf{x}} \cdot \text{PK}^{\mathbf{w}}$$

Given the inversion trapdoor $\mathbf{w}$ and $\Lambda^*_{\mathbf{U}}(\mathbf{x})$, we can compute $(-1)^{\mathbf{x}}$ and thus $\mathbf{x}$.

7 Instantiations from DCR

Fix a Blum integer $N = PQ$ for safe primes $P, Q \equiv 3 \pmod 4$ (such that $P = 2p + 1$ and $Q = 2q + 1$ for primes p, q). Let $m \in \mathbb{Z}^+$ be a parameter. The group $\mathbb{Z}^*_{N^{m+1}}$ is isomorphic to $\mathbb{Z}_{\phi(N)} \times \mathbb{Z}_{N^m}$.

First construction. We present the Cramer-Shoup DCR-based hash proof system [14], extended to the Damgård-Jurik scheme [15]. When instantiated with our generic transformation, this yields the DCR-based $(m \log N, m \log N - \log |\phi(N)|)$-lossy trapdoor functions given in [18].

Setup. $\text{HP} := (N, g^{N^m}), g \leftarrow_{\text{R}} \mathbb{Z}^*_{N^{m+1}}$. The language is given by

$$\Pi_{\text{Y}} := \left\{ g^{N^m w} : w \in \mathbb{Z}_{N^m} \right\} \qquad \text{and} \qquad \Pi_{\text{N}} := \left\{ g^{N^m w}(1+N) : w \in \mathbb{Z}_{N^m} \right\}$$

The uniform distributions over Π_{Y} and Π_{N} are computationally indistinguishable under the DCR assumption, as shown in [15].

Hashing. The hashing input is given by $x \in \mathbb{Z}_{N^m}$, with

$$\alpha(g^{N^m}, x) := g^{N^m x}$$

Private and public evaluation are given by:

$$\Lambda^*_u(x) := u^x \qquad \text{and} \qquad \mathsf{Pub}(\text{PK}, u, w) := \text{PK}^w$$

Observe that for $u = g^{N^m w} \in \Pi_{\text{Y}}$, we have

$$\Lambda^*_u(x) = g^{N^m wx} = (g^{N^m x})^w = \mathsf{Pub}(\text{PK}, u, w)$$

Inversion. The inversion trapdoor (which depends on HP) is the factorization of N. For $u = g^{N^m w}(1+N) \in \Pi_{\text{N}}$, we have

$$\Lambda^*_u(x) = g^{N^m wx}(1+N)^x$$

Given the inversion trapdoor (i.e. factorization of N), we can efficiently compute x from $g^{N^m wx}(1+N)^x$, c.f. [15].

Second construction. There is a second DCR-based hash proof system implicit in [11], which is a matrix analogue of original Cramer-Shoup construction [14]. It is similar to the second QR-based construction, except we replace (-1) with $1+N$. When instantiated with our generic transformations, this yields the DCR-based deterministic encryption scheme in [11].

8 Instantiations from LWE

We present the LWE-based construction, which is based on the lossy trapdoor functions in [34, Section 6.3]. For a real parameter $\beta \in (0,1)$, we denote by Ψ_β the distribution over $\mathbb{R}/\mathbb{Z}$ of a normal variable with mean 0 and standard deviation $\beta/\sqrt{2\,\pi}$ then reduced modulo 1. Denote by $\bar{\Psi}_\beta$ the discrete distribution over $\mathbb{Z}_q$ of the random variable $\lfloor q\,X \rceil \mod q$ where the random variable X has distribution Ψ_β.

In the following, we consider the standard LWE parameters m, n, q as well as additional parameters $\tilde{n}, p$ such that

$$m = O(n \log q) \quad \text{and} \quad \beta = \Theta(1/q) \quad \text{and} \quad p \leq q/4n \quad \text{and} \quad \tilde{n} = m/\log p$$

In particular, fix $\gamma < 1$ to be a constant. Then, we will set

$$q = \Theta(n^{1+1/\gamma}) \quad \text{and} \quad p = \Theta(n^{1/\gamma})$$

When instantiated with our generic transformations, this yields the LWE-based lossy trapdoor functions in [34] and a new LWE-based deterministic encryption scheme.

Setup. $\text{HP} := \mathbf{A} \leftarrow_{\text{R}} \mathbb{Z}_q^{n \times m}$. The language is given by

$$\Pi_{\text{Y}} \leftarrow_{\text{R}} \mathbf{A}^\top \mathbf{S} + \mathbf{E} \qquad \text{and} \qquad \Pi_{\text{N}} \leftarrow_{\text{R}} \mathbf{A}^\top \mathbf{S} + \mathbf{E} + \mathbf{G}$$

where $\mathbf{S} \leftarrow_{\text{R}} \mathbb{Z}_q^{n \times \tilde{n}}, \mathbf{E} \leftarrow_{\text{R}} (\bar{\Psi}_\beta)^{m \times \tilde{n}}$ and $1/\beta \geq 16pn$. Here, $\mathbf{G} \in \mathbb{Z}_q^{m \times \tilde{n}}$ is a fixed public matrix with special structure for which the bounded error-decoding problem is easy (see [34, Section 6.3.2]). These distributions are computationally distinguishable under LWE.

Hashing. The hashing input is given by a column vector $\mathbf{x} \leftarrow_{\text{R}} \{0,1\}^m$, with

$$\alpha(\mathbf{A}, \mathbf{x}) := \mathbf{A}\mathbf{x} \in \mathbb{Z}_q^n$$

Here, $\Lambda_{\mathbf{U}}^* : \{0,1\}^m \to \mathbb{Z}_q^{\tilde{n}}$, with private and public evaluation given by:

$$\Lambda_{\mathbf{U}}^*(\mathbf{x}) := \mathbf{x}^\top \mathbf{U} \qquad \text{and} \qquad \mathsf{Pub}(\mathbf{p}, \mathbf{U}, \mathbf{S}) := \mathbf{p}^\top \mathbf{S}$$

The projective property is approximate, that is,

$$\mathbf{x}^\top (\mathbf{A}^\top \mathbf{S} + \mathbf{E}) \approx (\mathbf{A}\mathbf{x})^\top \mathbf{S}$$

In fact, with overwhelming probability over $\mathbf{E}$, for all $\mathbf{x}$, we have $\mathbf{x}^\top \mathbf{E} \subseteq [q/p]^{\tilde{n}}$. That is, the projective property holds up to an additive error term in $[q/p]^{\tilde{n}}$.

Inversion. The inversion trapdoor is the matrix $\mathbf{S}$. For $\mathbf{U} \leftarrow \Pi_{\text{N}}$, we have

$$(\alpha(\mathbf{A}, \mathbf{x}), \Lambda_{\mathbf{U}}^*(\mathbf{x})) = (\mathbf{A}\mathbf{x}, (\mathbf{A}\mathbf{x})^\top \mathbf{S} + \mathbf{x}^\top \mathbf{E} + \mathbf{x}^\top \mathbf{G})$$

Given $\mathbf{S}$, we can recover $\mathbf{x}^\top \mathbf{E} + \mathbf{x}^\top \mathbf{G}$. The quantity $\mathbf{x}^\top \mathbf{E}$ has small norm, so we can do bounded-error decoding to recover $\mathbf{x}^\top \mathbf{G}$ and thus $\mathbf{x}$.

Lossy TDF. For lossy TDF, in the lossy mode, we can bound the size of the image by $|\operatorname{Im} \alpha| \cdot (q/p)^{\tilde{n}}$, where the latter term accounts for the error incurred by the approximate projective property. That is, the lossiness is given by

$$m - \left(n \log q + \frac{m}{\log p} \log \frac{q}{p} \right) = (1 - \gamma)m - n \log q$$

Deterministic Encryption. For deterministic encryption, the adversary $\mathcal{A}_1$ will guess the error term $\mathbf{x}^\top \mathbf{E}$, which incurs a multiplicative loss of $(p/q)^{\tilde{n}} = 1/2^{\gamma m}$. The rest of the security loss is $q^{2n} \cdot \text{poly}(m, \lambda)$. This means that for every constant $\gamma < 1$, we have a deterministic encryption scheme for m-bit messages, secure with respect to $2^{-\gamma m}$-hard-to-invert auxiliary input, based on the hardness of solving certain lattice problems with approximation factor better than $\tilde{O}(n^{2+1/\gamma})$.

Acknowledgments. I would like to thank Gil Segev and the anonymous referees for helpful and detailed comments.

References

[1] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001.

[2] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.

[3] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO*, pages 535–552, 2007.

[4] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *CRYPTO*, pages 360–378, 2008.

[5] 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 *ASIACRYPT*, pages 232–249, 2009.

[6] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.

[7] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO*, pages 335–359, 2008.

[8] D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.

[9] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from Decision Diffie-Hellman. In *CRYPTO*, pages 108–125, 2008.

[10] Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010. Also, Cryptology ePrint Archive, Report 2010/522.

[11] Z. Brakerski and G. Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In *CRYPTO*, pages 543–560, 2011.

[12] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO*, pages 126–144, 2003.

[13] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, pages 13–25, 1998.

[14] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002. Also, Cryptology ePrint Archive, Report 2001/085.

[15] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.

[16] Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

[17] Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.

[18] D. M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More constructions of lossy and correlation-secure trapdoor functions. In *PKC*, pages 279–295, 2010. Also, Cryptology ePrint Archive, Report 2009/590.

[19] B. Fuller, A. O'Neill, and L. Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In *TCC*, pages 582–599, 2012. Also Cryptology ePrint Archive, Report 2012/005.

[20] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. *ACM Trans. Inf. Syst. Secur.*, 9(2): 181–234, 2006.

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

[22] S. Halevi and Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. Cryptology ePrint Archive, Report 2007/118, 2007. Preliminary version in EUROCRYPT 2005.

[23] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[24] B. Hemenway and R. Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. Electronic Colloquium on Computational Complexity (ECCC), 2009.

[25] B. Hemenway, B. Libert, R. Ostrovsky, and D. Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *ASIACRYPT*, pages 70–88, 2011. also Cryptology ePrint Archive, Report 2009/088.

[26] J. Katz, R. Ostrovsky, and M. Yung. Efficient and secure authenticated key exchange using weak passwords. *J. ACM*, 57(1), 2009.
[27] E. Kiltz, A. O'Neill, and A. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In *CRYPTO*, pages 295–313, 2010.
[28] I. Mironov, O. Pandey, O. Reingold, and G. Segev. Incremental deterministic public-key encryption. In *EUROCRYPT*, pages 628–644, 2012.
[29] P. Mol and S. Yilek. Chosen-ciphertext security from slightly lossy trapdoor functions. In *PKC*, pages 296–311, 2010.
[30] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
[31] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
[32] R. Nishimaki, E. Fujisaki, and K. Tanaka. Efficient non-interactive universally composable string-commitment schemes. In *ProvSec*, pages 3–18, 2009.
[33] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
[34] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.
[35] L. Trevisan. Extractors and pseudorandom generators. *JACM*, 48(4):860–879, 2001.