Incremental Deterministic Public-Key Encryption

Ilya Mironov[1], Omkant Pandey[2], Omer Reingold[1], and Gil Segev[1]

[1] Microsoft Research Silicon Valley, Mountain View, CA 94043, USA.
{mironov,gil.segev,omer.reingold}@microsoft.com
[2] Microsoft, Redmond (USA) and Microsoft Research, Bangalore (India).
omkantp@microsoft.com

Abstract. Motivated by applications in large storage systems, we initiate the study of incremental deterministic public-key encryption. Deterministic public-key encryption, introduced by Bellare, Boldyreva, and O'Neill (CRYPTO '07), provides a realistic alternative to randomized public-key encryption in various scenarios where the latter exhibits inherent drawbacks. A deterministic encryption algorithm, however, cannot satisfy any meaningful notion of security for low-entropy plaintexts distributions, and Bellare et al. demonstrated that a strong notion of security can in fact be realized for relatively high-entropy plaintext distributions.

In order to achieve a meaningful level of security, a deterministic encryption algorithm should be typically used for encrypting rather long plaintexts for ensuring a sufficient amount of entropy. This requirement may be at odds with efficiency constraints, such as communication complexity and computation complexity in the presence of small updates. Thus, a highly desirable property of deterministic encryption algorithms is incrementality: small changes in the plaintext translate into small changes in the corresponding ciphertext.

We present a framework for modeling the incrementality of deterministic public-key encryption. Within our framework we propose two schemes, which we prove to enjoy an optimal tradeoff between their security and incrementality up to small polylogarithmic factors. Our first scheme is a generic method which can be based on any deterministic public-key encryption scheme, and in particular, can be instantiated with any semantically-secure (randomized) public-key encryption scheme in the random oracle model. Our second scheme is based on the Decisional Diffie-Hellman assumption in the standard model.

The approach underpinning our schemes is inspired by the fundamental "sample-then-extract" technique due to Nisan and Zuckerman (JCSS '96) and refined by Vadhan (J. Cryptology '04), and by the closely related notion of "locally-computable extractors" due to Vadhan. Most notably, whereas Vadhan used such extractors to construct *private-key* encryption schemes in the bounded-storage model, we show that techniques along these lines can also be used to construct incremental *public-key* encryption schemes.

1 Introduction

The fundamental notion of *semantic security* for public-key encryption schemes was introduced by Goldwasser and Micali [19]. While semantic security provides strong privacy guarantees, it inherently requires a *randomized* encryption algorithm. Unfortunately, randomized encryption breaks several assumptions of large storage systems that are crucial in efficient implementation of search (and, more generally, of indexing) and de-duplication [9, 23]. Further, randomized encryption necessarily expands the length of the plaintext, which may be undesirable in some applications, such as legacy code or in-place encryption.

Deterministic encryption. To deal with these and other drawbacks, Bellare, Boldyreva, and O'Neill [2] initiated the study of deterministic public-key encryption schemes. These are public-key encryption schemes where the encryption algorithm is deterministic. Bellare et al. formulate meaningful, and essentially "best possible", security requirements for such schemes which are inspired by and very close to semantic security. Clearly, in this setting, no meaningful notion of security can be achieved if the space of plaintexts is small. Therefore, Bellare et al. [2] required security to hold only when the plaintexts are drawn from a high min-entropy distribution.

Deterministic encryption already alleviates many of the above mentioned problems when dealing with large data volumes. For example, since the encryption algorithm is deterministic, we can now do indexing and perform fast search on encrypted data. Further, schemes that have length-preserving ciphertexts are possible as well [2]. Also, unlike randomized encryption, there is no fundamental reason that precludes noticeable savings in storage by using de-duplication techniques (which can be as large as 97% [27]); although one may not get the same amount of savings as with usual plaintext.

We emphasize that security of deterministic encryption is contingent on a very strong assumption about the underlying data distribution, namely that the plaintext has high min-entropy from the adversary's point of view. One possibility for improving security margin is to encrypt longer plaintexts whenever possible, for example, by not cutting files into smaller pieces or using larger blocks for in-place encryption. If, however, changing the plaintext requires recomputation of the ciphertext, doing that for any update may quickly negate all efficiency gains from using deterministic encryption. For a remedy we turn to *incremental cryptography*, explained below.

Incremental cryptography. Given that we are dealing with large plaintexts, computing the ciphertext from scratch for the modified plaintext can be quite an expensive operation. One such example is maintaining an (encrypted) daily backup of your hard-disk on an untrusted server. The disk may contain gigabytes of data, most of which is likely to remain unchanged between two successive backups. The problem is further intensified in various client-server settings where *all* of previous plaintext might not be available when the modification request is made. In such settings where plaintext is really large, downloading old data

can be a serious problem. This issue is clearly not specific to (deterministic) encryption, and is of very general interest.

To address this issue, Bellare, Goldreich and Goldwasser [5] introduced and developed the notion of *incremental* cryptography, first in application to digital signatures. The idea is that, once we have signed a document M, signing new versions of M should be rather quick. For example, if we only flip a single bit of M, we should be able to update the signature in time polynomial in $\log |M|$ (instead of $|M|$) and the security parameter λ. Clearly, incrementality is an attractive feature to have for any cryptographic primitive such as encryption, signatures, hash functions, and so on [6, 20, 15, 7, 11].

It is clear from our discussion that when dealing with deterministic encryption over large databases, where we are forced to encrypt rather long plaintexts for ensuring their min-entropy, what we really need is an *incremental* encryption scheme. That is, the scheme should allow quickly updating the ciphertexts to reflect small changes. In light of the observation that deterministic encryption is most desirable when dealing with large data volumes, perhaps it is not exaggerating to suggest that incrementality should be an important *design goal* for deterministic encryption rather than merely a "nice to have" feature.

1.1 Our Contributions

In this work we formalize the notion of *incremental* deterministic public-key encryption. We view incrementality and security as two orthogonal objectives, which together have a great potential in improving the deployment of deterministic encryption schemes with provable security properties in real-world applications.

Modeling incremental updates. Intuitively, a deterministic public-key encryption scheme is *incremental* if any small modification of a plaintext m resulting in a plaintext m' can be efficiently carried over for updating the encryption $c = \mathsf{Enc}_{pk}(m)$ of m to the encryption $c' = \mathsf{Enc}_{pk}(m')$ of m'. For capturing the efficiency of such an update operation we consider two natural complexity measures: (1) *input locality* (i.e., the number of ciphertexts bits that are affected when flipping a single plaintext bit), and (2) *query complexity* (i.e., the number of public-key, plaintext, and ciphertext bits that have to be read in order to update the ciphertext).

We note that modeling updates for deterministic public-key encryption is slightly different than for other primitives. For example, suppose that we allow "replacements" as considered by [5]. These are queries of the form (j, b) that replace the j-th bit of a given plaintext m by $b \in \{0, 1\}$. Then, if there exists a public algorithm Update for updating the ciphertext, then one can recover the entire plaintext from the ciphertext[3]. Therefore, we focus on the *bit flipping*

[3] The encryption algorithm is deterministic, and hence the ciphertext for every message is unique. The operation $\mathsf{Update}(j, 0)$ changes the ciphertext if and only if the jth bit of m is 1.

operation instead. This operation is specified by an index j, and sets the current value of $m[j]$ to $\neg m[j]$.

For capturing the above measures of efficiency we model the update operation as a probabilistic polynomial-time algorithm Update that receives as input the index i^* of a plaintext bit to be flipped, and has oracle access to the individual bits of the public key pk, the plaintext m to be modified, and to its encryption $c = \mathsf{Enc}_{pk}(m)$. That is, the algorithm Update can submit queries of the form (pk, i), (m, i) or (c, i), which are answered with the ith bit of pk, m, or c, respectively. We refer the reader to Section 3 for the formal description of our model, which considers also update in a "private" fashion in which the update algorithm can access the secret key but not the plaintext.

Locality lower bound. An important insight is that deterministic encryption cannot have very small incrementality. Deterministic encryption schemes require high min-entropy messages to provide any meaningful guarantee, and we show that any scheme with low incrementality can be secure only for messages with much higher entropy. Specifically, we show that for every deterministic public-key encryption scheme that satisfies the minimal notion of PRIV1-IND security for plaintext distributions of min-entropy k, plaintext length n, and ciphertext length t, the incrementality Δ of the scheme must satisfy: $\Delta \geq \frac{n-3}{k \log t}$.

Ignoring the lower-order $\log t$ factor, our proof shows in particular that the input locality of the encryption algorithm must be roughly n/k. This should be compared with the case of randomized encryption, where flipping a single plaintext bit may require to flip only a single ciphertext bit. Indeed, consider encrypting a plaintext m as the pair $(\mathsf{Enc}_{pk}(r), r \oplus m)$ for a randomly chosen mask r. Flipping a single bit of m requires flipping only a single bit of the ciphertext.

Constructions with optimal incrementality. We construct two deterministic public-key encryption schemes with optimal incrementality (up to lower-order polylogarithmic factors). Our first construction is a general transformation from any deterministic encryption scheme to an incremental one. Following the terminology developed in [2, 4, 8], the resulting scheme from this approach is PRIV1-IND secure if the underlying scheme is PRIV-IND secure. As a result, using the construction of Bellare et al. [2] in the random oracle model, we can instantiate our approach in the random oracle model based on any semantically-secure (randomized) public-key encryption scheme, and obtain a deterministic scheme with optimal incrementality.

Our second, more direct construction, avoids the random oracle model. It is based on the Decisional Diffie-Hellman assumption in the standard model, and enjoys optimal incrementality. The scheme relies on the notion of *smooth trapdoor functions* that we introduce (and was implicitly used by Boldyreva et al. [8]), and realize it in an incremental manner based on the Decisional Diffie-Hellman assumption. Both of our constructions guarantee PRIV1-IND security when encrypting n-bit plaintexts with min-entropy $k \geq n^{\epsilon}$, where $\epsilon > 0$ is any pre-specified constant.

1.2 Related Work

The problem of composing public-key encryption and de-duplication was addressed by Doucer et al. [14] via the concept of *convergent encryption*, in which files are encrypted using their own hash values as keys. Security of the scheme is argued in the random-oracle model and under implicit assumption of the plaintext's high min-entropy. The formal goal of leveraging entropy of the source to achieve information-theoretic security with a short symmetric key was articulated by Russell and Wang [24], followed by Dodis and Smith [13].

The notion of public-key deterministic encryption was introduced by Bellare, Boldyreva, and O'Neill [2], and then further studied by Bellare, Fischlin, O'Neill, and Ristenpart [4], Boldyreva, Fehr, and O'Neill [8], Brakerski and Segev [10], Wee [26], and Fuller, O'Neill and Reyzin [18]. Bellare et al. [2] proved their constructions in the random oracle model; subsequent papers demonstrated schemes secure in the standard model based on trapdoor permutations [4] and lossy trapdoor functions [8]. Brakerski and Segev [10] and Wee [26] address the question of security of public-key deterministic encryption in the presence of auxiliary input. Fuller et al. [18] presented a construction based on any trapdoor function that admits a large number of simultaneous hardcore bits, and a construction that is secure for a bounded number of possibly related plaintexts.

Constructions of deterministic public-key encryption found an intriguing application in "hedged" public-key encryptions [3]. These schemes remain secure even if the randomness used during the encryption process is not perfect (controlled by or leaked to the adversary) as long as the joint distribution of plaintext-randomness has sufficient min-entropy.

The concept of incremental cryptography started with the work of Bellare, Goldreich, and Goldwasser [5], who considered the case of hashing and signing. They also provided discrete-logarithm based constructions for incremental collision-resistant hash and signatures, that support block *replacement* operation. Constructions supporting block *insertion* and *deletion* were first developed in [6], with further refinements and new issues concerning incrementality such as tamper-proof updates, privacy of updates, and incrementality in symmetric encryption. In subsequent work, Fischlin presented an incremental signature schemes supporting insertion/deletion of blocks, and tamper-proof updates [15], and proved a $\Omega(\sqrt{n})$ lower bound on the signature size of schemes that support substitution and replacement operations (the bound can be improved to $\Omega(n)$ in certain special cases) [16]. Bellare and Micciancio [7] revisited the case of hashing, and provided new constructions for the same based on discrete logarithms and lattices. Buonanno, Katz, and Yung [11] considered the issue of incrementality in symmetric unforgeable encryption and suggested three modes of operations for AES achieving this notion.

The goal of incremental cryptography, i.e., *input locality*, can be contrasted with the dual question of placing cryptography in the NC^0 complexity class, i.e., identifying cryptographic primitives with constant *output locality*. This problem has essentially been resolved for public-key encryption in the positive by Applebaum, Ishai, and Kushilevitz [1], who construct schemes based on standard

number-theoretic assumptions and lattice problems where each bit of the encryption operation depends on at most four bits of the input. Applebaum et al. also argue impossibility of semantically-secure public-key encryption scheme with constant input locality [1, Section C.1].

1.3 Overview of Our Approach

In this section we present a high-level overview of our two constructions. First, we describe the well-known "sample-then-extract" approach [21, 25] that serves as our inspiration for constructing incremental schemes. Then, we describe the main ideas underlying our schemes, each of which is based on a different realization of the "sample-then-extract" approach.

"Sample-then-extract". A fundamental fact in the theory of pseudorandomness is that a random sample of bits from a string of high min-entropy essentially preserves the min-entropy rate. This was initially proved by Nisan and Zuckerman [21] and then refined by Vadhan [25] that captured the optimal parameters. Intuitively, the "sample-then-extract" lemma states that if $\mathcal{X} \in \{0,1\}^n$ has min-entropy rate δ, and $\mathcal{X}_S \in \{0,1\}^t$ is the projection of $\mathcal{X}$ onto a random set $S \subseteq [n]$ of t positions, then $\mathcal{X}_S$ is statistically-close to a source with min-entropy rate $\delta' = \Omega(\delta)$.

This lemma serves as a fundamental tool in the design of randomness extractors. Moreover, in the cryptographic setting, it was used by Vadhan [25] to construct *locally-computable extractors*, which allow to compute their output by examining a small number of input bits. Such extractors were used by Vadhan to design private-key encryption schemes in the bounded-storage model. In this work we demonstrate for the first time that the "sample-then-extract" approach can be leveraged to design not only *private-key* encryption schemes, but also *public-key* encryption schemes.

A generic construction via random partitioning. In the setting of randomized encryption, a promising approach for ensuring incrementality is to divide each plaintext m into consecutive and rather small blocks $m = m_1||\cdots||m_\ell$, and to separately encrypt each block m_i. Thus, changing a single bit of m affects only a single block of the ciphertext. Moreover, the notion of semantic security is sufficiently powerful to even allow each block m_i to be as small as a single bit. In the setting of deterministic encryption, however, security can hold only when each encrypted block has a sufficient amount of min-entropy. At this point we note that even if a plaintext $m = m_1||\cdots||m_\ell$ has high min-entropy, it may clearly be the case that some of its small blocks have very low min-entropy (or even fixed). Thus, this approach seems to fail for deterministic encryption.

As an alternative, however, we propose the following approach: instead of dividing the plaintext m into fixed blocks, we project it onto a *uniformly chosen partition* $S_1, \ldots, S_\ell$ of the plaintext positions to sets of equal sizes, and then separately encrypt each of the projections $m_{S_1}, \ldots, m_{S_\ell}$ using an underlying

(possibly non-incremental) deterministic encryption scheme[4]. By the fact that we use a partition of the plaintext positions we ensure on the one hand that the plaintext m can be fully recovered, and on the other that each plaintext position appears in only one set (and thus the scheme is incremental). In terms of security, since we use a uniformly chosen partition, the distribution of each individual set S_i is uniform, and therefore by carefully choosing the size of the sets the "sample-and-extract" lemma guarantees that with overwhelming probability each projection m_{S_i} preserves the min-entropy rate of m. Therefore, the scheme is secure as long as the underlying scheme guarantees PRIV-IND security (see Section 2.2 for the notions of security for deterministic encryption).

By instantiating this approach with the constructions of Bellare et al. [2] in the random oracle model, we obtain as a corollary a deterministic public-key encryption scheme with optimal incrementality based either on any semantically-secure (randomized) public-key encryption scheme, or on RSA-OAEP which yields a *length-preserving* incremental scheme.

A construction based on smooth trapdoor functions. Although our first construction is a rather generic one, constructions of PRIV-IND-secure schemes are known only in the random oracle model. In the standard model, Boldyreva et al. [8] introduced the slightly weaker notion of PRIV1-IND security, which considers plaintexts that have high min-entropy even when conditioned on other plaintexts, and showed that it can be realized by composing any lossy trapdoor function with a pairwise independent permutation. This approach, however, does not seem useful for constructing incremental schemes, since pairwise independence is inherently non-incremental. A simple observation, however, shows that the approach of Boldyreva et al. [8] requires in fact trapdoor functions with weaker properties, that we refer to as *smooth trapdoor functions* (this is implicit in [8]).

Informally, a collection of smooth 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 "smooth" in the sense that their output distribution on any source of input with high min-entropy is statistically close to their output distribution on a uniformly sampled input. The only security 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 smooth functions. We show that any collection of smooth trapdoor functions is a PRIV1-IND-secure deterministic encryption scheme (again, this is implicit in [8]).

Next, we construct a collection of *incremental* smooth trapdoor functions based on the Decisional Diffie-Hellman (DDH) assumption, by significantly refining the DDH-based lossy trapdoor functions of Freeman et al. [17] (which in turned generalized those of Peikert and Waters [22]). Our collection is parameterized by a group G of prime order p that is generated by an element $g \in G$. A

[4] A minor technical detail is that we would also like to ensure that we always encrypt distinct values, and therefore we concatenate the block number i to each projection m_{S_i}.

public key is of the form g^A, where $A \in \mathbb{Z}^{n \times n}$ is sampled from one distribution for injective keys, and from a different distribution for smooth keys[5]. Evaluating a function on an input $x \in \{0,1\}^n$ is done by computing $g^{Ax} \in G^n$ and inversion for injective keys is done using the secret key A^{-1}.

The key point in our scheme is the distribution of the matrix A for injective and smooth keys. For smooth keys the matrix A is generated to satisfy two properties. The first is that each of its first ℓ rows has t randomly chosen entries with values that are chosen uniformly from $\mathbb{Z}_p$, and all other $n-t$ entries are zeros (where ℓ and t are carefully chosen depending on the min-entropy rate). Looking ahead, when computing the inner product of such a sparse row with a source of min-entropy larger than $\log p$, the "sample-then-extract" lemma guarantees that the output is statistically close to uniform. In a sense, this is a realization of a locally-computable extractor that is embedded in our functions. The second property, is that each of its last $n - \ell$ rows are linear combinations of the first ℓ rows, and therefore the image of its corresponding linear map is determined by the first ℓ rows. This way, we can argue that smooth keys hide essentially all information on the underlying input distribution.

For injective keys, we sample a matrix A from the distribution of smooth keys, and then re-sample all its non-zero entries with independently and uniformly distributed elements of $\mathbb{Z}_p$. A subtle complication arises since such a matrix is not necessarily invertible, as required for injective keys, but this is easily resolved (without hurting the smooth keys – see Section 5 for more details). Observing that for injective keys each column of A contains roughly t non-zero entries, this yields a PRIV1-IND-secure scheme with optimal incrementality.

Paper organization. In Section 2 we introduce the notation and tools that are used in this paper. In Section 3 we present a framework for modeling the incrementality of deterministic public-key encryption schemes. In Section 4 we present our generic construction, and in Section 5 we present our DDH-based construction. Due to space limitations we refer the reader to the full version for the proof of the lower bound.

2 Preliminaries

2.1 Probability Distributions

For a distribution $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x according to $\mathcal{X}$. Similarly, for a set Ω we denote by $\omega \leftarrow \Omega$ the process of sampling a value ω from the uniform distribution over Ω. If $\mathcal{X}$ is a distribution and f is a function defined over its support, then $f(\mathcal{X})$ denotes the outcome of the experiment where $f(x)$ is evaluated on x sampled from $\mathcal{X}$. For any $n \in \mathbb{N}$ we denote by $\mathcal{U}_n$ the uniform distribution over the set $\{0,1\}^n$.

The *min-entropy* of a distribution $\mathcal{X}$ that is defined over a set Ω is defined as $\mathrm{H}_\infty(\mathcal{X}) = \min_{\omega \in \Omega} \log\left(1/\Pr[\mathcal{X} = \omega]\right)$. A *k-source* is distribution $\mathcal{X}$ with

[5] For any matrix $A = \{a_{ij}\}_{i \in [n], j \in [n]} \in \mathbb{Z}_p^{n \times n}$ we denote by $g^A \in G^{n \times n}$ the matrix $\{g^{a_{ij}}\}_{i \in [n], j \in [n]}$.

$\mathrm{H}_\infty(\mathcal{X}) \geq k$, and the *min-entropy rate* of a k-source over the set $\{0,1\}^n$ is k/n. The *statistical distance* between two distributions $\mathcal{X}$ and $\mathcal{Y}$ over a set Ω is defined as $\mathrm{SD}(\mathcal{X},\mathcal{Y}) = \max_{S\subseteq\Omega} |\Pr[\mathcal{X} \in S] - \Pr[\mathcal{Y} \in S]|$. A distribution $\mathcal{X}$ is ϵ-*close* to a k-source if there exists a k-source $\mathcal{Y}$ such that $\mathrm{SD}(\mathcal{X},\mathcal{Y}) \leq \epsilon$. The following standard lemma (see, for example, [12]) essentially states that revealing r bits of information on a random variable may reduce its min-entropy by roughly r.

Lemma 2.1. *Let $\mathcal{Z}$ be a distribution over at most 2^r values, then for any distribution $\mathcal{X}$ and for any $\epsilon > 0$ it holds that*

$$\Pr_{z\leftarrow\mathcal{Z}}\left[\mathrm{H}_\infty(\mathcal{X}|\mathcal{Z}=z) \geq \mathrm{H}_\infty(\mathcal{X}) - r - \log(1/\epsilon)\right] \geq 1-\epsilon \ .$$

We say that two families of distributions $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ are *statistically close*, denoted by $\mathcal{X} \approx \mathcal{Y}$, if there exists a negligible function $\nu(\lambda)$ such that $\mathrm{SD}(\mathcal{X},\mathcal{Y}) \leq \nu(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$. Two families of distributions $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ are *computationally indistinguishable*, denoted by $\mathcal{X} \overset{\mathrm{c}}{\approx} \mathcal{Y}$, if for any probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\lambda)$ such that

$$\left|\Pr_{x\leftarrow\mathcal{X}_\lambda}\left[A(1^\lambda,x)=1\right] - \Pr_{y\leftarrow\mathcal{Y}_\lambda}\left[A(1^\lambda,y)=1\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

The "sample-then-extract" lemma. The following lemma due to Vadhan [25] plays a major role in our constructions. This is a refinement of the fundamental "sample-then-extract" lemma that was originally proved by Nisan and Zuckerman [21], stating that a random of sample of bits from a string essentially preserves its min-entropy rate. Vadhan's refinement shows that the min-entropy rate is in fact preserved up to an arbitrarily small additive loss, whereas the original lemma loses a logarithmic factor. Intuitively, the lemma states that if $\mathcal{X} \in \{0,1\}^n$ is a δn-source, and $\mathcal{X}_S \in \{0,1\}^t$ is the projection of $\mathcal{X}$ onto a random set $S \subseteq [n]$ of t positions, then, with high probability, $\mathcal{X}_S$ is statistically-close to a $\delta' t$-source, where $\delta' = \Omega(\delta)$. Whereas Nisan and Zuckerman [21] and Vadhan [25] were concerned with the amount of randomness that is required for sampling the t positions, in our case we can allow ourselves to sample the set S uniformly at random, and this leads to the following simplified form of the lemma:

Lemma 2.2 ([25] – simplified). *Let $\mathcal{X}$ be a δn-source over $\{0,1\}^n$, let $t \in [n]$, and let $\mathcal{S}$ denote the uniform distribution over sets $S \subseteq [n]$ of size t. Then, there exists a distribution $\mathcal{W}$ over $\{0,1\}^t$, jointly distributed with $\mathcal{S}$, such that the following hold:*

1. *$(\mathcal{S},\mathcal{X}_\mathcal{S})$ is $2^{-\Omega(\delta t/\log(1/\delta))}$-close to $(\mathcal{S},\mathcal{W})$.*
2. *For any set $S \subseteq [n]$ of size t it holds that $\mathcal{W}|_{\mathcal{S}=S}$ is a $\delta' t$-source for $\delta' = \delta/4$.*

2.2 Deterministic Public-Key Encryption

A deterministic public-key encryption scheme is almost identical to a (randomized) public-key encryption scheme, where the only difference is that the encryption algorithm is deterministic. More specifically, a deterministic public-key

encryption scheme is a triple of polynomial-time algorithms $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. The key-generation algorithm KG is a randomized algorithm which takes as input the security parameter 1^λ, where $\lambda \in \mathbb{N}$, and outputs a pair (pk, sk) of a public key pk and a secret key sk. The encryption algorithm Enc takes as input the security parameter 1^λ, a public key pk, and a plaintext $m \in \{0,1\}^{n(\lambda)}$, and outputs a ciphertext $c \in \{0,1\}^{t(\lambda)}$. The (possibly deterministic) decryption algorithm Dec takes as input the security parameter 1^λ, a secret key sk, and a ciphertext $c \in \{0,1\}^{t(\lambda)}$, and outputs either a plaintext $m \in \{0,1\}^{n(\lambda)}$ or the special symbol $\perp$. For succinctness, we will always assume 1^λ as an implicit input to all algorithms and refrain from explicitly specifying it.

In terms of security, in this paper we follow the standard approach for formalizing the security of deterministic public-key encryption schemes introduced by Bellare, Boldyreva and O'Neill [2] and further studied by Bellare, Fischlin, O'Neill and Ristenpart [4] and by Boldyreva, Fehr and O'Neill [8]. Specifically, we consider the PRIV-IND notion of security asking that any efficient algorithm has only a negligible advantage in distinguishing between encryptions of different sequences of plaintexts as long as each plaintext is sampled from high-entropy sources. We also consider the PRIV1-IND notion of security that focuses on a single plaintext, and asks that any efficient algorithm has only a negligible advantage in distinguishing between encryptions of different plaintexts that are sampled from high-entropy sources. This notion of security was shown by Boldyreva, Fehr and O'Neill [8] to guarantee security for block-sources of messages (that is, for sequences of messages where each message has high-entropy even when conditioned on the previous messages).

For defining these notions of security we rely on the following notation. We denote by $\mathbf{m} = (m_1, \ldots, m_\ell)$ a sequence of plaintexts, and by $\mathbf{c} = \mathsf{Enc}_{pk}(\mathbf{m})$ the sequence of their encryptions $(\mathsf{Enc}_{pk}(m_1), \ldots, \mathsf{Enc}_{pk}(m_\ell))$ under a public key pk.

Definition 2.3 (k-source ℓ-message adversary). *Let $A = (A_1, A_2)$ be a probabilistic polynomial-time algorithm, and let $k = k(\lambda)$ and $\ell = \ell(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. For any $\lambda \in \mathbb{N}$ denote by $(\mathcal{M}_\lambda^{(0)}, \mathcal{M}_\lambda^{(1)}, \mathcal{STATE}_\lambda)$ the distribution corresponding to the output of $A_1(1^\lambda)$. Then, A is a* k-source ℓ-message adversary *if the following properties hold:*

1. *$\mathcal{M}_\lambda^{(b)} = \left(\mathcal{M}_{1,\lambda}^{(b)}, \ldots, \mathcal{M}_{\ell,\lambda}^{(b)}\right)$ is a distribution over sequences of ℓ plaintexts for each $b \in \{0,1\}$.*
2. *For any $\lambda \in \mathbb{N}$, $i, j \in [\ell]$, and $\left(\left(m_1^{(0)}, \ldots, m_\ell^{(0)}\right), \left(m_1^{(1)}, \ldots, m_\ell^{(1)}\right), \mathsf{state}\right)$ that is produced by $A_1(1^\lambda)$ it holds that $m_i^{(0)} = m_j^{(0)}$ if and only if $m_i^{(1)} = m_j^{(1)}$.*
3. *For any $\lambda \in \mathbb{N}$, $b \in \{0,1\}$, $i \in [\ell]$, and $\mathsf{state} \in \{0,1\}^*$ it holds that $\mathcal{M}_{i,\lambda}^{(b)}|_{\mathcal{STATE}_\lambda = \mathsf{state}}$ is a $k(\lambda)$-source.*

Definition 2.4 (PRIV-IND). *A deterministic public-key encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is* PRIV-IND-secure for $k(\lambda)$-source $\ell(\lambda)$-message adver-

saries *if for any probabilistic polynomial-time $k(\lambda)$-source $\ell(\lambda)$-message adversary $A = (A_1, A_2)$ there exists a negligible function $\nu(\lambda)$ such that*

$$\mathsf{Adv}^{\mathsf{PRIV-IND}}_{\Pi,A,\lambda} \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{PRIV-IND}}_{\Pi,A,\lambda}(0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{PRIV-IND}}_{\Pi,A,\lambda}(1) = 1\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where $\mathsf{Expt}^{\mathsf{PRIV-IND}}_{\Pi,A,\lambda}(b)$ is defined as follows:

1. $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$.
2. $(\mathbf{m}_0, \mathbf{m}_1, \mathsf{state}) \leftarrow A_1(1^\lambda)$.
3. $\mathbf{c} \leftarrow \mathsf{Enc}_{pk}(\mathbf{m}_b)$.
4. *Output* $A_2(1^\lambda, pk, \mathbf{c}, \mathsf{state})$.

Definition 2.5 (PRIV1-IND). *A deterministic public-key encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is* PRIV1-IND-secure for $k(\lambda)$-source adversaries *if for any probabilistic polynomial-time $k(\lambda)$-source 1-message adversary $A = (A_1, A_2)$ there exists a negligible function $\nu(\lambda)$ such that*

$$\mathsf{Adv}^{\mathsf{PRIV1-IND}}_{\Pi,A,\lambda} \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{PRIV1-IND}}_{\Pi,A,\lambda}(0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{PRIV1-IND}}_{\Pi,A,\lambda}(1) = 1\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where $\mathsf{Expt}^{\mathsf{PRIV1-IND}}_{\Pi,A,\lambda}(b)$ is defined as follows:

1. $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$.
2. $(m_0, m_1, \mathsf{state}) \leftarrow A_1(1^\lambda)$.
3. $c \leftarrow \mathsf{Enc}_{pk}(m_b)$.
4. *Output* $A_2(1^\lambda, pk, c, \mathsf{state})$.

3 Modeling Incremental Deterministic Public-Key Encryption

In this section we present a framework for modeling the incrementality of deterministic public-key encryption schemes. Intuitively, a deterministic public-key encryption scheme is *incremental* if any small modification of a plaintext m resulting in a plaintext m' can be efficiently carried over for updating the encryption $c = \mathsf{Enc}_{pk}(m)$ of m to the encryption $c' = \mathsf{Enc}_{pk}(m')$ of m'. For capturing the efficiency of such an update operation we consider two natural complexity measures[6]:

- Input locality: The number of ciphertexts bits that are affected when flipping a single plaintext bit.
- Query complexity: The number of public-key, plaintext, and ciphertext bits that have to be read in order to update the ciphertext when flipping a single plaintext bit.

[6] For simplicity we focus on the case where both plaintexts and ciphertexts are represented as bit strings. We note, however, that our approach easily generalizes to arbitrary message and ciphertext spaces.

For capturing the above measures of efficiency we model the update operation as a probabilistic polynomial-time algorithm Update that receives as input the index i^* of a plaintext bit to be flipped, and has oracle access to the individual bits of the public key pk, the plaintext m to be modified, and to its encryption $c = \mathsf{Enc}_{pk}(m)$. That is, the algorithm Update can submit queries of the form (pk, i), (m, i) or (c, i), which are answered with the ith bit of pk, m, or c, respectively.

More formally, let $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a deterministic public-key encryption scheme with message space $\{0,1\}^n$ and ciphertext space $\{0,1\}^t$ (where $n = n(\lambda)$ and $t = t(\lambda)$ are functions of the security parameter $\lambda \in \mathbb{N}$), and let Update be its corresponding update algorithm. We denote by $S \leftarrow \mathsf{Update}^{pk,m,c}(1^\lambda, i^*)$ the process in which the update algorithm with input $i^* \in [n]$ and oracle access to the individual bits of the public key pk, the plaintext m to be modified, and to its encryption $c = \mathsf{Enc}_{pk}(m)$, outputs a set $S \subseteq [t]$ of positions indicating which bits of the ciphertext c have to be flipped.

Definition 3.1 (Incremental deterministic PKE). *Let $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a deterministic public-key encryption scheme with message space $\{0,1\}^n$ and ciphertext space $\{0,1\}^t$, where $n = n(\lambda)$ and $t = t(\lambda)$ are functions of the security parameter $\lambda \in \mathbb{N}$. The scheme Π is* $\Delta(\lambda)$-incremental *is there exists a probabilistic polynomial-time algorithm* Update *satisfying the following requirements:*

1. *Correctness: There exists a negligible function $\nu(\lambda)$ such that for all sufficiently large $\lambda \in \mathbb{N}$, for any plaintext $m \in \{0,1\}^n$ and for any index $i^* \in [n]$ it holds that*

$$\Pr\left[c' = \mathsf{Enc}_{pk}(m') \;\middle|\; \begin{array}{c} c = \mathsf{Enc}_{pk}(m),\ S \leftarrow \mathsf{Update}^{pk,m,c}(1^\lambda, i^*) \\ m'[i^*] = \neg m[i^*] \\ m'[i] = m[i] \text{ for all } i \in [n] \setminus \{i^*\} \\ c'[j] = \neg c[j] \text{ for all } j \in S \\ c'[j] = c[j] \text{ for all } j \in [t] \setminus S \end{array} \right] \geq 1 - \nu(\lambda),$$

   where the probability is taken over the internal coin tosses of KG *and* Update.
2. *Efficiency: For all sufficiently large $\lambda \in \mathbb{N}$ the algorithm $\mathsf{Update}^{(\cdot)}(1^\lambda, \cdot)$ issues at most $\Delta(\lambda)$ oracle queries and outputs sets of size at most $\Delta(\lambda)$.*

Access to the plaintext. When providing the update algorithm with oracle access to the bits of the plaintext $m \in \{0,1\}^n$ we can assume without loss of generality that the only update operations are to flip the ith bit of m for $i \in [n]$. That is, one can also consider the operation of setting the ith bit of m to 0 or 1, but this can be handled by first querying the ith bit of m and then flipping it if it is different than the required value. We note, however, that for supporting only flipping operations it is not clear that access to the plaintext must be provided.

An important observation is that when access to the plaintext is not provided (i.e., when the update algorithm can query only the public key and the ciphertext), it is impossible to support the operation of setting a bit to 0 and 1 while providing PRIV1-IND security. That is, any such update algorithm can be

used to attack the PRIV1-IND security of the scheme by distinguishing between encryptions of high-entropy messages (and this holds for any level of incrementality)[7].

Privately-incremental schemes. In various scenarios it may be natural to provide the update algorithm with access not to the plaintext m but rather to the secret key sk (and thus indirect access to the plaintext which may be less efficient in terms of query complexity). Consider for example, a scenario in which a client stores an encrypted version $\bar{F}$ of a file F on a remote and untrusted server. In this the client does not have direct access to the file F, but only indirect access by using its secret key to recover parts of the file. In such a scenario it is required to capture the efficiency of the client by considering its query complexity to the secret key (and ciphertext) and not to the plaintext. This leads to a natural variant of Definition 3.1 in which the update algorithm is given oracle access to the public key pk, the secret key sk, and the ciphertext c (but no direct access to the plaintext).

4 A Generic Construction via Random Partitioning

In this section we present a generic construction of an incremental PRIV1-IND-secure deterministic public-key encryption scheme from any PRIV-IND-secure deterministic public-key encryption scheme. As discussed in Section 1.3 our approach is a "randomized" alternative to the commonly-used approach of dividing the plaintext into small blocks and encrypting each block. Instead of dividing an n-bit plaintext m into fixed blocks, we project it onto a uniformly chosen partition $S_1, \ldots, S_{n/t}$ of the plaintext positions $\{1, \ldots, n\}$ to sets of size t each, and then separately encrypt each of the projections $m_{S_1}, \ldots, m_{S_{n/t}}$ using the underlying encryption scheme. Thus, when flipping a single bit of m we only need to update the encryption of the projection m_{S_i} for which the corresponding position belongs to the set S_i. Therefore, the resulting scheme enjoys the same incrementality that the underlying scheme has for small blocks. A more formal description follows.

The scheme. Let $\Pi' = (\mathsf{KG}', \mathsf{Enc}', \mathsf{Dec}')$ be a deterministic public-key encryption scheme for n'-bit plaintexts that is IND-PRIV-secure for k'-source ℓ'-message adversaries, where $n' = n'(\lambda)$, $k' = k'(\lambda)$ and $\ell' = \ell'(\lambda)$ are functions of the security parameter $\lambda \in \mathbb{N}$. We construct a deterministic public-key encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ for n-bit plaintexts that is PRIV1-IND-secure

[7] Consider the adversary $A = (A_1, A_2)$ that is defined as follows. The algorithm A_1 outputs $(m_0, m_1, \mathsf{state})$ where $m_0 \leftarrow \mathcal{U}_k || 0^{n-k}$ and $m_1 \leftarrow \mathcal{U}_n$ are sampled independently at random, and $\mathsf{state} = \perp$. That is, m_0 is a distributed uniformly conditioned on ending with 0^{n-k}, and m_1 is distributed uniformly. The algorithm A_2 on input $c = \mathsf{Enc}_{pk}(m_b)$ invokes the update algorithm to set the leftmost k bits of the plaintext corresponding to c to 0, and then compares the resulting ciphertext to $\mathsf{Enc}_{pk}(0^n)$. Note that if $b = 0$ then the two ciphertexts are always equal, and if $b = 1$ then they are equal only with probability $2^{-(n-k)}$.

for k-source adversaries, where $n = n(\lambda)$ and $k = k(\lambda)$ are functions of the security parameter $\lambda \in \mathbb{N}$ as follows:

- The algorithm KG on input the security parameter 1^λ samples $(pk', sk') \leftarrow \mathsf{KG}'(1^\lambda)$ together with a uniformly chosen partition $S_1, \ldots, S_{n/t}$ of $[n]$, where each set in the partition is of size $t = \Theta(\frac{n}{k} \cdot k')$. It then outputs $pk = (pk', S_1, \ldots, S_{n/t})$ and $sk = sk'$.[8]
- The algorithm $\mathsf{Enc}_{pk}(\cdot)$ on input a plaintext $m \in \{0,1\}^n$ outputs the ciphertext $(\mathsf{Enc}'_{pk'}(1||m_{S_1}), \ldots, \mathsf{Enc}'_{pk'}(n/t||m_{S_{n/t}}))$.
- The algorithm $\mathsf{Dec}_{sk}(\cdot)$ on input a ciphertext $(c_1, \ldots, c_{n/t})$ computes $m_{S_i} = \mathsf{Dec}'_{sk'}(c_i)$ for every $i \in [n/t]$, and outputs the plaintext m defined by the projections $m_{S_1}, \ldots, m_{S_{n/t}}$.

We establish the security and incrementality of this scheme by proving the following theorem (due to space limitations the proof appears in the full version):

Theorem 4.1. *Assuming that Π' encrypts n'-bit plaintexts, for $n' = t + \log(n/t)$, and is IND-PRIV-secure for k'-source ℓ'-message adversaries, for some $k' = \omega(\log^2 n)$ and for $\ell' = n/t$, the scheme Π is PRIV1-IND-secure for k-sources.*

5 A Construction Based on the Decisional Diffie-Hellman Assumption

In this section we construct a deterministic public-key encryption scheme that enjoys essentially optimal incrementality, and guarantees PRIV1-IND security based on the Decisional Diffie-Hellman (DDH) assumption. We begin by introducing rather standard notation and then describe the scheme.

Notation. Let G be a group of prime order p that is generated by $g \in G$. For any matrix $A = \{a_{ij}\}_{i\in[n],j\in[n]} \in \mathbb{Z}_p^{n\times n}$ we denote by $g^A \in G^{n\times n}$ the matrix $\{g^{a_{ij}}\}_{i\in[n],j\in[n]}$. In addition, for a column vector $m = (m_1, \ldots, m_n)^\mathsf{T} \in \mathbb{Z}_p^n$ and a matrix $A = \{a_{ij}\}_{i\in[n],j\in[n]} \in \mathbb{Z}_p^{n\times n}$ we define

$$A \odot g^m \stackrel{\text{def}}{=} g^A \odot m \stackrel{\text{def}}{=} g^{Am} = (g^{\sum_i a_{1,i}m_i}, \ldots, g^{\sum_i a_{n,i}m_i})^\mathsf{T} \in G^n \ .$$

The scheme. Let $\mathsf{GroupGen}$ be a probabilistic polynomial-time algorithm that takes as input the security parameter 1^λ, and outputs a triplet (G, p, g) where G is a group of prime order p that is generated by $g \in G$, and p is a λ-bit prime number. The scheme is parameterized by the security parameter λ, the message length $n = n(\lambda)$, and the min-entropy $k = k(\lambda)$ for which the scheme is secure. Both n and k are polynomials in the security parameter. The scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows:

[8] Without loss of generality we can assume that t divides n, as otherwise we can pad plaintexts with at most t zeros, and for our choice of parameters this would only have a minor effect on the min-entropy rate.

- **Key generation.** The algorithm KG on input the security parameter 1^λ samples $(G, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$, and a matrix $A \leftarrow \mathcal{A}_{n,k,p}$, where $\mathcal{A}_{n,k,p}$ is a distribution over $\mathbb{Z}_p^{n\times n}$ which is defined below. It then outputs $pk = (G, p, g, g^A)$ and $sk = A^{-1}$.
- **Encryption.** The algorithm $\mathsf{Enc}_{pk}(\cdot)$ on input a plaintext $m \in \{0,1\}^n$ outputs the ciphertext $g^A \odot m = g^{Am} \in G^n$.
- **Decryption.** The algorithm $\mathsf{Dec}_{sk}(\cdot)$ on input a ciphertext $g^c = (g^{c_1}, \ldots, g^{c_n}) \in G^n$ first computes $w = A^{-1} \odot g^c = g^{A^{-1}c} \in G^n$, and lets $w = (g^{m_1}, \ldots, g^{m_n})$. If $m = (m_1, \ldots, m_n) \in \{0,1\}^n$ (note that this test can be computed efficiently) then it outputs m, and otherwise it outputs $\perp$.

The distribution $\boldsymbol{\mathcal{A}_{n,k,p}}$. For completing the description of our scheme it remains to specify the distribution $\mathcal{A}_{n,k,p}$ that is defined over $\mathbb{Z}_p^{n\times n}$. Looking ahead this distribution will be used to define the distribution of injective keys in our collection of smooth trapdoor functions. In fact, we find it convenient to first specify the distribution $\widetilde{\mathcal{A}}_{n,k,p}$ that will be used to define the distribution of smooth keys. These two distributions rely on the following distributions as building block:

- **$\boldsymbol{\mathcal{R}_{n,k,p}}$: sparse random $\boldsymbol{\ell \times n}$ matrices.** The distribution $\mathcal{R}_{n,k,p}$ is defined as a random sample from $\mathbb{Z}_p^{\ell\times n}$ matrices that have exactly $t = \Theta(\frac{n}{k} \cdot \log^3 n)$ non-zero entries in each row, where $\ell = \Theta(k/\log p)$.
- **$\boldsymbol{\mathcal{D}_{n,k,p}}$: diagonally-striped $\boldsymbol{\ell \times n}$ matrices.** The distribution $\mathcal{D}_{n,k,p}$ is defined as a random sample from $\mathbb{Z}_p^{\ell\times n}$ matrices whose elements d_{ij} are non-zero if and only if $i \equiv j \pmod{\ell}$ (for simplicity we assume that n is divisible by ℓ).

The distribution $\widetilde{\mathcal{A}}_{n,k,p}$ over $\mathbb{Z}_p^{n\times n}$ is defined as matrices $\widetilde{A}$ obtained by independently sampling $R \leftarrow \mathcal{R}_{n,k,p}$, $D_1 \leftarrow \mathcal{D}_{n,k,p}$, and $D_2 \leftarrow \mathcal{D}_{n,k,p}$, and letting $\widetilde{A} \stackrel{\text{def}}{=} D_2^\intercal \times (R + D_1)$. Then, the distribution $\mathcal{A}_{n,k,p}$ is defined as matrices A obtained by sampling a matrix $\widetilde{A} \leftarrow \widetilde{\mathcal{A}}_{n,k,p}$ and then *re-sampling* all its non-zero entries from $\mathbb{Z}_p$ independently and uniformly at random. In other words, the resulting matrix A preserves zeroes of the matrix $\widetilde{A}$, while randomizing all other elements (and thus linear dependencies between rows) of the original matrix. See Figure 1 for an illustration of the distributions $\mathcal{R}_{n,k,p}$, $\mathcal{D}_{n,k,p}$ and $\widetilde{\mathcal{A}}_{n,k,p}$.

Intuitively, the matrix D_1 is only meant to ensure that such the resulting matrix A is invertible. Indeed, the matrix D_1 guarantees that with an overwhelming probability all the elements on the main diagonal of A are non-zeros. Now, ignoring the matrix D_1, the matrix $\widetilde{A}$ is generated to satisfy two properties. The first is that each of its first ℓ rows has t randomly chosen entries with values that are chosen uniformly from $\mathbb{Z}_p$, and all other $n - t$ entries are zeros. Looking ahead, when computing the inner product of such a row with a source of min-entropy larger than $\log p$, the "sample-then-extract" lemma (see Lemma 2.2) guarantees that the output is statistically close to uniform. The second property, is that each of its last $n - \ell$ rows are linear combinations of the first ℓ rows, and therefore the image of its corresponding linear map is determined by the first ℓ rows.

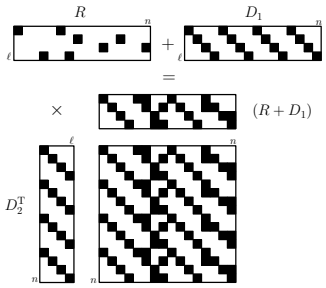

Fig. 1. The distributions $\mathcal{R}_{n,k,p}$, $\mathcal{D}_{n,k,p}$ and $\widetilde{\mathcal{A}}_{n,k,p}$.

The following theorem establishes the security of the scheme (due to space limitations the proof appears in the full version):

Theorem 5.1. *Under the Decisional Diffie-Hellman assumption the scheme Π is PRIV1-IND-secure for k-sources.*

Acknowledgements. We thank Salil Vadhan for useful discussions regarding Lemma 2.2.

References

1. B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006.
2. M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology – CRYPTO '07*, pages 535–552, 2007.
3. 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 '09*, pages 232–249, 2009.
4. M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Advances in Cryptology – CRYPTO '08*, pages 360–378, 2008.
5. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography: The case of hashing and signing. In *Advances in Cryptology – CRYPTO '94*, pages 216–233, 1994.
6. M. Bellare, O. Goldreich, and S. Goldwasser. Incremental cryptography and application to virus protection. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 45–56, 1995.
7. M. Bellare and D. Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *Advances in Cryptology – EUROCRYPT '97*, pages 163–192, 1997.
8. 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 '08*, pages 335–359, 2008.
9. W. J. Bolosky, S. Corbin, D. Goebel, and J. R. Douceur. Single instance storage in Windows 2000. In *Proceedings of the 4th USENIX Windows Systems Symposium*, pages 13–24, 2000.

10. Z. Brakerski and G. Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In *Advances in Cryptology – CRYPTO '11*, pages 543–560, 2011.
11. E. Buonanno, J. Katz, and M. Yung. Incremental unforgeable encryption. In *Proceedings of the 8th International Workshop on Fast Software Encryption*, pages 109–124, 2001.
12. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.
13. Y. Dodis and A. Smith. Entropic security and the encryption of high entropy messages. In *Proceedings of the 2nd Theory of Cryptography Conference*, pages 556–577, 2005.
14. J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 617–624, 2002.
15. M. Fischlin. Incremental cryptography and memory checkers. In *Advances in Cryptology – EUROCRYPT '97*, pages 293–408, 1997.
16. M. Fischlin. Lower bounds for the signature size of incremental schemes. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 438–447, 1997.
17. D. M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More constructions of lossy and correlation-secure trapdoor functions. In *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*, pages 279–295, 2010.
18. B. Fuller, A. O'Neill, and L. Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. Cryptology ePrint Archive, Report 2012/005, 2012.
19. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.
20. D. Micciancio. Oblivious data structures: Applications to cryptography. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 456–464, 1997.
21. N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.
22. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 187–196, 2008.
23. S. Quinlan and S. Dorward. Venti: A new approach to archival storage. In D. D. E. Long, editor, *FAST*, pages 89–101. USENIX, 2002.
24. A. Russell and H. Wang. How to fool an unbounded adversary with a short key. In *Advances in Cryptology – EUROCRYPT '02*, pages 133–148, 2002.
25. S. P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Jounal of Cryptology*, 17(1):43–77, 2004.
26. H. Wee. Dual projective hashing and its applications - lossy trapdoor functions and more. To appear in *Advances in Cryptology – EUROCRYPT '12*, 2012.
27. B. Zhu, K. Li, and R. H. Patterson. Avoiding the disk bottleneck in the data domain deduplication file system. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, pages 269–282, 2008.