Incompressible Encodings

Tal Moran[*] Daniel Wichs[†]

June 30, 2020

Abstract

An *incompressible encoding* can probabilistically encode some data m into a codeword c, which is not much larger. Anyone can decode the codeword c to recover the original data m. However, the codeword c cannot be efficiently compressed, even if the original data m is given to the decompression procedure on the side. In other words, c is an efficiently decodable representation of m, yet is computationally incompressible even given m. An incompressible encoding is *composable* if many encodings cannot be simultaneously compressed.

The recent work of Damgård, Ganesh and Orlandi (CRYPTO '19) defined a variant of incompressible encodings as a building block for "proofs of replicated storage". They constructed incompressible encodings in an ideal permutation model, but it was left open if they can be constructed under standard assumptions, or even in the more basic random-oracle model. In this work, we undertake the comprehensive study of incompressible encodings as a primitive of independent interest and give new constructions, negative results and applications:

- We construct incompressible encodings in the *common random string (CRS)* model under either Decisional Composite Residuosity (DCR) or Learning with Errors (LWE). However, the construction has several drawbacks: (1) it is not composable, (2) it only achieves selective security, and (3) the CRS is as long as the data m.
- We leverage the above construction to also get a scheme in the random-oracle model, under the same assumptions, that avoids all of the above drawbacks. Furthermore, it is significantly more efficient than the prior ideal-model construction.
- We give black-box separations, showing that incompressible encodings in the plain model cannot be proven secure under any standard hardness assumption, and incompressible encodings in the CRS model must inherently suffer from all of the drawbacks above.
- We give a new application to "big-key cryptography in the bounded-retrieval model", where secret keys are made intentionally huge to make them hard to exfiltrate. Using incompressible encodings, we can get all the security benefits of a big key without wasting storage space, by having the key to encode useful data.

[*]IDC Herzliya. Email: `talm@idc.ac.il`

[†]Northeastern University and NTT Research Inc. Email: `wichs@ccs.neu.edu`. Research supported by NSF grants CNS-1314722, CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship.

1 Introduction

The entire contents of Wikipedia can be downloaded as a compressed (gzip) file that is several gigabytes in size. Can it be compressed further? This is an interesting question, and there is much work on optimizing compression for various types of data. But there is also an uninteresting answer – the Wikipedia contents can be easily compressed via the link "`www.wikipedia.org`", which allows anyone on the Internet to recover the data! We consider the compression problem in exactly the above scenario, where the data is readily available publicly. However, our goal is to make such data incompressible. That is, we want to come up with an *incompressible encoding* scheme that takes some such data (e.g., Wikipedia content) and probabilistically represents it in a way that is guaranteed to be incompressible, even when the decompression procedure is given the original data on the side for free (e.g., can access Wikipedia over the Internet). We now elaborate on what this primitive is and why it is useful.

Incompressible Encodings. An incompressible encoding scheme consists of a pair of efficient and key-less encoding and decoding procedures $(\mathsf{Enc}, \mathsf{Dec})$, such that $c \leftarrow \mathsf{Enc}(m)$ probabilistically encodes some data m into a codeword c that anybody can efficiency decode to recover $m = \mathsf{Dec}(c)$. The size of the codeword c should not be "much larger" than that of the underlying data m. Even though the codeword c is an efficiently decodable representation of the data m, and cannot contain much additional information since its size is not much larger, we want c to be incompressible even given m. In particular, we consider the following *incompressibility* security game:

1. An adversary chooses some arbitrary data m, which is encoded to get a codeword $c \leftarrow \mathsf{Enc}(m)$.
2. The codeword c is given to an adversarial compression algorithm that outputs some compressed value w.
3. The compressed value w along with underlying data m are given to an adversarial decompressor, who wins if it outputs the codeword c.

We require that no efficient adversary can win the above game with better than negligible probability, unless w is "almost as large" as the entire codeword.

Good vs. Trivial Encodings. The definition of incompressible encodings has two parameters: for data $m \in \{0,1\}^k$, we let $\alpha(k)$ denote the *codeword size*, and $\beta(k)$ denote the *incompressibility parameter*, which bounds size of the value w output by the compressor in the security game.

For a "good" incompressible encoding, we want the codeword size to be essentially the same as the size of the underlying data $\alpha(k) = (1+o(1))k$ and we want the encoded data to be incompressible below the size of essentially the entire data/codeword $\beta(k) = (1-o(1))k = (1-o(1))\alpha(k)$.[1]

On the other hand, there is a "trivial" incompressible encoding, which just appends some randomness r to the data and sets the codeword to $c = (m, r)$. Since r cannot be compressed, the encoding also cannot be compressed below the length of r. This ensures that $\alpha(k) = k + |r|$ and $\beta(k) = |r|$. Therefore a scheme with $\alpha(k) \geq k + \beta(k)$ is "trivial" and we say that an incompressible encoding is "non-trivial" if it beats this bound.

[1]Throughout the introduction, we will ignore additive polynomial terms in the security parameter, which means that the above bounds only hold when the data size k is sufficiently large.

It is easy to see that non-trivial incompressible encodings cannot be constructed information theoretically. This is because, there are at most $2^{\alpha(k)-k}$ possible codewords per message on average, and therefore also certainly for the worst-case message m. A pair of inefficient compression/decompression procedures can enumerate the list of all such codewords (e.g., in lexiographic order) and compress/decompress any codeword in the list just by writing down its index using $\beta(k) = \alpha(k) - k$ bits. Therefore, we will need to rely on computational assumptions to construct non-trivial incompressible encodings.

Local Decoding. We will also want our incompressible encodings to be *locally decodable*, so that any bit of the message can be recovered by only reading a few bits of the codeword. While our constructions have this property, most of the challenges are already present even without this requirement.

Composability. We say that an incompressible encoding scheme, with some incompressibility parameter β, is *composable* if any n independently generated encodings of various messages cannot be compressed below $\sum_{i\in[n]} \beta(k_i)$ bits, where k_i denotes the length of the i'th message. Moreover, the probability of the adversarial compression algorithm outputting less than $\sum_{i\in\mathcal{I}} \beta(k_i)$ bits and the decompression procedure reconstructing all of the codewords $c_i \ : \ i \in \mathcal{I}$ for some set $\mathcal{I} \subseteq [n]$ should be negligible. As we will discuss later, we do not know whether all incompressible encodings are inherently composable (we conjecture otherwise) and therefore we define it as a separate security property of interest.

Keyed vs Key-less Schemes. We mention that any semantically secure encryption scheme would give an incompressible encoding where the decoding procedure needs a secret key. This is because the encryption of some arbitrary data m is indistinguishable from an encryption of random data, which is inherently incompressible even given m. The main difficulty of our problem is that we require the encoding and decoding procedures to be public and key-less; anybody should be able to decode the codeword to recover the data.

1.1 Prior and Concurrent Work

Proofs of Replicated Storage. The work of Damgård, Ganesh and Orlandi [DGO19] studied a primitive called "Proofs of Replicated Storage", which considers a setting where we want to store some data m with a cloud storage provider, who promises to store n replicated copies of the data at different locations ("replicas") to increase fault tolerance. We want to be able to check that the provider is indeed storing n copies of the data. To solve this problem they defined a novel building block called a "replica encoding", which corresponds to our notion of an incompressible encodings that is also *bounded self-composable*; i.e., it is composable when the same message is encoded n times for some a-priori bound n. Using such encodings, they construct "proofs of replicated storage" by encoding the data n separate times and having each replica store a different codeword. This is combined with "Proof of Retreivability" [JK07,SW08,DVW09], which are used to periodically audit each replica and check that it is storing its codeword. The cloud provider cannot meaningfully save on storage while maintaining the ability to pass the audit with non-negligible probability. "Proofs of Replicated Storage" also have applications to the *Filecoin* cryptocurrency [Lab17a,Lab17b], where replicated storage of user data is a resource for mining coins; see [DGO19] for details. We note that some prior notions of "proofs of replicated storage" and "incompressible/replica encodings" were also previously considered by [BBBF18, Fis18, CFMJ19] in settings where the adversary is computationally more constrained than the encoding/decoding procedures and does not have the ability to run these procedures; here we focus on the setting where the adversary can run in arbitrary polynomial time.

Construction of Incompressible Encodings of [DGO19]. In the above context, the work of Damgård, Ganesh and Orlandi [DGO19] (building on an idea from an earlier work of [vJO+12]) proposed a construction of incompressible (replica) encodings based on a family of trapdoor permutations (TDPs) denoted by f_{pk} and an ideal invertible public random permutation P, P^{-1}. To encode a message m, the encoder samples a random TDP public key and trapdoor $(\mathsf{pk}, \mathsf{td})$ and outputs a codeword $c = (\mathsf{pk}, (f_{\mathsf{pk}}^{-1} \circ P)^r(m))$ by applying the function $g(x) = f_{\mathsf{pk}}^{-1}(P(x))$ iteratively for r rounds starting with the message m, where r is some parameter. The codeword is efficiently decodable by computing f_{pk} in the forward direction and making calls to P^{-1}. While the above requires using a TDP whose domain is as large as the entire file, it is also possible to apply the TDP on smaller blocks of the file separately. Unfortunately, the construction has several big deficiencies:

- We discovered that the security proof in the published version of [DGO19] is fundamentally flawed. In fact, we identified some heuristic counter-examples to suggest that their scheme is unlikely to be secure in general with the number of rounds r claimed. However, we conjectured (and it has since been confirmed by a concurrent work, see below) that the scheme can be proved secure when the number of rounds r is made very large: $r = \Omega(kn)$ where k is the file-size (in bits), n is the number of compositions and λ is the security parameter. Since each round takes $\Omega(k)$ time, this means that the scheme has complexity $\Omega(k^2 n)$ and, in particular, runs in time quadratic in the file size, which we envision to be large. Moreover, the scheme needs to perform essentially that many public key operations (RSA exponentiations), which makes the scheme highly impractical. Furthermore, the scheme only achieves bounded-composability where the number of compositions n needs to be known ahead of time and affects the complexity of the scheme.

- The construction works in the "ideal invertible random permutation" model. Furthermore, the domain/range of the ideal permutation has to match that of the TDP. For example, if we use the RSA TDP, then the domain/range of P needs to be $\mathbb{Z}_N^*$ where N is the RSA modulus. It remained as an open problem to give a construction in the standard model, or even in the random oracle model, or even using an ideal permutations where the domain/range is just $\{0,1\}^s$ for some parameter s. (While it is possible to construct indifferentiable [MRH04] ideal permutations from a random oracle [CHK+16], they will not have the required structured domain/range. Furthermore, the indifferentiability framework is insufficient to guarantee security for multi-stage games [RSS11], which includes the security game for incompressible encodings.)

Concurrent Work of [GLW20]. A concurrent and independent work of Garg, Lu and Waters [GLW20] independently discovered the flaw in the proof of [DGO19]. They managed to patch the result by providing a completely new (and highly involved) proof of security for their scheme, when the number of rounds r is made sufficiently large $r = \Omega(kn)$ as discussed above. They also managed to remove the reliance on an "ideal invertible random permutation" with a structured domain and showed how to instantiate the scheme using the random oracle model alone.

1.2 Our Results

Our work initiates the study of incompressible encodings as a primitive of independent interest. We give new positive and negative results as well as a new application of this primitive as follows.

Constructions in the CRS and RO Model. We give a new construction of incompressible encodings in the *common random string (CRS)* model under either the *Decisional Composite Residuosity (DCR)* or the *Learning with Errors (LWE)* assumptions. Our construction relies on certain types of *lossy trapdoor functions* (LTFs) [PW08] that are also permutations/surjective, and our constructions of these may be of independent interest. The encodings have good parameters with codeword size $\alpha(k) = (1 + o(1))k$ and incompressibility parameter $\beta(k) = (1 - o(1))k = (1 - o(1))\alpha(k)$. Furhtermore, the encoding/decoding run-time only scales linearly in the data-size. The scheme is also locally decodable. However, it suffers from three drawbacks:

1. It is *not composable* if all the encodings use the same CRS (but is composable if each encoding gets a fresh CRS).
2. It only achieves *selective security*, where the message being encoded cannot be chosen adaptively depending on the CRS.
3. It has a *long CRS*, linear in the length of the data m.

We also leverage our construction in the CRS model to get a construction in the *random-oracle (RO) model* under the same assumptions that avoids all of the above drawbacks. Namely, it is fully composable without any a priori bound on the number of compositions n, and achieves adaptive security, where the adversary can choose the message after making random-oracle queries.

Our schemes represent a significant improvement over the construction and analysis of [DGO19, GLW20] since:

- We get the first constructions in the CRS model without relying on an ideal object (ideal permutation or random oracle), albeit with the above-mentioned drawbacks.
- Our schemes in both the CRS and RO models are significantly more efficient and our encoding/decoding run time is $O(k)$ rather than $O(k^2)$, where k is the data size. Since we generally envision encoding large data where k is several gigabytes or terabytes, the difference between linear and quadratic run-time is highly significant. (We omit factors in the security parameter in the above bounds).
- Our RO scheme is fully composable and we do not need to bound the number of compositions n ahead of time nor does the efficiency of the scheme degrade with n.
- We can plausibly achieve post-quantum security via our LWE-based construction whereas [DGO19, GLW20] seemed to inherently require trapdoor permutations for which we have no good candidates with post-quantum security.
- Our proof of security is in many ways significantly simpler than that of [GLW20].

Black-Box Separations. We give black-box separations, showing that non-trivial incompressible encodings in the plain model cannot be proven secure via a black-box reduction from any standard hardness assumption, including strong assumptions such as the existence of indistinguishability obfuscation. Moreover, we show that similar black-box separations apply to good incompressible encodings in the CRS model, unless they suffer from all 3 of the above drawbacks: they cannot be fully composable with a single CRS, they cannot achieve adaptive security, and the CRS needs to be essentially as long as the data m. This shows that our results are in some sense optimal.

Application to Big-Key Crypto. We give a new application of incompressible encodings to "big-key cryptography in the bounded-retrieval model" [Dzi06,DLW06,CDD$^+$07,ADW09,ADN$^+$10, BKR16, ...]. Big-key cryptosystems are designed with intentionally huge secret keys in order to make them hard to exfiltrate. They guarantee security even if the adversary can get large amounts of arbitrary "leakage" on the secret key.

Using incompressible encodings, we can get all the security benefits of a big key without wasting storage space, by allowing the key to encode useful data. We do not need to assume that the data has any entropy from the point of view of the adversary; for example the user's secret key could encode the contents of Wikipedia, or the user's movie collection, or other data that a user may want to store offline but is also publicly available on the Internet and may be fully known to the adversary.

In particular, we show how to construct public-key encryption in this model, where the secret key is an incompressible encoding of the user's arbitrary data. Security is maintained even if the adversary can exfiltrate arbitrary information about the secret key, as long as the size of such leakage is bounded by some $(1-o(1))$ fraction of the secret key size. The public key size, ciphertext size and the encryption/decryption run time are all small and only poly-logarithmic in the data size. In particular, each decryption operation only accesses some small subset of the secret key bits.

1.3 Our Techniques

We now delve into each of the results above and the relevant techniques in turn.

Incompressible Encodings in the CRS model. We construct incompressible encodings in the CRS model. In this model, the honest encoding/decoding algorithms as well as the adversarial compression/decompression algorithms have access to a public uniformly random string. Our construction relies on certain types of *lossy tradpor functions* (LTFs) [PW08]. An LTF consists of a family of function f_{pk} indexed by a public key pk. The public key can be sampled in one of two indistinguishable modes: in *injective mode* the function f_{pk} is injective and we can sample pk along with a trapdoor td that allows us to efficiently invert it, and in *lossy mode* the function f_{pk} has a very small image, meaning that $f_{\mathsf{pk}}(x)$ reveals very little information about the input x.

As a starting point, to illustrate the main ideas, let's assume that we have an LTF family $f_{\mathsf{pk}} : \{0,1\}^d \to \{0,1\}^d$ where both the domain and range are equal to $\{0,1\}^d$ for some polynomial d. This means that f_{pk} is a permutation over $\{0,1\}^d$ in injective mode. Let's also assume that the LTF is highly lossy, meaning that the output $f_{\mathsf{pk}}(x)$ in lossy mode only reveals $o(d)$ bits of information about x. To encode some data $m \in \{0,1\}^k$ we think of $m = (m_1, \ldots, m_{k'})$ as consisting of $k' = k/d$ blocks of length d each. We also rely on a common random string $\mathsf{crs} = (u_1, \ldots, u_{k'})$ consisting of k' random blocks of length d each. The encoding procedure $\mathsf{Enc}_{\mathsf{crs}}(m)$

samples a random pk in injective mode, together with a trapdoor td and sets the codeword to be $c = (\mathsf{pk}, f_{\mathsf{pk}}^{-1}(m_1 \oplus u_1), \ldots, f_{\mathsf{pk}}^{-1}(m_{k'} \oplus u_{k'})$. The decoding procedure $\mathsf{Dec}_{\mathsf{crs}}(c)$ recovers m by applying f_{pk} in the forward direction and xor'ing out the u_i components. Moreover, it's easy to see that individual locations of the data can be decoded locally. The codeword is of size $k + |\mathsf{pk}| = (1+o(1))k$.

We prove incompressible security of the above scheme in the selective setting, where the choice of the message m is worst-case but cannot depend on the CRS. We first observe that the joint distribution of $\mathsf{crs} = (u_1, \ldots, u_{k'}), c = (\mathsf{pk}, x_1, \ldots, x_{k'})$ sampled as above with $u_i \leftarrow \{0,1\}^d$ and $x_i = f_{\mathsf{pk}}^{-1}(m_i \oplus u_i)$ is identical to the distribution where we choose $x_i \leftarrow \{0,1\}^d$ uniformly at random and set $u_i = f_{\mathsf{pk}}(x_i) \oplus m_i$. The latter distribution can be sampled without a trapdoor. Therefore, we can indistinguishably switch pk to lossy mode. Now the codeword c has $(1 - o(1))k$ bits of true entropy even conditioned on crs and m, since each of the values $u_i = f_{\mathsf{pk}}(x_i) \oplus m_i$ reveals only $o(d)$ bits of information about each x_i. Therefore, we can argue that in this case the codeword c is (even information-theoretically) incompressible below $(1 - o(1))k$ bits, even given crs and m. But since this case is computationally indistinguishable from the real distribution of crs, c, the same must hold computationally there as well.

To give some more intuition on the above idea, note that in reality the codeword c has very little actual entropy given crs, m. This is inherent since we want to c to be almost the same size as m and it has to decode to m so there is no space left to inject actual entropy. But in the security proof, we indistinguishably move the information about the message m into the crs and allow the codeword to have a high amount of real entropy even given crs, m while preserving the condition that it decodes to m. Therefore, we can argue incompressibility.

Surjective Lossy Functions. There are many constructions of lossy trapdoor functions (LTFs) in the literature (e.g.,) [PW08, KOS10, FGK+13, Zha16, AKPS19]. However, the vast majority of them are not *surjective* and hence are unusable for our scheme where we need to compute inverses $f_{\mathsf{pk}}^{-1}(m_1 \oplus u_1)$ on arbitrary values that we cannot force to be in the image. Fortunately, the work of [FGK+13] gives a construction of a surjective LTF (permutation) based on Paillier's Decisional Composite Residuosity (DCR) assumption [Pai99] using the ideas behind the Damgård-Jurik cryptosystem [DJ01]. Furthermore, this LTF can be made highly lossy to ensure that the output only reveals an $o(1)$ fraction of the information in the input. There is still some subtlety in that the domain and range of the LTF are not bit-strings $\{0,1\}^d$ but rather the group $\mathbb{Z}^*_{N^{s+1}}$ for some RSA modulus N and some parameter s. It turns out that we can nevertheless use it in our construction with minor modifications, while maintaining a uniformly random CRS. This is because we can use truly random bits in the CRS to obliviously sample random elements in the group $\mathbb{Z}^*_{N^{s+1}}$.

Since surjective LTFs are also trapdoor permutations (TDPs), all good candidates rely on assuming the hardness of factoring and it is a long-standing open problem to get constructions from other assumptions, such as DDH or LWE. However, we notice that we don't need our functions to be *injective*. Instead we define a relaxation of surjective LTFs that we call *Surjective Lossy Functions (SLFs)*. SLFs have two modes: a surjective (but not necessarily injective) mode and a lossy mode. In surjective mode, the image of the function f_{pk} is the entire range while in lossy mode the image is much smaller. Furthermore, in surjective mode we require a trapdoor that allows us to sample a random preimage of any value in the range. If the function is surjective but not injective, the input-length must be larger than the output-length, and we will require that it is at most $(1 + o(1))$ times larger to ensure that our encodings have small α. We show that such SLFs suffice in our construction of incompressible encodings. We then proceed to construct such SLFs under the learning with errors (LWE) assumption [Reg05]. Our starting point is the surjective trapdoor function of [Ajt96, GPV08, MP12] defined as $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{x} \in \mathbb{Z}_q^m$ has "small" entries. Unfortunately, in the best known instantiations [MP12], the input length (in bits) is at least twice as large as the output length, while we only want a $(1 + o(1))$ increase. We show a new technique to get around this. The high level idea is somewhat similar to a recent work of [CGM19], and we rely on "approximate trapdoors" where, given $\mathbf{y}$, we can sample $\mathbf{x}$ such that $\mathbf{A}\mathbf{x}$ is close but not identical to $\mathbf{y}$. We modify the function to $f_{\mathbf{A}}(\mathbf{x}) = \lceil \mathbf{A}\mathbf{x} \rfloor_p$ by applying some rounding to the output to get rid of this difference. This allows us to optimize the ratio of input-size to output-size and get an improvements over exact trapdoors. To ensure a $(1 + o(1))$ overhead, our instantiation of this idea is somewhat different and arguably simpler than that of [CGM19]. Furthermore, we also show that this function has a lossy mode where the output only reveals an $o(1)$ fraction of the information in the input, using the techniques of [GKPV10, AKPW13]. Overall, we believe that this construction may be of independent interest.

Composability and HILL vs Yao Entropies. We cannot prove our construction of incompressible encodings in the CRS model to be composable if the same CRS is used to generate all the encodings. However, we show that it is composable if each encoding is given a fresh CRS. But first, let us discuss why composability is generally difficult.

One may be tempted to conjecture that *all* incompressible encodings are inherently composable. For example, it seems intuitive that if the adversary needs to store $\beta(k)$ bits to compress one codeword, she would need $2\beta(k)$ bits to compress two codewords of two length k messages (potentially the same message). Surprisingly, this intuition does not naturally translate into a proof — how would one design a reduction which leverages a compression algorithm that takes two codewords and compressed them below 2β bits to compress a single codeword below β bits? The situation is highly reminiscent of the "leakage amplification" problem in leakage-resilient cryptography: if some cryptosystem is secure even given β bits of leakage on its secret key, does that mean that two copies of the cryptosystem cannot be simultaneously broken even given 2β bits of leakage on the two keys? Surprisingly this is not the case and it was possible to construct clever counter-examples showing that the above does not generically hold in some cases [JP11, LW10, DJMW12]. We conjecture that counter-examples may also exist for incompressible encodings, at least under strong enough assumptions, and leave it as an open problem to come up with one.

Given that it is unknown (and perhaps unlikely) that all incompressible encodings are composable, we leverage a special property of our construction to show composability. In particular, the definition of incompressible encodings essentially says that c has high "Yao incompressibility entropy" [Yao82, BSW03, HLR07] even given crs. However, for our construction, we showed that c even has a high HILL entropy [HILL99] given crs, since the distribution of (crs, c) is computationally indistinguishable from one, where c has true (statistical) entropy given crs. Having HILL entropy is a stronger property than having incompressibility entropy and we leverage this to prove composition. In particular, while we do not know if the "Yao incompressibility entropy" of independent samples adds up, we do know that this is the case for HILL entropy: i.e., the HILL entropy of independent samples (crs_i, c_i) for $i \in [n]$ is the sum of the individual HILL entropies. This property implies composability for our encodings.

Construction in the Random Oracle Model. Our CRS model construction needed a long CRS, and a fresh CRS for each encoding if we wanted to argue composition. Furthermore, it only achieved selective security. We show how to resolve all of these drawbacks in the random oracle model. In this model all honest and adversarial algorithms have access to a truly random (fixed-length) function $\mathsf{RO} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$, where λ is the security parameter. The idea behind our construction is simple: the encoding procedure chooses fresh short randomness $r \leftarrow \{0,1\}^\lambda$ on each execution and then calls $\mathsf{RO}(r,1), \mathsf{RO}(r,2), \ldots$ to expand it into an arbitrarily long crs as needed; it then uses the incompressible encoding scheme in the CRS model with the above crs to encode the data and appends r to the codeword. This essentially corresponds to using a fresh CRS for each encoding. Note that, in the random oracle model, we allow the adversary to choose the message adaptively after making random oracle queries. However, we only need to rely on selective security in the CRS model since the adversary cannot predict r ahead of time and hence the crs used in each encoding is essentially freshly chosen after the adversary selects the message.

Black-Box Separations. We give black-box separations, showing that incompressible encodings in the plain model cannot be proven secure under any standard hardness assumption, and incompressible encodings in the CRS model must inherently suffer from all of the drawbacks that ours has. Our black-box separations have a similar flavor to those of [Wic13] and we first explain their framework tailored to our setting.

They define the class of "single-stage game" assumptions, which are modeled via a game between a (potentially inefficient) challenger and a single stateful adversary; the assumption states that any polynomial-time adversary should have at most a negligible advantage in winning the game. This captures essentially all standard assumptions used in cryptography, from the hardness of factoring to indistinguishability obfuscation (iO).[2] We observe that the security definition of incompressible encodings is *not* a single-stage game since it involves two separate entities (the compressor and the decompressor) who cannot fully share state or communicate with each other — the compressor is limited in the number of bits it can pass to the decompressor. This makes it possible to separate the security of incompressible encodings from all single-stage game assumptions.

The separation works by constructing a "simulatable attacker". This is an *inefficient* (exponential size) attacker $\mathcal{A} = (\mathcal{A}.\mathsf{Compress}, \mathcal{A}.\mathsf{Expand})$ that breaks incompressible-encoding security. However we also design an efficient simulator $\mathcal{A}'$, such that one cannot (statistically) distinguish between black-box access to $\mathcal{A}$ versus $\mathcal{A}'$. Unlike the attackers $\mathcal{A}$, the simulator $\mathcal{A}'$ is a single fully stateful entity that can fully remember any inputs for invocations of $\mathcal{A}'.\mathsf{Compress}$ and use them to answer future invocations of $\mathcal{A}'.\mathsf{Expand}$. Therefore $\mathcal{A}'$ is not a legal attacker against the incompressible encoding. However, if some reduction can break a single-stage game assumption given black-box access to the legal (but inefficient) $\mathcal{A}$ then it would also be able to do so given access to the efficient (but illegal) $\mathcal{A}'$ which means that the assumption is false.

On a very high level, our adversary $\mathcal{A}$ uses "brute-force" to find the index i of the codeword in the lexicographic ordering of all codewords that decode to m and applies a random permutation on i to get the compressed value w. The decompressor inverts the permutation on w to recover i and uses that to recover the codeword. The efficient simulator $\mathcal{A}'$ just chooses a fresh random w to compress each codeword but keeps a table of codewords it has seen with the corresponding w

[2]This is a larger class than falsifiable assumptions [Nao03, GW11], where the challenger is also required to be efficient.

values it gave. To decompress given w, it just finds the corresponding codeword in the table.

In the above, we need to argue that the brute-force approach always works and that the number of codewords that decode to m is small so that the index i does not require too many bits to transmit. This holds in the plain model, when we choose the worst case message, or even in the CRS model when the message m can be chosen randomly but after the CRS if fixed. However, it fails in the selective-security setting in the CRS model – as it should, since we have a construction! This is because $\mathcal{A}$ may be given a CRS for which there are too many codewords that decode to the message m and hence it cannot write down the index i. If $\mathcal{A}$ failed in these cases but succeeded otherwise, we could use it to distinguish between such a CRS and a truly random one, which is something that cannot be efficiently simulated and indeed allows for a security reduction under standard assumptions.

Application to Big-Key Crypto. We give a new application of incompressible encodings to "big-key cryptography in the bounded-retrieval model" [Dzi06,DLW06,CDD$^+$07,ADW09,ADN$^+$10, BKR16], where secret keys are intentionally huge to make them hard to exfiltrate. In particular, these works envision cryptosystems where the keys are many gigabytes or terabytes in size. Even if an adversary hacks into the system storing the secret key and manages to leak out large amounts of arbitrary data (but sufficiently less than the key size), we want the security of the cryptosystem to be maintained. For example, in the case of encryption, this should not enable the adversary to break semantic security of any ciphertexts sent in the future (unfortunately, the adversary can always perform decryption on past ciphertexts on the compromised device and leak out some plaintext bits so we cannot guarantee security of past ciphertexts). In such schemes, we want to maintain efficiency of honest users even as the key grows, and therefore the cryptosystem cannot even read the entire key to perform basic operations such as encryption and decryption. Prior work focused on constructing various primitives in this setting including symmetric-key encryption, public-key encryption and authenticated key agreement.

One big disadvantage of big-key cryptography is that it forces honest users to "waste" space by storing a huge random key of the cryptosystem. In this work, we propose to get rid of this waste by converting useful data that the user would store anyway into a cryptographic key. For example, the user can take their local movie/music collection (or an offline copy of Wikipedia etc.) and turn it into a cryptographic key for a "big-key" cryptosystem. We do not assume that the underlying data has any entropy from the point of view of the attacker; for example, the movie/music collection may be easily compressible into a very short list of titles that are available for download on the Internet. In general, we allow the attacker to choose the data in a worst-case fashion. This seems to make such data unusable as a cryptographic key, since it may be completely known to the adversary! However, this is where incompressible encodings come in. We ask the user to store an encoded version of the data and use the codeword as the key. The user can still access the data since it can be efficiently (locally) decoded. However, an adversary cannot easily exfiltrate the key by compressing it, even if the underlying data is completely known.

Turning the above high-level idea into workable big-key cryptosystems presents two challenges:

- We need to rely on big-key cryptosystems where the secret key can be an arbitrary string, rather than cryptosystems that choose a carefully structured secret key. This is especially a challenge for public-key encryption (PKE) schemes, where keys tend to have a lot of structure.

- We need to rely on a big-key cryptosystem that is secure with leakage, as long as the secret key is incompressible, even if it is not uniformly random. In particular, any attack against the leakage resilience of the encryption scheme should translate into a compression/decompression procedure on the secret key.

We call such cryptosystems *encoding-friendly*, and it is easy to see that they remain secure if we set their key to be an incompressible encoding of some arbitrary data.

We construct an *encoding-friendly* big-key PKE in the bounded-retrieval model, where the secret key can be arbitrarily large but the public key, ciphertexts, and encryption/decryption complexity are all small, only poly-logarithmic in the key size. Our work departs significantly from the prior construction of big-key PKE in the bounded-retrieval model of [ADN+10], which was not encoding friendly (the secret key in that scheme consisted of many "identity secret keys" in a special identity-based encryption scheme and hence required a large amount of structure). Instead, our construction relies on *laconic oblivious transfer (LOT)* [CDG+17], which can in turn be constructed under a wide range of assumptions such as CDH, LWE, and Factoring [BLSV18]. In an LOT scheme, one can take a long string $x \in \{0,1\}^k$ and hash it down into a short digest which acts as a public-key $\mathsf{pk} = h(x)$. One can then encrypt some arbitrary data μ under the public key pk with respect to a tuple $(i, b) \in [k] \times \{0,1\}$ such that the resulting ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu, (i, b))$ can be decrypted correctly using x if the i'th bit of x is $x[i] = b$. On the other hand, if $x[i] = 1-b$ then the encryption is semantically secure even given x.

In our construction of an encoding-friendly big-key PKE, we can take an arbitrary (big) value x as the secret key and define the corresponding (short) public key as $\mathsf{pk} = h(x)$. To encrypt a message μ, we apply a secret sharing to derive random shares $\mu_1, \ldots, \mu_\lambda$ that sum up to μ, where λ is the security parameter. We then choose λ random indices $i_1, \ldots, i_\lambda$ and create 2λ LOT ciphertexts $\mathsf{ct}_{j,0} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu_j, (i_j, 0))$ and $\mathsf{ct}_{j,1} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mu_j, (i_j, 1))$ where $\mathsf{ct}_{j,0}$ can be decrypted if $x[i_j] = 0$ and $\mathsf{ct}_{j,1}$ can be decrypted if $x[i_j] = 1$ respectively. We send all the ciphertexts along with the indices i_j. The decryption algorithms runs the LOT decryption on the ciphertexts $\mathsf{ct}_{j,x[i_j]}$ using the secret key x.

To prove security, we argue that we can extract (almost all of) x from any successful distinguisher given the leakage. We do this by choosing random ciphertexts and for each j testing if the distinguisher's advantage goes down noticeably when we replace the component $\mathsf{ct}_{j,0}$ by an encryption of random junk: if it does then we learn $x[i_j] = 0$ (since otherwise he could not notice this change) and otherwise we know $x[i_j] = 1$ (since if his advantage remains high, he must be decrypting $\mathsf{ct}_{j,1}$). By doing this for sufficiently many random ciphertexts we can recover a $(1 - o(1))$ fraction of the bits of x. This gives us a way to compresss x, by writing down the leakage together with a few additional bits that we can't recover from the distinguisher, and decompress it by running the distinguisher.

2 Organization

We present the definition(s) of incompressible encodings in Section 4. Then in Section 5 we discuss composability and give a general transformation from certain types of (HILL-entropic) incompressible encodings in the CRS model to composable encodings in the RO model. In Section 6 we define *surjective lossy functions* (SLFs), which will be our main tool for constructing incompressible encodings and we give constructions under DCR and LWE. Then, in Section 7, we show how to construct incompressible encodings from SLFs. In Section 8, we give our black-box separation results. Lastly, in Section 9, we present an application to big-key cryptography in the bounded retrieval model.

3 Preliminaries

Throughout, we let λ denote the security parameter. By default, all our statements hold in the non-uniform model of computation and we define PPT algorithms as circuits of size polynomial in their input and the security parameter. For $n \in \mathbb{Z}$ we let $[n]$ denote the set $[n] = \{1, \ldots, n\}$. When X is a distribution, or a random variable following this distribution, we let $x \leftarrow X$ denote the process of sampling x according to the distribution X. If $\mathcal{X}$ is a set, we let $x \leftarrow \mathcal{X}$ denote sampling x uniformly at random from $\mathcal{X}$.

Let X, Y be random variables, potentially parameterized by the security parameter. We define their *statistical difference* as $\mathsf{SD}(X, Y) = \frac{1}{2}\sum_u |\Pr[X = u] - \Pr[Y = u]|$. We write $X \overset{\text{s}}{\approx} Y$ if the statistical distance is negligible in the security parameter. We write $X \overset{\text{c}}{\approx} Y$ if they are computationally indistinguishable: for all PPT distinguishers D we have $|\Pr[D(X) = 1] - Pr[D(Y) = 1]| \leq \text{negl}(\lambda)$.

The *min-entropy* of a random variable X is $H_\infty(X) = -\log(\max_x \Pr[X = x])$. Following Dodis et al. [DORS08], we define the (average) conditional min-entropy of X given Y as: $H_\infty(X|Y) = -\log\left(\mathbb{E}_{y \leftarrow Y}\left[2^{-H_\infty(X|Y=y)}\right]\right)$. Note that $H_\infty(X|Y) = k$ iff the optimal strategy for guessing X given Y succeeds with probability 2^{-k}.

Lemma 3.1. *For any random variables X, Y where Y is supported over a set of size T we have $H_\infty(X|Y) \leq H_\infty(X) - \log T$.*

4 Defining Incompressible Encodings

We begin by giving a definition of incompressible encodings. Our first definition does not require composability and can be thought of as a simplified version of the replica encoding definition of [DGO19].

Definition 4.1. *An (α, β)-incompressible encoding scheme consists of PPT algorithms* $(\mathsf{Enc}, \mathsf{Dec})$*. We require the following properties:*

Correctness: *There is some negligible μ such that for all $\lambda \in \mathbb{N}$ and all $m \in \{0,1\}^*$ we have* $\Pr[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, m)) = m] = 1 - \mu(\lambda)$.

α-Expansion: *For all $\lambda, k \in \mathbb{N}$ and all $m \in \{0,1\}^k$ we have* $\Pr[|\mathsf{Enc}(1^\lambda, m)| \leq \alpha(\lambda, k)] = 1$.

β-Incompressibility: *Consider the following "compression experiment"* $\mathsf{CompExp}_{\mathcal{A}}(1^\lambda)$ *with an adversary* $\mathcal{A} = (\mathcal{A}.\mathsf{Select}, \mathcal{A}.\mathsf{Compress}, \mathcal{A}.\mathsf{Expand})$*:*

- $(m, \mathsf{aux}) \leftarrow \mathcal{A}.\mathsf{Select}(1^\lambda)$.
- $c \leftarrow \mathsf{Enc}(1^\lambda, m)$.
- $w \leftarrow \mathcal{A}.\mathsf{Compress}(\mathsf{aux}, c)$.
- $c' \leftarrow \mathcal{A}.\mathsf{Expand}(\mathsf{aux}, w)$.
- *Output 1 if $c = c'$ and $|w| \leq \beta(\lambda, |m|)$.*

We require that for all PPT $\mathcal{A}$ we have $\Pr[\mathsf{CompExp}_{\mathcal{A}}(1^\lambda) = 1] = \text{negl}(\lambda)$.

We also refer to a "good" incompressible encoding, without specifying parameters (α, β) to refer to an (α, β)-incompressible encoding where $\alpha(\lambda, k) = k(1 + o(1)) + \text{poly}(\lambda)$ and $\beta(\lambda, k) = k(1 - o(1)) - \text{poly}(\lambda) = \alpha(\lambda, k)(1 - o(1)) - \text{poly}(\lambda)$.

Composability. We also define a stronger notion of *composable* incompressible encodings. This can be thought of as a generalization of the replica encoding definition of [DGO19], which only required "self-composition" when the same message was encoded many times.

Definition 4.2. *An (α, β)-incompressible encoding is* composable *if the following holds. Consider the following "composable compression experiment"* $\mathsf{CCExp}_{\mathcal{A}}(1^\lambda)$ *with an adversary* $\mathcal{A} = (\mathcal{A}.\mathsf{Select}, \mathcal{A}.\mathsf{Compress}, \mathcal{A}.\mathsf{Expand})$*:*

- $(\{m_i\}_{i=1}^n, \mathsf{aux}) \leftarrow \mathcal{A}.\mathsf{Select}(1^\lambda)$.
- *For* $i = 1 \ldots, n$*:* $c_i \leftarrow \mathsf{Enc}(1^\lambda, m_i)$.
- $w \leftarrow \mathcal{A}.\mathsf{Compress}(\mathsf{aux}, \{c_i\}_{i=1}^n)$.
- $\{c'_i\}_{i=1}^n \leftarrow \mathcal{A}.\mathsf{Expand}(\mathsf{aux}, w)$.
- *Let* $\mathcal{I} = \{i \ : \ c'_i = c_i\}$*. Output 1 if* $\mathcal{I} \neq \emptyset$ *and* $|w| \leq \sum_{i \in \mathcal{I}} \beta(\lambda, |m_i|)$.

We require that for all PPT $\mathcal{A}$ we have $\Pr[\mathsf{CCExp}_{\mathcal{A}}(1^\lambda) = 1] = \text{negl}(\lambda)$.

CRS Model. We can generalize the above definitions to the *common random string* (CRS) model, where the encoding/decoding algorithms as well as the adversary $\mathcal{A}$ are given a random string $\mathsf{crs} \leftarrow \{0,1\}^{t(\lambda,k)}$ as an input. The length $t(\lambda, k)$ of the crs may depend on the message length k. In the CRS model, we distinguish between *selective security* and *adaptive security*. For selective security, the crs is not given to $\mathcal{A}.\mathsf{Select}$ but is given to $\mathcal{A}.\mathsf{Compress}, \mathcal{A}.\mathsf{Expand}$, meaning that the adversary's choice of the data m cannot depend on the crs. For adaptive security, $\mathcal{A}.\mathsf{Select}$ is also given the crs, and therefore the choice of m can depend on the crs. We say a scheme is selectively (resp. adaptively) β-incompressible in the CRS model.

Random Oracle Model. We can also generalize the above definitions to the *random-oracle model*, where the encoding/decoding algorithms as well as the adversarial algorithms $\mathcal{A}.\mathsf{Select}$, $\mathcal{A}.\mathsf{Compress}$, $\mathcal{A}.\mathsf{Expand}$ are given oracle access to a truly random function $\mathsf{RO} \ : \ \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$. Note that, in the random oracle model, we automatically require adaptive security, where the adversary's choice of the message m can adaptively depend on its random-oracle queries.

Locally Decodable. An incompressible encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ is locally decodable if there is some additional local decoding procedure that can recover any bit of the encoded message in time that is only poly-logarithmic in the message length k. In particular, we require that there is an algorithm $\mathsf{LocalDec}^c(1^\lambda, k, i)$ that gets RAM access to the input c and runs in time $\text{poly}(\lambda, \log k)$ such that the following holds. There is a negligible μ such that for any $\lambda, k \in \mathbb{N}$, any $m = (m_1, \ldots, m_k) \in \{0,1\}^k$ and any $i \in [k]$ we have $\Pr[\mathsf{LocalDec}^c(1^\lambda, k, i) = m_i \ : \ c \leftarrow \mathsf{Enc}(1^\lambda, m)] = 1 - \mu(\lambda)$.

Note on (Im)Perfect Correctness. While it will be convenient to allow imperfect correctness (with negligible failure probability) in our definition, we note that we can take any scheme with imperfect correctness and convert it into one with perfect correctness at a slight loss of security. The idea is to modify the encoding algorithm to also tests whether decoding succeeds: if so it outputs the codeword with a 0 appended to it and otherwise it outputs the message in the clear with a 1 appended to it. The decoding checks the appended bit and either applies the original decoding if it is a 0 or just outputs the rest of the codeword if the bit is a 1. If correctness of the original scheme holds with overwhelming probability than the above transformation cannot harm security.

Note on Message Selection. In the above, we allow an adversary $\mathcal{A}.\mathsf{Select}$ to choose the message m along with some auxiliary information aux (which can without loss of generality include the message m). For non-uniform adversaries, we can assume that $\mathcal{A}.\mathsf{Select}$ is deterministic (by hard-coding the worst-case choice of its random coins). For incompressibility in the plain model or with selective security in the CRS model, this means that we can get rid of $\mathcal{A}.\mathsf{Select}$ from the definition and simply quantify over all worst-case choices of the message m and think of the corresponding aux as being hard-coded in the algorithms $\mathcal{A}.\mathsf{Compress}, \mathcal{A}.\mathsf{Expand}$. However, for adaptive security in the CRS model or security in the random oracle model, we must allow m to be chosen adaptively depending on the CRS or the random oracle queries.

5 HILL-Entropic Encodings and Composition

HILL-Entropic Encodings. We can think of the definition of incompressible encodings as roughly requiring that c has high "Yao incompressibility entropy" [Yao82, BSW03, HLR07]. In other words, one cannot efficiently compress c below β bits. We could have defined a stronger variant of "HILL-Entropic Encodings" that would require c to have high HILL entropy [HILL99]. In other words, for any fixed message m, if we consider the random variable C denoting the output of $\mathsf{Enc}(1^\lambda, m)$ then it is computationally indistinguishable from some C' such that the conditional min-entropy $H_\infty(C') \geq \beta$.

Impossibility in the Plain Model. Unfortunately, the notion of "HILL-Entropic Encodings" is simply unachievable in the plain model for any non-trivial $\beta \geq \alpha - k$. In particular, consider the message $m^* = \text{argmin}_m |\mathcal{C}_m|$ that minimizes the size of the set $\mathcal{C}_m = \{c \ : \ \mathsf{Dec}(c) = m\}$ of codewords that decode to m. Since $\sum_m |\mathcal{C}_m| \leq 2^\alpha$ we know that $|\mathcal{C}_{m^*}| \leq 2^{\alpha-k}$. Consider a PPT distinguisher $\mathcal{D}$ that gets m^* as non-uniform advice such that $\mathcal{D}(c) = 1$ if $\mathsf{Dec}(c) = m^*$. Then $\Pr[\mathcal{D}(C) = 1] = 1$ for $C \equiv \mathsf{Enc}(1^\lambda, m^*)$ but for any random variable C' such that $H_\infty(C') \geq \alpha - k + 1$ we have

$$\Pr[\mathcal{D}(C') = 1] = \Pr[C' \in \mathcal{C}_{m^*}] = \sum_{c \in \mathcal{C}_{m*}} \Pr[C' = c] \leq 2^{\alpha-k} 2^{-(\alpha-k+1)} \leq \frac{1}{2}.$$

HILL-Entropic Encodings in the CRS Model. On the other, the above impossibility fails in the CRS model and we will construct (selectively-secure) "HILL-Entropic Encodings" in the CRS model. We define these precisely as follows.

Definition 5.1 (HILL-Entropic Encoding)**.** *An* (α, β)*-HILL-Entropic encoding scheme with selective security in the CRS model consists of PPT algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *with the same syntax,* *correctness, and expansion requirements as incompressible encodings. We also require that there is a (potentially inefficient) algorithm* SimEnc. *For any polynomial $k = k(\lambda)$ and any ensemble of messages $m = \{m_\lambda\}$ of length $|m_\lambda| = k(\lambda)$, consider the following "real" experiment:*

- $\mathsf{crs} \leftarrow \{0,1\}^{t(\lambda,k)}$
- $c \leftarrow \mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m_\lambda)$

and let CRS, C *denote the random variables for the corresponding values in the "real" experiment. Also consider the following "simulated" experiment:*

- $(\mathsf{crs}', c') \leftarrow \mathsf{SimEnc}(1^\lambda, m_\lambda)$

and let CRS', C' *denote the random variables for the corresponding values in the "simulated" experiment. We require that* $(\mathsf{CRS}, C) \overset{c}{\approx} (\mathsf{CRS}', C')$ *and* $H_\infty(C' \mid \mathsf{CRS}') \geq \beta(\lambda, k)$.

Theorem 5.2. *Any (α, β)-HILL-Entropic encoding with selective security in the CRS model is also a (α, β')-incompressible encoding scheme with selective security in the CRS model, where $\beta' = \beta - \lambda$.*

Proof. Assume that a PPT attacker $\mathcal{A} = (\mathcal{A}.\mathsf{Select}, \mathcal{A}.\mathsf{Compress}, \mathcal{A}.\mathsf{Expand})$ has a non-negligible probability in breaking the selective β'-incompressible security. In particular, this means that the following experiment outputs 1 with non-negligible probability:

- $(m, \mathsf{aux}) \leftarrow \mathcal{A}.\mathsf{Select}(1^\lambda)$, $|m| = k$
- $\mathsf{crs} \leftarrow \{0,1\}^{t(\lambda,k)}$
- $c \leftarrow \mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m)$
- $w \leftarrow \mathcal{A}.\mathsf{Compress}(\mathsf{crs}, \mathsf{aux}, c)$.
- $c^* \leftarrow \mathcal{A}.\mathsf{Expand}(\mathsf{crs}, \mathsf{aux}, w)$.
- Output 1 if $c = c^*$ and $|w| \leq \beta'(\lambda, k)$.

Without loss of generality, we can assume $\mathcal{A}.\mathsf{Select}$ is deterministic (since we can hard-code optimal choice of randomness as advice). Let $\{m_\lambda, \mathsf{aux}_\lambda\} = \mathcal{A}.\mathsf{Select}(1^\lambda)$ and define $\mathcal{A}.\mathsf{Compress}(\mathsf{crs}, c) = \mathcal{A}.\mathsf{Compress}(\mathsf{crs}, \mathsf{aux}_\lambda, c)$ and $\mathcal{A}.\mathsf{Expand}(\mathsf{crs}, w) = \mathcal{A}.\mathsf{Expand}(\mathsf{crs}, \mathsf{aux}_\lambda, w)$ with the values aux_λ hard-coded as advice. We can then rewrite the above experiment to see that the following outputs 1 with non-negligible probability:

- $\mathsf{crs} \leftarrow \{0,1\}^{t(\lambda,k)}$
- $c \leftarrow \mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m_\lambda)$
- $w \leftarrow \mathcal{A}.\mathsf{Compress}(\mathsf{crs}, c)$.
- $c^* \leftarrow \mathcal{A}.\mathsf{Expand}(\mathsf{crs}, w)$.
- Output 1 if $c = c^*$ and $|w| \leq \beta'(\lambda, k)$.

But by the definition of HILL-entropic security, the following experiment is indistinguishable from the above and must therefore also output 1 with non-negligible probability:

- $(\mathsf{crs}', c') \leftarrow \mathsf{SimEnc}(1^\lambda, m_\lambda)$
- $w \leftarrow \mathcal{A}.\mathsf{Compress}(\mathsf{crs}', c')$.
- $c^* \leftarrow \mathcal{A}.\mathsf{Expand}(\mathsf{crs}', w)$.
- Output 1 if $c' = c^*$ and $|w| \le \beta'(\lambda, k)$.

But (letting capital letters denote random variables for the corresponding lower-case values in the experiment), since $H_\infty(C'|\mathsf{CRS}') \ge \beta$ we have $H_\infty(C'|\mathsf{CRS}', W) \ge \beta - \beta' \ge \lambda$. Therefore $\Pr[\mathcal{A}.\mathsf{Expand}(\mathsf{CRS}', W) = C'] \le 2^{-\lambda}$ and the probability of the above experiment outputting 1 is negligible. □

Composable Encodings in the RO Model. We now show how to convert any selectively HILL-entropic encoding in the CRS model into a composable incompressible encoding scheme in the random oracle model. We rely on a "fixed length" random oracle $\mathsf{RO} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$. Note that, although we start with a selectively secure scheme in the CRS model where the adversary must choose the message before seeing the CRS, our resulting scheme in the random-oracle model allows the adversary to choose the message after making random-oracle queries.

Our construction proceeds as follows. Assume $(\mathsf{Enc}', \mathsf{Dec}')$ is an encoding in the CRS model with a CRS of length $t(\lambda, k)$. We define:

- $\mathsf{Enc}^{\mathsf{RO}}(1^\lambda, m)$: Choose $r \leftarrow \{0,1\}^\lambda$. Compute $\mathsf{crs} = (\mathsf{RO}(r,1), \dots, \mathsf{RO}(r,t')) \in \{0,1\}^{t(\lambda,|m|)}$ where $t' = t(\lambda,k)/\lambda$. Let $\hat{c} \leftarrow \mathsf{Enc}'_{\mathsf{crs}}(1^\lambda, m)$. Output $c = (r, \hat{c})$.
- $\mathsf{Dec}^{\mathsf{RO}}(c = (r, \hat{c}))$: Compute $\mathsf{crs} = (\mathsf{RO}(r,1), \dots, \mathsf{RO}(r,t')) \in \{0,1\}^{t(\lambda,|m|)}$ where $t' = t(\lambda,k)/\lambda$. Output $\mathsf{Dec}'_{\mathsf{crs}}(\hat{c})$.

Theorem 5.3. *If $(\mathsf{Enc}', \mathsf{Dec}')$ is an (α', β')-HILL-entropy encoding with selective security in the CRS model, then $(\mathsf{Enc}, \mathsf{Dec})$ is a composable (α, β)-incompressible encoding in the Random Oracle model where $\alpha = \alpha' + \lambda$ and $\beta = \beta' - \lambda$.*

Proof. The proof proceeds by a sequence of hybrids. In all of these hybrids we assume random oracle queries are answered "on the fly". This means that we keep a table $\mathcal{T}$ which is initially empty and, whenever the adversary or the challenger queries the oracle at some new point $x = (x_1, x_2)$, we look up if some pair (x, y) is in the table and if so we answer with y, else we choose $y \leftarrow \{0,1\}^\lambda$ randomly and add (x, y) to the table. In some hybrids, we "program" the oracle on x to output y, which corresponds to deleting any previous tuples of the form $(x, \cdot)$ from $\mathcal{T}$ and then adding the tuple (x, y) to $\mathcal{T}$.

Hybrid 0: This is the "composable compression experiment" $\mathsf{CCExp}_{\mathcal{A}}(1^\lambda)$ in the random oracle model:

- $(\{m_i\}_{i=1}^n, \mathsf{aux}) \leftarrow \mathcal{A}.\mathsf{Select}^{\mathsf{RO}}(1^\lambda)$.
- For $i = 1 \dots, n$, sample $c_i \leftarrow \mathsf{Enc}^{\mathsf{RO}}(1^\lambda, m_i)$ as follows:
  - Choose $r_i \leftarrow \{0,1\}^\lambda$.
  - Compute $\mathsf{crs}_i = (\mathsf{RO}(r_i, 1), \dots, \mathsf{RO}(r_i, t'))$.
  - Sample $\hat{c}_i \leftarrow \mathsf{Enc}'_{\mathsf{crs}_i}(1^\lambda, m_i)$ and output $c_i = (r_i, \hat{c}_i)$.

- $w \leftarrow \mathcal{A}.\mathsf{Compress}^{\mathsf{RO}}(\mathsf{aux}, \{c_i\}_{i=1}^n)$.
- $\{c'_i\}_{i=1}^n \leftarrow \mathcal{A}.\mathsf{Expand}^{\mathsf{RO}}(\mathsf{aux}, w)$.
- Let $\mathcal{I} = \{i \ : \ c'_i = c_i\}$. Output 1 if $\mathcal{I} \neq \emptyset$ and $|w| \leq \sum_{i \in \mathcal{I}} \beta(\lambda, |m_i|)$.

Hybrid 1: In this hybrid, we compute $c_i \leftarrow \mathsf{Enc}^{\mathsf{RO}}(1^\lambda, m_i)$ by first choosing $\mathsf{crs}_i \leftarrow \{0,1\}^{t(\lambda,k)}, r \leftarrow \{0,1\}^\lambda$ and then programming the oracle on the values $((r,1), \ldots, (r,t'))$ to output crs. In detail, the hybrid experiment is defined as follows:

- $(\{m_i\}_{i=1}^n, \mathsf{aux}) \leftarrow \mathcal{A}.\mathsf{Select}^{\mathsf{RO}}(1^\lambda)$.
- For $i = 1 \ldots, n$: sample c_i as follows:
  - Choose $r_i \leftarrow \{0,1\}^\lambda$, $\mathsf{crs}_i = (\mathsf{crs}_i[1], \ldots, \mathsf{crs}_i[t']) \leftarrow \{0,1\}^{t'(\lambda,|m_i|)\cdot\lambda}$.
  - For $j \in [t']$, program the oracle on (r_i, j) to output $\mathsf{crs}_i[j]$.
  - Sample $\hat{c}_i \leftarrow \mathsf{Enc}'_{\mathsf{crs}_i}(1^\lambda, m_i)$ and output $c_i = (r_i, \hat{c}_i)$.
- $w \leftarrow \mathcal{A}.\mathsf{Compress}^{\mathsf{RO}}(\mathsf{aux}, \{c_i\}_{i=1}^n)$.
- $\{c'_i\}_{i=1}^n \leftarrow \mathcal{A}.\mathsf{Expand}^{\mathsf{RO}}(\mathsf{aux}, w)$.
- Let $\mathcal{I} = \{i \ : \ c'_i = c_i\}$. Output 1 if $\mathcal{I} \neq \emptyset$ and $|w| \leq \sum_{i \in \mathcal{I}} \beta(\lambda, |m_i|)$.

The only time that hybrids 0,1 differ is if in hybrid 1 there exists some $i \in [n]$ such that either:

- $\mathcal{A}.\mathsf{Select}$ queried the random oracle on r_i.
- There exists some $j \leq i$ such that $r_j = r_i$.

If $\mathcal{A}.\mathsf{Select}$ makes $q = \mathrm{poly}(\lambda)$ queries, then the probability of the above is $\frac{qn^2}{2^\lambda} = \mathrm{negl}(\lambda)$. Therefore, hybrid 0 and 1 are indistinguishable.

Hybrid 2: In this hybrid, we sample $(\mathsf{crs}_i, \hat{c}_i) \leftarrow \mathsf{SimEnc}(1^\lambda, m_i)$. In particular, the hybrid proceeds as follows:

1. $(\{m_i\}_{i=1}^n, \mathsf{aux}) \leftarrow \mathcal{A}.\mathsf{Select}^{\mathsf{RO}}(1^\lambda)$.
2. For $i = 1 \ldots, n$: sample c_i as follows:
   - Choose $r_i \leftarrow \{0,1\}^\lambda$, $(\mathsf{crs}_i, \hat{c}_i) \leftarrow \mathsf{SimEnc}(1^\lambda, m_i)$. Output $c_i = (r_i, \hat{c}_i)$
   - For $j \in [t']$, program the oracle on (r_i, j) to output $\mathsf{crs}_i[j]$.
3. $w \leftarrow \mathcal{A}.\mathsf{Compress}^{\mathsf{RO}}(\mathsf{aux}, \{c_i\}_{i=1}^n)$.
4. $\{c'_i\}_{i=1}^n \leftarrow \mathcal{A}.\mathsf{Expand}^{\mathsf{RO}}(\mathsf{aux}, w)$.
5. Let $\mathcal{I} = \{i \ : \ c'_i = c_i\}$. Output 1 if $\mathcal{I} \neq \emptyset$ and $|w| \leq \sum_{i \in \mathcal{I}} \beta(\lambda, |m_i|)$.

Hybrids 1 and 2 are indistinguishable by the definition of a HILL-entropic encoding and a simple hybrid argument where we switch the distribution of $(\mathsf{crs}_i, \hat{c}_i)$ for each i one at a time. (In the reduction, we can hard-code the random coins of $\mathcal{A}.\mathsf{Select}$ as well as the random oracle responses to the queries made by $\mathcal{A}.\mathsf{Select}$ that maximize the distinguishing advantage as advice. In the i'th hybrid, we also hard-code all the other values $(\mathsf{crs}_j, \hat{c}_j)$ for $j \neq i$ as advice since SimEnc may not be efficient).

Assume that the probability of hybrid 2 outputting 1 is $\varepsilon(\lambda)$. We will show that the above implies that ε is negligible. Since hybrid 0 and 2 are indistinguishable, this implies that the probability of hybrid 0 outputting 1 is also negligible, which proves the theorem.

If the probability of hybrid 2 outputting 1 is $\varepsilon(\lambda)$, then:

$$\begin{aligned}\varepsilon(\lambda) = \Pr[\text{ Hybrid 2 outputs 1}] &= \sum_{\mathcal{I}^* \subseteq [n]} \Pr[\text{ Hybrid 2 outputs } 1 \wedge \mathcal{I} = \mathcal{I}^*] \\ = \sum_{t \in [n]} \sum_{\mathcal{I}^* \subseteq [n], |\mathcal{I}^*| = t} &\Pr[\text{ Hybrid 2 outputs } 1 \wedge \mathcal{I} = \mathcal{I}^*]\end{aligned}$$

Therefore, there must be some choice of $\mathcal{I}^* \subseteq [n]$ of some size $|\mathcal{I}^*| = t$ such that

$$\Pr[\text{ Hybrid 2 outputs } 1 \wedge \mathcal{I} = \mathcal{I}^*] \geq \frac{\varepsilon(\lambda)}{n \cdot \binom{n}{t}} \geq \varepsilon(\lambda)/n^{t+1}.$$

Let us fix all of the randomness of the adversary and the challenger in Hybrid 2, *except* for the choice of $(\mathsf{crs}_i, \hat{c}_i) \leftarrow \mathsf{SimEnc}(1^\lambda, m_i)$ for $i \in \mathcal{I}^*$, so as to maximize the probability $\Pr[$ Hybrid 2 outputs $1 \wedge \mathcal{I} = \mathcal{I}^*]$. Clearly, with this fixing, the probability remains at least $\varepsilon(\lambda)/n^{t+1}$. Let us define $\beta^* = \sum_{i \in \mathcal{I}^*} \beta(\lambda, |m_i|)$ and we can also assume without loss of generality that $w \leftarrow \mathcal{A}.\mathsf{Compress}^{\mathsf{RO}}(\cdots)$ always outputs w of length $|w| \leq \beta^*$ while preserving the probability of the above event.

Note that the above choices also fix the entire random oracle RO in steps 3,4 of the experiment, except that on the (fixed) inputs (r_i, j) it outputs $\mathsf{crs}_i[j]$.

Let us define random variables $\mathsf{CRS}_{\mathcal{I}^*}, \hat{C}_{\mathcal{I}^*}$ for the values $\{\mathsf{crs}_i\}_{i \in \mathcal{I}^*}$ and $\{\hat{c}_i\}_{i \in \mathcal{I}^*}$ respectively. We can think of the entire random oracle RO in steps 3,4 as a random variable which only depends on $\mathsf{CRS}_{\mathcal{I}^*}$ but is otherwise independent of $\hat{C}_{\mathcal{I}^*}$. We can also define a random variable W for the value w in the experiment. Then

$$\begin{aligned} H_\infty(\hat{C}_{\mathcal{I}^*} \mid \mathsf{RO}, W) &\geq H_\infty(\hat{C}_{\mathcal{I}^*} \mid \mathsf{RO}) - \beta^* \\ &\geq H_\infty(\hat{C}_{\mathcal{I}^*} \mid \mathsf{CRS}_{\mathcal{I}^*}) - \beta^* \\ &\geq \sum_{i \in \mathcal{I}^*} H_\infty(\hat{C}_i \mid \mathsf{CRS}_i) - \beta^* \\ &\geq \sum_{i \in \mathcal{I}^*} \beta'(\lambda, |m_i|) - \sum_{i \in \mathcal{I}^*} \beta(\lambda, |m_i|) \\ &\geq t \cdot \lambda, \end{aligned}$$

where the first inequality follows since $|W| \leq \beta^*$, the second since RO is a function of $\mathsf{CRS}_{\mathcal{I}^*}$ but is otherwise independent $\hat{C}_{\mathcal{I}^*}$, the third since the tuples $(\hat{C}_i, \mathsf{CRS}_i)$ are independent of each other.

Therefore

$$\varepsilon(\lambda)/n^{t+1} \leq \Pr[C_{\mathcal{I}^*} = \mathcal{A}.\mathsf{Expand}^{\mathsf{RO}}(\mathsf{aux}, W)_{\mathcal{I}^*}] \leq 2^{H_\infty(\hat{C}_{\mathcal{I}^*} \mid \mathsf{RO}, W)} \leq 2^{-t \cdot \lambda}$$

which implies $\varepsilon(\lambda) \leq n^{t+1} 2^{-t\lambda} = \text{negl}(\lambda)$, completing the proof. □

Composability in the Multi-CRS Model. We observe that the above result in the RO model also implies that our CRS model construction is composable in a setting where each encoding is performed with a fresh CRS. In particular, if there were an attack against composable security in the latter setting, it would immediately translate into an attack on the composable security of the former setting.

6 Surjective Lossy Trapdoor Functions

We now introduce our main building block, which we call surjective lossy functions (SLFs). This can be thought of as a relaxation of lossy trapdoor functions [PW08] that are also *permutations*. In particular, while we insist on the functions being surjective, we relax the requirement that they are injective. Instead, we require a surjective mode and a lossy mode. In surjective mode, we also have an inversion trapdoor. There is some domain distribution D such that, when we sample a random value in the range and invert it using the trapdoor, we get (statistically close to) a random sample from D. In lossy mode, there is some small set of size $\leq 2^\ell$ such that the output of the function almost always ends up in that set - we call ℓ the leakage and want to make it as small as possible.

Definition 6.1. *A family of* ℓ-surjective lossy trapdoor functions (ℓ-SLFs) *consists of a polynomial-time computable family of functions* $f_{\mathsf{pk}} : \mathcal{D}_{\mathsf{pk}} \to \mathcal{R}_{\mathsf{pk}}$ *along with the following PPT algorithms:*

- $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$*: generates a public-key in lossy mode.*
- $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$*: generates a public-key in surjective mode, along with a trapdoor.*
- $x \leftarrow D(\mathsf{pk})$*: samples a value* $x \in \mathcal{D}_{\mathsf{pk}}$.
- $x \leftarrow \mathsf{Inv}_{\mathsf{td}}(y)$*: samples a pre-image* x *of* $\in \mathcal{R}_{\mathsf{pk}}$.

We require the following properties.

Surjective Mode: *The following distributions over* (pk, x, y) *are statistically indistinguishable:*

- *Sample* $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$, $x \leftarrow D(\mathsf{pk}), y = f_{\mathsf{pk}}(x)$ *and output* (pk, x, y)
- *Sample* $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$, $y \leftarrow \mathcal{R}_{\mathsf{pk}}, x \leftarrow \mathsf{Inv}_{\mathsf{td}}(y)$ *and output* (pk, x, y).

The above implies that, in particular, we invert correctly:

$$\Pr[f_{\mathsf{pk}}(\mathsf{Inv}_{\mathsf{td}}(y)) = y \;:\; (\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda), y \leftarrow \mathcal{R}_{\mathsf{pk}}] \geq 1 - \text{negl}(\lambda).$$

Lossy Mode: *For any* pk *in the support of* $\mathsf{LossyGen}(1^\lambda)$ *there exists a set* $\mathcal{L}_{\mathsf{pk}}$ *of size* $|\mathcal{L}_{\mathsf{pk}}| \leq 2^{\ell(\lambda)}$ *such that* $\Pr[f_{\mathsf{pk}}(x) \in \mathcal{L}_{\mathsf{pk}} \;:\; \mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda), x \leftarrow D(\mathsf{pk})] = 1 - \text{negl}(\lambda)$.

Indistinguishability: *The following distributions are computationally indistinguishable:*

$$\{\mathsf{pk} : \mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)\} \overset{c}{\approx} \{\mathsf{pk} \;:\; (\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)\}.$$

The above definition captures the main properties of an SLF. However, in the application, we also need some additional properties on the domain, range and the domain distribution. All of the properties would be satisfied ideally if we had a permutation where the domain and range were just $\mathcal{D}_{\mathsf{pk}} = \mathcal{R}_{\mathsf{pk}} = \{0,1\}^{d(\lambda)}$ and the domain distribution $D(\mathsf{pk})$ was just uniform. However, we will need to be more flexible subject to satisfying the following properties.

Definition 6.2 (SLF*: Enhanced SLF)**.** *We say that an* ℓ*-SLF is an* (r, r', d, e, ℓ)-enhanced SLF*, denoted by SLF*, if the domain* $\mathcal{D}_{\mathsf{pk}}$*, the range* $\mathcal{R}_{\mathsf{pk}}$ *and the domain distribution* $D(\mathsf{pk})$ *satisfy the following properties:*

- *Elements of $\mathcal{D}_{\mathsf{pk}}$ can be represented using (at most) $d(\lambda)$ bits.*
- *For any fixed* pk*, the min-entropy of the distribution $D(\mathsf{pk})$ is at least $H_\infty(D(\mathsf{pk})) \geq e(\lambda)$.*
- *The range $\mathcal{R}_{\mathsf{pk}}$ is a group. (We will denote the group operation by addition.)*
- *We can efficiently embed bit-string of length $r(\lambda)$ as elements of $\mathcal{R}_{\mathsf{pk}}$. In particular, there are efficiently computable functions $\mathsf{embed}_{\mathsf{pk}} : \{0,1\}^{r(\lambda)} \to \mathcal{R}_{\mathsf{pk}}$ and $\mathsf{unembed}_{\mathsf{pk}} : \mathcal{R}_{\mathsf{pk}} \to \{0,1\}^{r(\lambda)}$ such that for all $m \in \{0,1\}^{r(\lambda)}$ we have*

$$\Pr[\mathsf{unembed}_{\mathsf{pk}}(\mathsf{embed}_{\mathsf{pk}}(m)) = m \ : \ (\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)] = 1 - \operatorname{negl}(\lambda).$$

- *We can obliviously sample uniformly random elements of $\mathcal{R}_{\mathsf{pk}}$. In particular, there exists some PPT algorithm $y \leftarrow \mathsf{sam}(\mathsf{pk})$ that uses $r'(\lambda)$ random bits to sample a uniformly random values $y \in \mathcal{R}_{\mathsf{pk}}$ along with a PPT algorithm $\mathsf{explain}_{\mathsf{pk}}(y)$ such that $(u, y) \overset{\mathrm{s}}{\approx} (u', y')$, where $u \leftarrow \{0,1\}^{r'(\lambda)}$, $y = \mathsf{sam}(\mathsf{pk}; u)$, $y' \leftarrow \mathcal{R}_{\mathsf{pk}}$, $u' \leftarrow \mathsf{explain}_{\mathsf{pk}}(y)$.*

We say that a scheme is a "good" SLF, without specifying parameters, if it is an (r, r', d, e, ℓ)-SLF* for some $r = r(\lambda)$ with $d = (1 + o(1))r$, $r' = (1 + o(1))r$, $e = (1 - o(1))r$, and $\ell = o(r)$.*

6.1 SLFs from Decision Composite Residuosity

We describe a construction of a good SLF* under the Decision Composite Residuosity (DCR) assumption of Paillier [Pai99]. The construction is identical to that of [FGK+13] and is based on the Damgård-Jurik Cryptosystem [DJ01]. We provide it here for completeness.

The Damgård-Jurik Cryptosystem. Let $N = PQ$ where P, Q are odd primes such that $\gcd(N, \varphi(N)) = 1$. We call such N admissible. When P and Q are sufficiently large and randomly chosen,$N = PQ$ is admissible with all but negligible probability. The following theorem gives the structure of the group $\mathbb{Z}^*_{N^{s+1}}$.

Theorem 6.3 ([DJ01])**.** *For any admissible $N = PQ$ and $s < \min\{P, Q\}$ the map $\psi_{N,s} : \mathbb{Z}_{N^s} \times \mathbb{Z}^*_N \to \mathbb{Z}^*_{N^{s+1}}$ given by $\psi_{N,s}(m, r) = (1 + N)^m r^{N^s} \mod N^{s+1}$ is an isomprphism satisfying*

$$\psi_{N,s}(m_1 + m_2 \bmod N, r_1 r_2 \bmod N^s) = \psi_{N,s}(m_1, r_1) \cdot \psi_{N,s}(m_2, r_2) \bmod N^{s+1}.$$

Moreover, $\psi_{N,s}$ can be inverted in polynomial time given P, Q.

In the Damgård-Jurik [DJ01] cryptosystem, the public key is N and the secret key is P, Q. The encryption of a message $m \in \mathbb{Z}_{N^s}$ is $\psi_{N,s}(m, r)$ for a random $r \in \mathbb{Z}^*_N$ and the decryption inverts $\psi_{N,s}$ using the secret key. The cryptosystem is proven secure under the decision composite residuosity (DCR) assumption stated below.

Definition 6.4 ([Pai99])**.** *The decision composite residuosity (DCR) assumption states that for randomly chosen primes P, Q in the range $[2^{\lambda-1}, 2^\lambda]$ and $N = PQ$ the distributions (N, x) and (N, y) are computatinally indistinguishable where $x \leftarrow \mathbb{Z}^*_{N^2}$ is uniformly random and $y \leftarrow \{z^N \bmod N^2 \ : \ z \in \mathbb{Z}^*_N\}$ is a random N-residue over $\mathbb{Z}^*_{N^2}$.*

Intuitively the DCR assumption states that for $s = 1$, one cannot distinguish between $\psi_{N,s}(m, r)$ versus $\psi_{N,s}(0, r)$ for a uniformly random m, r. It's easy to see that this implies that for any fixed m, m' one cannot distinguish between $\psi_{N,s}(m, r)$ versus $\psi_{N,s}(m', r)$. Moreover, it turns out that the DRC assumption, which is stated for $s = 1$, automatically implies security for arbitrary polynomial s. The following theorem essentially states the that the Damgård-Jurik cryptosystem is semantically secure under the DCR assumption.

Theorem 6.5 ([DJ01]). *For any polynomial $s = \mathrm{poly}(\lambda)$, and for randomly chosen primes P, Q in the range $[2^{\lambda-1}, 2^{\lambda}]$ with $N = PQ$ and for any values $m, m' \in \mathbb{Z}_{N^s}$, the distributions $(N, \psi_{N,s}(m, r))$ and $(N, \psi_{N,s}(m', r))$ over $r \leftarrow \mathbb{Z}_N^*$ are computationally indistinguishable under the DCR assumption.*

Constructing SLFs from DCR. Since $\psi_{N,s}$ is a permutation with cryptographic properties, we could think of setting $\psi_{N,s}$ as the SLF. Unfortunately, it's not clear how to make it lossy directly. Instead, in addition to the modulus N, we add a "ciphertext" $c \in \mathbb{Z}_{N^{s+1}}^*$ to the public key and define

$$f_{\mathsf{pk}} : \mathbb{Z}_{N^s} \times \mathbb{Z}_N^* \to \mathbb{Z}_{N^{s+1}}^* \quad \text{given by} \quad f_{\mathsf{pk}}(x = (m, r)) = c^m \psi_{N,s}(0, r)$$

In surjective (bijective) mode, we set $c = \psi_{N,s}(1, \hat{r})$ to be an encryption of the message 1, and we also add the randomness $\hat{r}$ to the secret key. In that case, $f_{\mathsf{pk}}(m, r) = c^m \psi_{N,s}(0, r) = \psi_{N,s}(m, \hat{r}^m \cdot r)$. We can invert f_{pk} on any value $y \in \mathbb{Z}_{N^{s+1}}^*$ using the secret key, by computing $\psi_{N,s}^{-1}(y) = (m, r')$ and outputting $x = (m, r)$ were $r = r'/\hat{r}^m$. In lossy mode, we set $c = \psi_{N,s}(0, \hat{r})$ to be a random encryption of 0. In that case, the image of the function f_{pk} is the set $\mathcal{L}_{\mathsf{pk}} = \{\psi_{N,s}(0, r') : r' \in \mathbb{Z}_N^*\}$. In other words, in lossy mode, $f_{\mathsf{pk}}(x)$ only contains $\approx \log(N)$ bits of information about the $\approx (s + 1) \log N$ bit value x.

We describe the construction in detail below. Let $s = s(\lambda)$ be a parameter.

- $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$: Generate random λ-bit primes P, Q such that $N = PQ$ is admissible. Let $\hat{r} \leftarrow \mathbb{Z}_N^*$ and set $c = \psi_{N,s}(1, \hat{r})$. Output $\mathsf{pk} = (N, c), \mathsf{sk} = (P, Q, \hat{r})$.
- $\mathsf{LossyGen}(1^\lambda)$: Generate random λ-bit primes P, Q such that $N = PQ$ is admissible. Let $\hat{r} \leftarrow \mathbb{Z}_N^*$ and set $c = \psi_{N,s}(0, \hat{r})$. Output $\mathsf{pk} = (N, c)$.
- $y = f_{\mathsf{pk}}(x)$: The function $f_{\mathsf{pk}} : \mathcal{D}_{\mathsf{pk}} \to \mathcal{R}_{\mathsf{pk}}$ has domain $\mathcal{D}_{\mathsf{pk}} = \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$ and range $\mathcal{R}_{\mathsf{pk}} = \mathbb{Z}_{N^{s+1}}^*$. It is defined by $f_{\mathsf{pk}}(x) = c^m \psi_{N,s}(0, r) \bmod N^{s+1}$, where $x = (m, r) \in \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$.
- $x = (m, r) \leftarrow D(\mathsf{pk})$: samples a uniformly random value $x \leftarrow \mathcal{D}_{\mathsf{pk}}$.
- $x \leftarrow \mathsf{Inv}_{\mathsf{td}}(y)$: use the secret key P, Q to computr $\psi_{N,s}^{-1}(y) = (m, r')$ and output $x = (m, r)$ were $r = r'/\hat{r}^m$.

Theorem 6.6. *For any polynomial $s = s(\lambda)$ the above construction is an (r, r', d, e, ℓ)-SLF* where:*

$$r = (s + 1)2(\lambda - 1), \quad r' = (s + 1)2\lambda + \lambda, \quad d = (s + 1)2\lambda, \quad e = (s + 1)2(\lambda - 1) - 1, \quad \ell = 2\lambda$$

In particular, when $s = \omega(1)$ then the above construction is a good SLF.*

Proof. We begin by showing each of the SLF properties:

- Surjective Mode: When $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$ is sampled in surjective mode, the function f_{pk} is a bijection over $\mathcal{D}_{\mathsf{pk}} \cong \mathcal{R}_{\mathsf{pk}}$ and $\mathsf{Inv}_{\mathsf{td}}$ is the inverse of f_{pk}. In particular, the distribution of (pk, x, y) for $x \leftarrow D(\mathsf{pk})$, $y = f_{\mathsf{pk}}(x)$ is identical to sampling $y \leftarrow \mathcal{R}_{\mathsf{pk}}$ and $x = \mathsf{Inv}_{\mathsf{td}}(y)$.

- Lossy Mode: When $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$ is sampled in lossy mode, we have

$$\mathcal{L}_{\mathsf{pk}} = \{f_{\mathsf{pk}}(x) \;:\; x \in \mathcal{D}_{\mathsf{pk}}\} = \{\psi_{N,s}(0, r') \;:\; r' \in \mathbb{Z}_N^*\}$$

  and therefore $|\mathcal{L}_{\mathsf{pk}}| \leq \log N \leq 2\lambda$.

- Indistinguishability: This follows immediately from Theorem 6.5 with $m = 0$ and $m' = 1$.

Next we discuss the augmented properties to show that the above SLF is also an SLF*.

- Elements of $\mathcal{D}_{\mathsf{pk}} = \mathbb{Z}_{N^s} \times \mathbb{Z}_N^*$ can be represented using $d(\lambda) = (s+1)2\lambda$ bits.

- The min-entropy of the distribution $D(\mathsf{pk})$, which is uniform over $\mathcal{D}_{\mathsf{pk}}$ is $\log|\mathcal{D}_{\mathsf{pk}}| \geq (s+1)2(\lambda-1)-1$. This is because $N \geq 2^{2(\lambda-1)}$.

- The range $\mathcal{R}_{\mathsf{pk}} = \mathbb{Z}_{N^{s+1}}^*$ is a group under multiplication.

- We can efficiently embed bit-string of length $r(\lambda) = (s+1)2(\lambda-1)-1$ as elements of $\mathcal{R}_{\mathsf{pk}}$. We do so, by simply taking the string and interpreting it as an integer $y < N^{s+1}$ in binary. The probability of $y \notin \mathbb{Z}_{N^{s+1}}^*$ is negligible over the random choice of $N = PQ$.

- We can obliviously sample from $\mathcal{R}_{\mathsf{pk}} = \mathbb{Z}_{N^{s+1}}^*$ using $r'(\lambda) = ((s+1)2\lambda + \lambda)$-bits. We do so by defining $\mathsf{sam}(\mathsf{pk})$ to choose a random $r'(\lambda)$-bit integer z and outputting $y = z \bmod N^{s+1}$. This is $2^{-\lambda}$ statistically close to sampling $y \leftarrow \mathbb{Z}_N^{s+1}$ which is statistically close to sampling $y \leftarrow \mathbb{Z}_{N^{s+1}}^*$. The $\mathsf{explain}_{\mathsf{pk}}(y)$ algorithm outputs a random $r'(\lambda)$-bit value z such that $z = y \bmod N^{s+1}$; it does so by setting $t = \lfloor 2^{r'}/N^{s+1} \rfloor$ and outputting $z = y + v \cdot N^{s+1}$ where $v \leftarrow \{0, \ldots, t\}$. For any y, $z = \mathsf{explain}(y)$ is uniformly random over all z such that $\mathsf{sam}(\mathsf{pk}; z) = y$.

□

6.2 SLFs from Learning with Errors

6.2.1 Lattice Preliminaries

For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q. For a vector $\mathbf{e} \in \mathbb{Z}^n$ we write $||\mathbf{e}||_\infty \leq \beta$ if each entry e_i in $\mathbf{e}$ satisfies $|e_i| \leq \beta$. Similarly, for a matrix $\mathbf{E} \in \mathbb{Z}_q^{n \times m}$ we write $||\mathbf{E}||_\infty \leq \beta$ if each entry $e_{i,j}$ in $\mathbf{E}$ satisfies $|e_{i,j}| \leq \beta$. We say that a distribution χ over $\mathbb{Z}$ is β-bounded if $\Pr[|x| \leq \beta \;:\; x \leftarrow \chi] \leq \mathrm{negl}(\lambda)$. By default, all vectors are assumed to be *column* vectors. For integers $q \geq p \geq 2$ we define the rounding function

$$\lceil \cdot \rfloor_p \;:\; \mathbb{Z}_q \to \mathbb{Z}_p \;:\; x \mapsto \lfloor (p/q) \cdot x \rceil$$

If p divides q then the rounding function divides $\mathbb{Z}_q$ into p intervals of size q/p each. If $q = 2^k$ and $p = 2^{k'}$ then $\lceil x \rfloor_p$ corresponds to outputting the k' most significant bits of the k-bit integer x. If $\mathbf{x}$ is a vector we let $\lceil \mathbf{x} \rfloor_p$ denote the component-wise rounding operation.

Learning with Errors (LWE). The learning with errors (LWE) assumption was introduced by Regev in [Reg05].

Definition 6.7 ([Reg05]). *Let n, q be integers and χ a probability distribution over $\mathbb{Z}_q$, all parameterized by the security parameter λ. The (n, q, χ)-LWE assumption says that for all polynomial m the following distributions are computationally indistinguishable*

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \overset{c}{\approx} (\mathbf{A}, \mathbf{u}^t) \qquad : \mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m, \mathbf{u} \leftarrow \mathbb{Z}_q^m.$$

The work of [ACPS09] showed that the (n, q, χ)-LWE assumption above also implies security when the secret is chosen from the error distribution χ:

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \overset{c}{\approx} (\mathbf{A}, \mathbf{u}) \qquad : \mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}, \mathbf{s} \leftarrow \chi^n, \mathbf{e} \leftarrow \chi^m, \mathbf{u} \leftarrow \mathbb{Z}_q^m.$$

The works of [Reg05, Pei09, BLP+13] show that the LWE assumption is as hard as (quantum) solving GapSVP and SIVP under various parameter regimes. In particular, we will assume that for every $q = 2^{\mathrm{poly}(\lambda)}$ there exists some polynomial $n = \mathrm{poly}(\lambda)$ and $\beta = \mathrm{poly}(\lambda)$ along with β-bounded distribution χ such that the $\mathsf{LWE}_{n,q,\chi}$ assumption holds. We refer to the above as the LWE assumption when we don't specify parameters. This is known to be as hard as solving GapSVP and (quantum) SIVP with sub-exponential approximation factors, which is believed to be hard.

The Gadget Matrix and Preimage Sampling. Let $q = B^\gamma$ be a modulus. We define the base-B gadget matrix of dimension n as the matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times\gamma n}$ given by

$$\mathbf{G} = \mathbf{I}_n \times \mathbf{g} = \begin{bmatrix} \cdots \mathbf{g}^t \cdots & & & \\ & \cdots \mathbf{g}^t \cdots & & \\ & & \ddots & \\ & & & \cdots \mathbf{g}^t \cdots \end{bmatrix}$$

where $\mathbf{g} = [1, B, B^2, \ldots, B^{\gamma-1}]$.

Lemma 6.8 (Preimage Sampling [GPV08, MP12]). *Let $q = B^\gamma$ and n, n' be some parameters such that $n, n', \log q$ are polynomial in the security parameter λ and $n \geq \lambda$. There exist PPT algorithms* $\mathsf{SamPre}, \mathsf{Sam}$ *such that the following holds. Let $\mathbf{G} \in \mathbb{Z}_q^{n\times\gamma n}$ be the base-B gadget matrix of dimension n. Let $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n\times n'}$ and let $\mathbf{A} = [\overline{\mathbf{A}} \mid \overline{\mathbf{A}}\mathbf{R} + \mathbf{G}] \in \mathbb{Z}_q^{n\times m}$ where $m = n' + \gamma n$ and $\mathbf{R} \in \mathbb{Z}_q^{n'\times\gamma n}$ with $||\mathbf{R}||_\infty \leq \beta$. Then:*

- $\mathbf{u} \leftarrow \mathsf{Sam}(1^\lambda)$ *samples $\mathbf{u} \in \mathbb{Z}_q^m$ such that $||\mathbf{u}||_\infty \leq m^{2.5}\beta B$,*
- $\mathbf{u} \leftarrow \mathsf{SamPre}_{\mathbf{A},\mathbf{R}}(\mathbf{v})$ *samples $\mathbf{u} \in \mathbb{Z}_q^m$ such that $\mathbf{A}\mathbf{u} = \mathbf{v}$ and $||\mathbf{u}||_\infty \leq m^{2.5}\beta B$,*

where the distribution of $(\mathbf{u}, \mathbf{v})$ is statistically close to $(\mathbf{u}', \mathbf{v}')$ with $\mathbf{u} \leftarrow \mathsf{Sam}(1^\lambda), \mathbf{v} = \mathbf{A}\mathbf{u}$, $\mathbf{v}' \leftarrow \mathbb{Z}_q^n, \mathbf{u}' \leftarrow \mathsf{SamPre}_{\mathbf{A},\mathbf{R}}(\mathbf{v}')$. Furthermore, the distribution of $\mathsf{Sam}(1^\lambda)$ *has min-entropy $H_\infty(\mathsf{Sam}(1^\lambda)) \geq m \log B$.*

The above lemma follows from the works of [GPV08, MP12]. Firstly, [MP12] shows that the lattice $\Lambda^\perp(\mathbf{G})$ has a public basis $\mathbf{S} \in \mathbb{Z}^{m'\times m'}$ with $||\mathbf{S}||_\infty \leq B$ where $m' = \gamma n$. Furthermore, this can be efficiently extended to a basis $\mathbf{T} \in \mathbb{Z}^{m\times m}$ for the lattice $\Lambda^\perp(\mathbf{A})$ using knowledge of

$\mathbf{A}, \mathbf{R}$, where $||\mathbf{T}||_\infty \leq m'\beta B$. This also shows that the smoothing parameter of $\eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \leq ||\mathbf{T}|| \leq m^{1.5}\beta B$. We define $\mathsf{Sam}(1^\lambda)$ to sample from the Discrete Gaussian $D_{\mathbb{Z}^m,s}$ with parameter $s = m^2\beta B$. Following [GPV08], the algorithm $\mathsf{SamPre}_{\mathbf{A},\mathbf{R}}(\mathbf{v})$ uses $\mathbf{A}, \mathbf{R}$ to find the basis $\mathbf{T}$ and uses that to sample from the Discrete Gaussaian $\mathbf{t} + D_{\Lambda^\perp(\mathbf{A}),s,-\mathbf{t}}$ where $\mathbf{t}$ is any solution to $\mathbf{At} = \mathbf{v}$ (which is guaranteed to exist and can be found efficiently). For $\mathbf{u} \leftarrow \mathsf{Sam}(1^\lambda)$ the value $\mathbf{Au}$ is then statistically close to uniform over $\mathbb{Z}_q^n$ and for any $\mathbf{v}$, the distribution of $\mathbf{u}' \leftarrow \mathsf{SamPre}_{\mathbf{A},\mathbf{R}}(\mathbf{v})$ is exactly the conditional distribution of $\mathbf{u} \leftarrow \mathsf{Sam}(1^\lambda)$ subject to $\mathbf{Au} = \mathbf{v}$. The probability that $\mathbf{u} \leftarrow \mathsf{Sam}(1^\lambda)$ or $\mathbf{u}' \leftarrow \mathsf{SamPre}_{\mathbf{A},\mathbf{R}}(\mathbf{v})$ have norm greater than $s\sqrt{m} = m^{2.5}\beta B$ is negligible and, for simplicity, we will modify the algorithms to never output such values. Lastly, we rely on Lemma 2.11 in [PR06] and the fact that the min-entropy of the discrete Gaussian $D_{\mathbb{Z}^m,s}$ is greater than $(s/(2\eta_\varepsilon(\mathbb{Z}^m)))^m \geq (s/\log m)^m \geq B^m$ for the min-entropy claim.

6.2.2 Constructing of SLFs from LWE

Let $n > n'$ and $B, q = B^\gamma, C, p$ be parameters depending on the security parameter λ and assume that p divides q. Let χ be some β-bounded error distribution. Define $m' = \gamma n$ and $m = n' + m'$. Let $\mathbf{G}$ be the base-B gadget matrix of dimension n over $\mathbb{Z}_q$.

For a public key $\mathsf{pk} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$ define the function $f_{\mathsf{pk}} : \{-C, \ldots, C\}^m \to \mathbb{Z}_p^n$ via

$$f_{\mathsf{pk}}(\mathbf{x}) = \lceil \mathbf{Ax} \rfloor_p .$$

The domain is $\mathcal{D} = \{-C, \ldots, C\}^m$ and the range is $\mathcal{R} = \mathbb{Z}_p^n$. We define all of the algorithms of the SLF as follows

- $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$: Choose a random $\overline{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times n'}$, $\mathbf{R} \leftarrow \chi^{n' \times m'}$, $\mathbf{E} \leftarrow \chi^{n \times m'}$ and set
$$\begin{aligned} \mathsf{pk} &= \mathbf{A} = [\overline{\mathbf{A}} \mid \overline{\mathbf{A}}\mathbf{R} + \mathbf{G} + \mathbf{E}] \\ \mathsf{td} &= (\mathbf{A}, \mathbf{R}, \mathbf{E}) \end{aligned}$$
- $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$: Choose a random $\overline{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times n'}$, $\mathbf{S} \leftarrow \mathbb{Z}_q^{n' \times m}$, $\mathbf{E} \leftarrow \chi^{n \times m}$ and set
$$\mathbf{A} = \overline{\mathbf{A}} \cdot \mathbf{S} + \mathbf{E}$$
- $\mathbf{x} \leftarrow D(\mathsf{pk})$: samples a uniformly random value $\mathbf{x} \leftarrow \mathsf{Sam}(1^\lambda)$.
- $\mathbf{x} \leftarrow \mathsf{Inv}_{\mathsf{td}}(\mathbf{y})$: Choose $\mathbf{v} \in \mathbb{Z}_q^n$ uniformly at random subject to $\lceil \mathbf{v} \rfloor_p = \mathbf{y}$ by choosing each coordinate uniformly from the appropriate interval. Let $\mathbf{A}' = \mathbf{A} - [\mathbf{0}|\mathbf{E}] = [\overline{\mathbf{A}} \mid \overline{\mathbf{A}}\mathbf{R} + \mathbf{G}]$ and output $\mathbf{x} \leftarrow \mathsf{SamPre}_{\mathbf{A}',\mathbf{R}}(\mathbf{v})$.

For concreteness, we set $B = 2^\lambda$, we set $\gamma = \lambda$ so that $q = B^\gamma = 2^{\lambda^2}$ and $p = 2^{\lambda^2 - 2\lambda}$. By the LWE assumption, there are some polynomials $n' = \mathrm{poly}(\lambda), \beta = \mathrm{poly}(\lambda)$ and a β-bounded distribution χ so that the $\mathsf{LWE}_{n',q,\chi}$ assumption hold. We set $n = n' \cdot \lambda$. Lastly, we choose $C = \lceil m^{2.5}\beta B \rceil$. This ensures that $\mathbf{x} \leftarrow \mathsf{Sam}(1^\lambda)$ outputs $\mathbf{x}$ such that $||\mathbf{x}||_\infty \leq C$.

Theorem 6.9. *The above construction is an (r, r', d, e, ℓ)-SLF* under the LWE assumption where*

$$\begin{aligned} & r = r' = n \log p = n(\lambda^2 - 2\lambda) = \lambda^2 n(1 - o(1)) \\ & d = \lceil m \log(2C + 1) \rceil = (n + \gamma n)(b + O(\log m + \log \beta) = \lambda^2 n(1 + o(1)) = (1 + o(1))r \\ & e = m \log B = (n + \gamma n)\lambda \geq \lambda^2 n \geq r \\ & \ell = n' \log q \leq (n/\lambda)\lambda^2 = o(r) \end{aligned}$$

In particular, it is a good SLF.*

Proof. We show each property of SLFs in turn below:

- Surjective Mode: Let $(\mathsf{pk} = \mathbf{A}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$ be some key pair sampled in surjective mode with $\mathbf{A} = [\overline{\mathbf{A}} \mid \overline{\mathbf{A}}\mathbf{R} + \mathbf{G} + \mathbf{E}]$ and let $\mathbf{A}' = \mathbf{A} - [\ \mathbf{0} \mid \mathbf{E}\ ] = [\overline{\mathbf{A}} \mid \overline{\mathbf{A}}\mathbf{R} + \mathbf{G}]$. By Lemma 6.8, we have
$$(\mathbf{x}, \mathbf{A}'\mathbf{x}) \overset{\mathrm{s}}{\approx} (\mathbf{x}', \mathbf{v})$$
where $\mathbf{x} \leftarrow \mathsf{Sam}(1^\lambda)$, $\mathbf{v} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{x}' \leftarrow \mathsf{SamPre}_{\mathbf{A}',\mathbf{R}}(\mathbf{v})$. This implies that
$$(\mathbf{x}, \lceil \mathbf{A}\mathbf{x} \rfloor_p) \equiv (\mathbf{x}, \lceil \mathbf{A}'\mathbf{x} + [\ \mathbf{0} \mid \mathbf{E}\ ]\mathbf{x} \rfloor_p) \overset{\mathrm{s}}{\approx} (\mathbf{x}', \lceil \mathbf{v} + [\ \mathbf{0} \mid \mathbf{E}\ ]\mathbf{x}' \rfloor_p) \overset{\mathrm{s}}{\approx} (\mathbf{x}', \lceil \mathbf{v} \rfloor_p) \equiv (\mathbf{x}', \mathbf{y})$$
where $\mathbf{y} \leftarrow \mathbb{Z}_p^n$. Here we use the fact that $\Pr[\lceil \mathbf{v} \rfloor_p \neq \lceil \mathbf{v} + \mathbf{e} \rfloor_p] = \mathrm{negl}(\lambda)$ where $\mathbf{e} = [\ \mathbf{0} \mid \mathbf{E}\ ]\mathbf{x}'$. This is because $\mathbf{v}$ is uniform over $\mathbb{Z}_q$ and $\mathbf{e}$ has norm $\tau = ||\mathbf{e}||_\infty \leq Cm'\beta = 2^{\lambda + O(\log \lambda)}$. Therefore the only way that $\lceil \mathbf{v} + \mathbf{e} \rfloor_p \neq \lceil \mathbf{v} \rfloor_p$ is if some coordinate of $\mathbf{v}$ lies within distance τ of the boundary of an interval of size q/p that gets rounded to the same value, but for any coordinate this happens with probability $(2\tau + 1)/(q/p) = 2^{\lambda + O(\log \lambda)}/2^{2\lambda} = \mathrm{negl}(\lambda)$.

- Lossy Mode: For $(\mathsf{pk} = \mathbf{A}) \leftarrow \mathsf{LossyGen}(1^\lambda)$ we have $\mathbf{A} = \overline{\mathbf{A}}\mathbf{S} + \mathbf{E}$. We define
$$\mathcal{L}_{\mathsf{pk}} = \left\{ \left\lceil \overline{\mathbf{A}}\mathbf{S}\mathbf{x} \right\rfloor_p \ :\ \mathbf{x} \in \{-C, \dots, C\}^m \right\} \subseteq \left\{ \left\lceil \overline{\mathbf{A}}\mathbf{y} \right\rfloor_p \ :\ \mathbf{y} \in \mathbb{Z}_q^{n'} \right\}$$
which is of size $|\mathcal{L}_{\mathsf{pk}}| \leq q^{n'} \leq 2^\ell$. For any $\mathbf{x} \in \{-C, \dots, C\}^m = \mathcal{D}_{\mathsf{pk}}$ and a random $\mathbf{A} \leftarrow \mathsf{LossyGen}(1^\lambda)$ we have
$$\Pr\left[\lceil \mathbf{A}\mathbf{x} \rfloor_p \notin \mathcal{L}_{\mathsf{pk}} \right] \leq \Pr\left[\left\lceil \overline{\mathbf{A}}\mathbf{S}\mathbf{x} + \mathbf{E}\mathbf{x} \right\rfloor_p \neq \left\lceil \overline{\mathbf{A}}\mathbf{S}\mathbf{x} \right\rfloor_p \right] \leq \mathrm{negl}(\lambda)$$
Here, we rely on the fact that $\tau = ||\mathbf{E}\mathbf{x}||_\infty \leq Cm\beta = 2^{\lambda + O(\log \lambda)}$. If $\mathbf{S}\mathbf{x} = \mathbf{0}$ then the above can never happen. Otherwise $\overline{\mathbf{A}}\mathbf{S}\mathbf{x}$ is uniformly random over the choice of $\overline{\mathbf{A}}$ and the above can only happen if some some coordinate of $\overline{\mathbf{A}}\mathbf{S}\mathbf{x}$ lies within distance τ of the boundary of an interval of size q/p that gets rounded to the same value, but for any coordinate this happens with probability $(2\tau + 1)/(q/p) = 2^{\lambda + O(\log \lambda)}/2^{2\lambda} = \mathrm{negl}(\lambda)$.

- Indistinguishability: We claim that the distributions of pk sampled from either $\mathsf{LossyGen}(1^\lambda)$ or $\mathsf{SurGen}(1^\lambda)$ are both computationally indistinguishable from the uniform distribution over $\mathbb{Z}_q^{n \times m}$, and therefore also indistinguishable from each other.

  Firstly for $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$ we have $\mathsf{pk} = [\overline{\mathbf{A}} \mid \overline{\mathbf{A}}\mathbf{R} + \mathbf{G} + \mathbf{E}]$. By thinking of the columns of $\mathbf{R}$ as LWE secrets that come from the error distribution and $\overline{\mathbf{A}}$ as the LWE coefficients, we get that $[\overline{\mathbf{A}}, \overline{\mathbf{A}}\mathbf{R} + \mathbf{E}]$ is computationally indistinguishable from uniform, which also shows that $\mathsf{pk} = [\overline{\mathbf{A}}, \overline{\mathbf{A}}\mathbf{R} + \mathbf{E}] + [\mathbf{0} \mid \mathbf{G}]$ is indistinguishable from uniform.

  Second for $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$ we have $\mathsf{pk} = [\overline{\mathbf{A}}\mathbf{S} + \mathbf{E}]$. By thinking of the columns of $\mathbf{S}$ as LWE secrets and $\overline{\mathbf{A}}$ as the LWE coefficients, it is immediate that pk is computationally indistinguishable from uniform.

Next, we prove that the domain and range satisfy the enhanced properties that make it an SLF*.

- Elements of $\mathcal{D}_{\mathsf{pk}} = \{-C, \ldots, C\}^m$ can be represented using $d = \lceil m \log(2C+1) \rceil$ bits.
- By Lemma 6.8, the min-entropy of the distribution $D(\mathsf{pk}) = \mathsf{Sam}(1^\lambda)$ is at least $m \log B$.
- The range $\mathcal{R}_{\mathsf{pk}} = \mathbb{Z}_p^n$ is a group under addition (or we can interpret $\mathcal{R}_{\mathsf{pk}} = \{0,1\}^{n(\lambda^2 - 2\lambda)}$ as a group under XOR).
- We can efficiently embed bit-string of length $r = n \log p = n(\lambda^2 - 2\lambda)$ as elements of $\mathcal{R}_{\mathsf{pk}} = \mathbb{Z}_p^n$.
- We can obliviously sample from $\mathcal{R}_{\mathsf{pk}} = \mathbb{Z}_p^n$ using r' random bits by equating elements of $\mathbb{Z}_p^n$ with bit-strings of length $r' = n \log p$.

□

7 Incompressible Encodings from SLFs

We now construct incompressible encodings in the CRS model. We will show that these encodings satisfy the stronger notion of HILL-entropic security (Definition 5.1 from Section 5), which implies incompressibility by Theorem 5.2. Furthermore, this means that we can also use this construction to get a composable incompressible encoding in the random oracle model using Theorem 5.3.

Construction. Given an (r, r', d, e, ℓ)-SLF*, we define the incompressible encoding scheme in the CRS model as follows:

- $\mathsf{crs} = (u_1, \ldots, u_{k'}) \leftarrow \{0,1\}^{r'(\lambda)\cdot k'}$, where $r'(\lambda)$ is the number of random bits needed to obliviously sample from the range of the SLF and $k' = \lceil k/r(\lambda) \rceil$.
- $\mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m)$: Choose $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$. Interpret $m = (m_1, \ldots, m_{k'}) \in \{0,1\}^{r(\lambda)\cdot k'}$. For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$, $y_i := \mathsf{sam}(\mathsf{pk}; u_i) \in \mathcal{R}_{\mathsf{pk}}$ and $c_i \leftarrow \mathsf{Inv}_{\mathsf{td}}(y_i + \hat{m}_i)$. Output $c = (\mathsf{pk}, c_1, \ldots, c_{k'})$.
- $\mathsf{Dec}_{\mathsf{crs}}(\ c = (\mathsf{pk}, c_1, \ldots, c_{k'}))$: For $i \in [k']$, let $y_i := \mathsf{sam}(\mathsf{pk}; u_i)$, $\hat{m}_i := f_{\mathsf{pk}}(c_i) - y_i$, $m_i := \mathsf{embed}^{-1}_{\mathsf{pk}}(\hat{m}_i)$. Output $(m_1, \ldots, m_{k'})$.

Theorem 7.1. *Assuming the existence of an (r, r', d, e, ℓ)-SLF*, the above construction yields an (α, β)-HILL-Entropic encoding scheme with selective security in the CRS model, where $\alpha(\lambda, k) = k'(\lambda)d(\lambda) + \mathrm{poly}(\lambda)$, $\beta(\lambda, k) = k'(\lambda)(e(\lambda) - \ell(\lambda))$ and the crs is of length $t(\lambda, k) = k'(\lambda)\cdot r'(\lambda)$ for $k'(\lambda) = \lceil \frac{k}{r(\lambda)} \rceil$. Furthermore, the encoding is locally decodable.*

In particular, any good SLF yields a good incompressible encoding that is locally decodable and achieves either:*

1. *Selective security in the CRS model, where the CRS is of length $t(\lambda, k) = k(1+o(1)) + \mathrm{poly}(\lambda)$.*
2. *Composable security in the Random Oracle model.*

Proof. We only prove the first part of the theorem and the second part ("in particular...") follows from Theorems 5.2 and 5.3.

The correctness of the scheme and the parameter α (length of encoding), t (length of CRS) are clear from the construction. To show β-HILL-Entropic security, define the procedure $(\mathsf{crs}, c) \leftarrow \mathsf{SimEnc}(1^\lambda, m)$ that, on input $m = (m_1, \ldots, m_{k'}) \in \{0,1\}^{r(\lambda)\cdot k'}$, samples $\mathsf{crs} = (u_1, \ldots, u_{k'})$ and $c = (\mathsf{pk}, c_1, \ldots, c_{k'})$ as follows:

- Choose $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$.
- For $i \in [k']$, choose $c_i \leftarrow D(\mathsf{pk})$.
- For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$. Let $y_i = f_{\mathsf{pk}}(c_i) - \hat{m}_i$ if $f_{\mathsf{pk}}(c_i) \in \mathcal{L}_{\mathsf{pk}}$ or else set $y_i = L - \hat{m}_i$ where L is some arbitrary element of $\mathcal{L}_{\mathsf{pk}}$. Let $u_i \leftarrow \mathsf{explain}_{\mathsf{pk}}(y_i)$.

First we show that SimEnc satisfies the entropy requirements. For any fixed m, pk, let $\mathsf{CRS} = (U_1, \ldots, U_{k'}), C = (\mathsf{pk}, C_1, \ldots, C_{k'})$ be random variables for the output $(\mathsf{crs}, c) \leftarrow \mathsf{SimEnc}(1^\lambda, m)$. Then

$$H_\infty(C|\mathsf{CRS}) \geq \sum_{i \in [k']} H_\infty(C_i|U_i) \geq k'(\lambda)(e(\lambda) - \ell(\lambda))$$

where the first inequality follows from the fact that (C_i, U_i) are k' independent random variables, and the second inequality follows since $U_i \in \mathcal{L}_{\mathsf{pk}} - \hat{m}_i$ is supported over a set of size $2^{\ell(\lambda)}$. This shows that the SimEnc satisfies the entropy requirement.

Let $m = \{m_\lambda\}$ be any ensemble of messages of length $|m_\lambda| = k(\lambda)$. We show that the two distributions of (crs, c) are indistinguishable for $\mathsf{crs} \leftarrow \{0,1\}^{t(\lambda,k)}, c \leftarrow \mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m)$ versus $(\mathsf{crs}, c) \leftarrow \mathsf{SimEnc}(1^\lambda, m)$. We do so via a sequence of hybrids.

Hybrid 0: This is the distribution of $\mathsf{crs} \leftarrow \{0,1\}^{t(\lambda,k)}, c \leftarrow \mathsf{Enc}_{\mathsf{crs}}(1^\lambda, m)$ where $m = (m_1, \ldots, m_{k'}) \in \{0,1\}^{r(\lambda)\cdot k'}$. The values are sampled as follows:

- For $i \in [k']$, choose $u_i \leftarrow \{0,1\}^{r'(\lambda)}$.
- Choose $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$.
- For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$, $y_i := \mathsf{sam}(\mathsf{pk}; u_i) \in \mathcal{R}_{\mathsf{pk}}$ and $c_i \leftarrow \mathsf{Inv}_{\mathsf{td}}(y_i + \hat{m}_i)$.
- Output $(\mathsf{crs} = (u_1, \ldots, u_{k'}), c = (\mathsf{pk}, c_1, \ldots, c_{k'})$.

Hybrid 1: Instead of choosing $u_i \leftarrow \{0,1\}^{r'(\lambda)}$ and setting $y_i := \mathsf{sam}(\mathsf{pk}; u_i) \in \mathcal{R}_{\mathsf{pk}}$, we now choose $y_i \leftarrow \mathcal{R}_{\mathsf{pk}}$ and set $u_i \leftarrow \mathsf{explain}_{\mathsf{pk}}(y_i)$. That is, hybrid 1 is defined as follows:

- Choose $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$.
- For $i \in [k']$, choose $y_i \leftarrow \mathcal{R}_{\mathsf{pk}}$ and $u_i \leftarrow \mathsf{explain}_{\mathsf{pk}}(y_i)$.
- For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$, $y_i := \mathsf{sam}(\mathsf{pk}; u_i) \in \mathcal{R}_{\mathsf{pk}}$ and $c_i \leftarrow \mathsf{Inv}_{\mathsf{td}}(y_i + \hat{m}_i)$.
- Output $(\mathsf{crs} = (u_1, \ldots, u_{k'}), c = (\mathsf{pk}, c_1, \ldots, c_{k'})$.

Hybrids 0 and 1 are statistically indistinguishable by the "oblivious sampling" property of the range $\mathcal{R}_{\mathsf{pk}}$ of the SLF*.

Hybrid 2: In hybrid 2, instead of choosing $y_i \leftarrow \mathcal{R}_{\mathsf{pk}}$ and setting $c_i \leftarrow \mathsf{Inv}_{\mathsf{td}}(y_i + \hat{m}_i)$, we choose $c_i \leftarrow D(\mathsf{pk})$ at random and set $y_i = f_{\mathsf{pk}}(c_i) - \hat{m}_i$. That is, hybrid 2 is defines as follows:

- Choose $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$.
- For $i \in [k']$, choose $c_i \leftarrow D(\mathsf{pk})$.
- For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$, $y_i = f_{\mathsf{pk}}(c_i) - \hat{m}_i$ and $u_i \leftarrow \mathsf{explain}_{\mathsf{pk}}(y_i)$.

- Output $(\mathsf{crs} = (u_1, \ldots, u_{k'}), c = (\mathsf{pk}, c_1, \ldots, c_{k'})$.

Hybrids 1 and 2 are statistically indistinguishable by the requirement on the surjective mode of the SLF*, which ensures that the two distributions on (y_i, c_i) in hybrids 1 and 2 are indistinguishable.

Hybrid 3: In hybrid 3, instead of choosing $(\mathsf{pk}, \mathsf{td}) \leftarrow \mathsf{SurGen}(1^\lambda)$ we choose $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$. Note that td is never used hybrid 2. That is, hybrid 3 is defined as follows.

- Choose $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$.
- For $i \in [k']$, choose $c_i \leftarrow D(\mathsf{pk})$.
- For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$, $y_i = f_{\mathsf{pk}}(c_i) - \hat{m}_i$ and $u_i \leftarrow \mathsf{explain}_{\mathsf{pk}}(y_i)$.
- Output $(\mathsf{crs} = (u_1, \ldots, u_{k'}), c = (\mathsf{pk}, c_1, \ldots, c_{k'})$.

Hybrids 2 and 3 are computationally indistinguishable by the indistinguishability requirement on the SLF.

Hybrid 4: In hybrid 4, if $f_{\mathsf{pk}}(c_i) \notin \mathcal{L}_{\mathsf{pk}}$ we set $y_i = L - \hat{m}_i$ where L is some arbitrary fixed element of $\mathcal{L}_{\mathsf{pk}}$. That is, hybrid 4 is defined as follows:

- Choose $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$.
- For $i \in [k']$, choose $c_i \leftarrow D(\mathsf{pk})$.
- For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$. Let $y_i = f_{\mathsf{pk}}(c_i) - \hat{m}_i$ if $f_{\mathsf{pk}}(c_i) \in \mathcal{L}_{\mathsf{pk}}$ or else set $y_i = L - \hat{m}_i$ where L is some arbitrary element of $\mathcal{L}_{\mathsf{pk}}$. Let $u_i \leftarrow \mathsf{explain}_{\mathsf{pk}}(y_i)$.
- For $i \in [k']$, let $\hat{m}_i = \mathsf{embed}_{\mathsf{pk}}(m_i) \in \mathcal{R}_{\mathsf{pk}}$, $y_i = f_{\mathsf{pk}}(c_i) - \hat{m}_i$ and $u_i \leftarrow \mathsf{explain}_{\mathsf{pk}}(y_i)$.
- Output $(\mathsf{crs} = (u_1, \ldots, u_{k'}), c = (\mathsf{pk}, c_1, \ldots, c_{k'})$.

Hybrids 3,4 are indistinguishable by the lossy mode property of the SLF*. In particular, for a random $\mathsf{pk} \leftarrow \mathsf{LossyGen}(1^\lambda)$ and $c_i \leftarrow D(\mathsf{pk})$, the probability that $f_{\mathsf{pk}}(c_i) \notin \mathcal{L}_{\mathsf{pk}}$ is negligible.

Hybrid 4 is exactly the distribution of $(\mathsf{crs}, c) \leftarrow \mathsf{SimEnc}(1^\lambda, m)$. Therefore, we have shown that the two target distributions in Hybrid 0 and Hybrid 4 are indeed indistinguishable, which concludes the proof. □

Corollary 7.2. *Under either the DCR or LWE assumptions, there exist good incompressible encodings that are locally decodable and achieve either:*

1. *Selective security in the CRS model, with a CRS of length $t(\lambda, k) = k(1 + o(1))$.*
2. *Composable security in the Random Oracle model.*

Furthermore the complexity of encoding/decoding is $k \cdot \mathrm{poly}(\lambda)$.

8 Black-Box Separations

Our results are of the same flavor as the work of Wichs [Wic13] and we follow their terminology and framework. They define the class of "single-stage game" assumptions (see Definition 3.1 of [Wic13]), which are modeled via a game between a (potentially inefficient) challenger and a stateful adversary. The assumption states that any polynomial-time (or sub-exponential time) attacker has at most a negligible (or inverse sub-exponential) success probability in winning the game. This captures essentially all standard assumptions used in cryptography, from the hardness of factoring to indistinguishability obfuscation (iO). However, the security definition of incompressible encodings is *not* a single-stage game since it involves two separate entities (the compressor $\mathcal{A}.\mathsf{Compress}$ and the decompresser $\mathcal{A}.\mathsf{Expand}$) who cannot fully share state between each other - the compressor is restricted in the number of bits it can pass to the decompressor.

We use the "simulatable attacker" framework (also called a meta-reduction) to prove our black box separations, and we give a high-level overview of it. See Definition 4.1 in [Wic13] for the formal definition. A simulatable attacker consists of a class of *inefficient* (exponential size) attackers $\mathcal{A}_h$ indexed by some $h \in \mathcal{H}_\lambda$, such that every attacker in the class breaks the intended security property. However we also design an efficient simulator $\mathcal{A}'$, such that one cannot (statistically) distinguish between black-box access to $\mathcal{A}_h$ for a random $h \leftarrow \mathcal{H}_\lambda$ versus $\mathcal{A}'$. Unlike the attackers $\mathcal{A}_h$, the simulator $\mathcal{A}'$ is a single fully stateful entity that can fully keep state between invocations. For example, in the case of incompressible encodings, the attacker $\mathcal{A}_h = (\mathcal{A}.\mathsf{Select}_h, \mathcal{A}.\mathsf{Compress}_h, \mathcal{A}.\mathsf{Expand}_h)$ consists of three separate stateless algorithms whereas $\mathcal{A}'$ consists of one monolithic and stateful entity with three interfaces ($\mathcal{A}'.\mathsf{Select}, \mathcal{A}'.\mathsf{Compress}, \mathcal{A}'.\mathsf{Expand}$) and can therefore "remember" any input given to $\mathcal{A}'.\mathsf{Select}$ and use it to answer calls to $\mathcal{A}'.\mathsf{Expand}$. Therefore $\mathcal{A}'$ is not a legal attacker against the incompressible encoding. However, if some reduction can break a single-stage game assumption given black-box access to $\mathcal{A}_h$ then it would also be able to do so given access to the efficient $\mathcal{A}'$ which means that the assumption is false. One important parameter of the above framework is ε which denotes the degree to which $\mathcal{A}_h$ and $\mathcal{A}'$ are indistinguishable; any algorithm making $q(\lambda)$ should not be able to distinguish $\mathcal{A}_h$ from $\mathcal{A}'$ with probability better than $O(q(\lambda))\varepsilon(\lambda)$. In the case of $\varepsilon(\lambda) = 0$ we say we have a perfectly simulatable attacker. In the case of a perfectly simulatable attacker, we can even rule out reductions from (sub-)exponential assumptions that may rely on complexity leveraging. See Theorem 4.2 of [Wic13] for the formal statement of this result.

A Separation in the Plain Model. We now prove a black box separation showing that any non-trivial incompressible encoding in the plain model cannot have a black box proof of security under any single-stage game based assumption.

Theorem 8.1. *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be any candidate encoding scheme with perfect correctness and* α*-expansion in the plain model. Then, for any "non-trivial"* $\beta \geq \alpha - k$ *there is a perfectly simulatable attacker against the* β*-incompressible security of the scheme. In particular, one cannot prove the* β*-incompressible security of the scheme under any single-stage game assumption (even assuming exponential security level, for an arbitrary exponential function) via a black-box reduction.*

Proof. We only prove the first part of the theorem and the "in particular" part follows from Theorem 4.2 of [Wic13].

Fix any values of λ, k. Let $m^* = \operatorname{argmin}_{m \in \{0,1\}^k} |\mathcal{C}_m|$ be the message that minimizes the size of the set $\mathcal{C}_m = \{c \ : \ \mathsf{Dec}(c) = m\}$ of codewords that decode to m. Since $\sum_m |\mathcal{C}_m| \leq 2^\alpha$ we know that

$|\mathcal{C}_{m^*}| \leq 2^{\alpha-k}$. Let $B = |\mathcal{C}_{m^*}|$ so that $\lceil \log B \rceil \leq \alpha - k$. Let the set $\mathcal{H}_\lambda$ consists of all permutations h over $[B]$. We define a class of inefficient attackers $\mathcal{A}_h = (\mathcal{A}.\mathsf{Select}_h, \mathcal{A}.\mathsf{Compress}_h, \mathcal{A}.\mathsf{Expand}_h)$ indexed by $h \in \mathcal{H}_\lambda$ as follows. Define $\mathcal{A}.\mathsf{Select}_h$ to output the message m^*. For any $c \in \mathcal{C}_{m*}$ define $i(c) \in [B]$ to output the index of c among the elements of $\mathcal{C}_{m^*}$ ordered lexiographically and define $i^{-1} : [B] \to \mathcal{C}_{m*}$ to be the inverse function. Note that i, i^{-1} may not be efficiently computable. We define $\mathcal{A}.\mathsf{Compress}_h(c)$ to output $w = h(i(c)) \in [B]$ if $c \in \mathcal{C}_{m*}$, and output 1 (or any other dummy value) otherwise. Note that $|w| \leq \alpha - k$. We define $\mathcal{A}.\mathsf{Expand}_h(w)$ to output $i^{-1}(h^{-1}(w))$. For any $h \in \mathcal{H}_\lambda$, the attacker $\mathcal{A}_h$ wins the β-incompressible security game with probability 1.

We now define an efficient (non-uniform) stateful simulator $\mathcal{A}' = (\mathcal{A}'.\mathsf{Select}, \mathcal{A}'.\mathsf{Compress}, \mathcal{A}'.\mathsf{Expand})$. The simulator has the message m^* defined as above hard-coded, along with $q(\lambda)$ distinct codewords $c_1, \ldots, c_q$ chosen randomly from the set $\mathcal{C}_{m^*}$. It initializes a table $\mathcal{T} \subseteq \mathcal{C}_m \times [B]$ which starts off empty. The algorithm $\mathcal{A}'.\mathsf{Select}(1^\lambda)$ outputs the message m^*. On any invocation of $\mathcal{A}'.\mathsf{Compress}(c)$, first check if $\mathsf{Dec}(c) = m^*$ and if not output 1. Else check if there is a tuple of the form (c, w) in the table and if so output w. Else pick a fresh $w \in [B]$ which is not yet in the table, and add the tuple (c, w) to the table and output w. On an invocation of $\mathcal{A}'.\mathsf{Expand}(w)$ with some input w, check if there is a tuple of the form (c, w) in the table and if so output c. Else, if this is the i'th invocation of $\mathcal{A}'.\mathsf{Expand}$ add the tuple (w, c_i) to the table and output c_i.

It is clear the the simulation above is perfect: in both cases, each invocation of $\mathcal{A}'.\mathsf{Compress}(c)$ with a fresh $c \in \mathcal{C}_m$ outputs a random fresh value $w \in [B]$ and each invocation of $\mathcal{A}'.\mathsf{Expand}(w)$ with a fresh $w \in [B]$ outputs a random fresh $c \in \mathcal{C}_m$. □

Implication for Short CRS and Composability. The above black box separation shows that one cannot prove the security of any non-trivial incompressible encoding in the plain model under any single-stage game based assumption via black-box reduction. We note two simple implications of this result.

Firstly, the result extends to give a separation for incompressible encodings (even with just selective security) in the CRS model, where *the CRS is short*. In particular, any such (α, β)-incompressible encoding with a CRS of size $t = t(\lambda, k)$ also yields an $(\alpha' = \alpha + t, \beta)$-incompressible encoding in the plain model: we simply choose a fresh random CRS during encoding time and append it to the codeword. Therefore, we get the same type of black-box seperation for any encoding in the CRS model as long as $\beta \geq \alpha + t - k$. In particular we have black-box separation for any encoding that has "good" expansion $\alpha(\lambda, k) \leq (1 + o(1))k + \mathrm{poly}(\lambda)$, 'good incompressibility" $\beta(\lambda, k) \geq (1 - o(1))k - \mathrm{poly}(\lambda)$ and a "short" CRS of size $t(\lambda, k) \leq o(k) + \mathrm{poly}(\lambda)$.

Secondly, the result also extends to give a separation for unbounded-time *composable* incompressible encodings in the CRS model with selective security, even if the CRS is arbitrarily large, as long as all encodings use the same CRS. This is because, given any such (α, β)-incompressible encoding scheme with composable security and with an arbitrarily large CRS of size $t(\lambda, k)$, for any polynomial $k'(\lambda)$ we would also get an $(\alpha' = (k/k')\alpha(k', \lambda), \beta' = (k/k')\beta(\lambda, k'))$-incompressible encoding with a CRS of size $t'(\lambda, k) = t(\lambda, k')$ by just dividing the data in to blocks of length k' and encoding each such block separately. Therefore if we had had such a composable encoding with "good" expansion $\alpha(\lambda, k') = (1 + o(1))k' + \mathrm{poly}(\lambda)$ and "good" incompressibility $\beta(\lambda, k') = (1 - o(1))k' - \mathrm{poly}(\lambda)$ but an arbitrarily large CRS, then, by choosing a sufficiently large polynomial $k'(\lambda)$, we would get an encoding scheme that maintains "good" expansion $\alpha'(\lambda, k) = (1 + o(1))k$ and "good" incompressiblity $\beta'(\lambda, k) = (1 - o(1))k$ but also has a small CRS of length $t'(\lambda, k) = t(\lambda, k') = \mathrm{poly}(\lambda)$. We already showed that the separation result extends to this case in the previous paragraph.

Separation in the CRS model with Adaptive Security. Lastly, we also give a (slightly modified) separation result for incompressible encodings with adaptive security in the CRS model, even without composition and even if the CRS is arbitrarily large.

Theorem 8.2. *Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be any candidate encoding scheme xin the CRS model with perfect correctness and* α*-expansion with a CRS of length* $t(\lambda, k)$*. Let* $\ell(\lambda)$ *be an arbitrary polynomial. Then, for any* $\beta \geq \alpha - k + 2\ell(\lambda)$ *there is a* $(\varepsilon(\lambda) = 2^{-\ell(\lambda))})$*-simulatable attacker against the adaptive* β*-incompressible security of the scheme. In particular, for* $\ell(\lambda) = \omega(\log \lambda)$ *one cannot prove the* β*-incompressible security of the scheme under any single-stage game assumption, even assuming* $2^{\ell(\lambda)}$*-security, via a black-box reduction.*

Proof. We only prove the first part of the theorem and the "in particular" part follows from Theorem 4.2 of [Wic13].

Fix any values of λ, k. Let the set $\mathcal{H}_\lambda$ consists of all tuples of functions $h = (h_1, h_2)$ where $h_1 : \{0,1\}^t \to \{0,1\}^k$ and $h_2\{0,1\}^t \times \{0,1\}^\beta \to \{0,1\}^\beta$ such that for each $\mathsf{crs} \in \{0,1\}^t$ the functions $\pi_{\mathsf{crs}} = h_2(\mathsf{crs}, \cdot)$ is a permutation. We define a class of inefficient attackers $\mathcal{A}_h = (\mathcal{A}.\mathsf{Select}_h, \mathcal{A}.\mathsf{Compress}_h, \mathcal{A}.\mathsf{Expand}_h)$ indexed by $h \in \mathcal{H}_\lambda$ as follows. Define $\mathcal{A}.\mathsf{Select}_h(\mathsf{crs})$ to output the message $m = h_1(\mathsf{crs})$. Define $\mathcal{A}.\mathsf{Compress}_h(\mathsf{crs}, c)$ as follows. Let $m = h_1(\mathsf{crs})$ and check that $\mathsf{Dec}(c) = m$; if this is not the case then output some dummy value and quit. Else let $\mathcal{C}_m = \{c : \mathsf{Dec}(c) = m\}$. If $|\mathcal{C}_m| \geq 2^{\alpha - k + \ell}$ then output some dummy value and quit. Else let $i \ leq 2^{\alpha - k + \ell}$ be the index of c in the set $\mathcal{C}_m$ ordered lexiographically. Output $w = \pi_{\mathsf{crs}}(i, 0^\ell)$. Define $\mathcal{A}.\mathsf{Expand}_h(\mathsf{crs}, w)$ as follows. Let $m = h_1(\mathsf{crs})$. Let $(i, \mu) = \pi_{\mathsf{crs}}^{-1}(w)$. If $\mu \neq 0^\ell$ then output a dummy value and quit. Else outputs the codeword c that has index i in the set $\mathcal{C}_m$ when ordered lexiographically.

We now define an efficient stateful simulator $\mathcal{A}' = (\mathcal{A}'.\mathsf{Select}, \mathcal{A}'.\mathsf{Compress}, \mathcal{A}'.\mathsf{Expand})$. The simulator initializes two tables $\mathcal{T}_1, \mathcal{T}_2$ which start off empty. The algorithm $\mathcal{A}'.\mathsf{Select}(\mathsf{crs})$ checks if there is a tuple of the form (crs, m) in $\mathcal{T}_1$ and if so it outputs m, else it chooses a random message $m \leftarrow \{0,1\}^k$, adds (crs, m) to the table $\mathcal{T}_1$ and outputs m. On any invocation of $\mathcal{A}'.\mathsf{Compress}(\mathsf{crs}, c)$, first verify that there is a tuple of the form $(\mathsf{crs}, m) \in \mathcal{T}_1$ and that $\mathsf{Dec}(c) = m$; if this is not the case then output a dummy value. THen check if there is a tuple of the form (crs, c, w) in $\mathcal{T}_2$ and if so then output w. Else pick a fresh uniformly random w that's not in the table $\mathcal{T}_2$ yet and add the tuple (crs, c, w) to $\mathcal{T}_2$ and output w. On an invocation of $\mathcal{A}'.\mathsf{Expand}(\mathsf{crs}, w)$ with some input w, check if there is a tuple of the form (crs, c, w) in the table $\mathcal{T}_2$ and if so output c, else output a dummy value.

We now argue that the attacker $\mathcal{A}_h$ for a random h and the simulator $\mathcal{A}'$ are indistinguishable. Let E_1 be the event that for some query to $\mathcal{A}.\mathsf{Select}_h(\mathsf{crs})$ the output m satisfies $|\mathcal{C}_m| \geq 2^{\alpha - k + \ell}$. Let E_2 be the event that a call to $\mathcal{A}'.\mathsf{Expand}(\mathsf{crs}, w)$ returns a non-dummy value for some w which was not previously returned by a query to $\mathcal{A}'.\mathsf{Compress}(\mathsf{crs}, c)$. If neither E_1 nor E_2 occur than the simulation is perfect. Therefore we are left to analyze the probability of E_1 and E_2. Firstly, we note that there can be at most $2^{k-\ell}$ messages m for which $\mathcal{C}_m \geq 2^{\alpha - k + \ell}$ and therefore $\Pr[|\mathcal{C}_m| \geq 2^{\alpha - k + \ell} : m \leftarrow \{0,1\}^k] \leq 2^{-\ell}$. This means that the probability of E_1 occuring if the distinguisher makes q queries is $q 2^{-\ell/2}$. On the other than E_2 only occurs if the distinguisher can guess a value (crs, w) such that $\pi_{\mathsf{crs}}^{-1}(w) = (\cdot, 0^\ell)$; the probability of this occuring if the attacker makes q queries $O(q)/2^\ell$. Therefore, the overall distinguishing advantage is bounded by $O(q)2^{-\ell}$, which completes the theorem. □

9 Big-Key Encryption in the Bounded Retrieval Model

A home computer is attacked, on average, within forty seconds of connecting to the Internet [RBC07]. Most of these attacks are "brute force" attempts to guess passwords; once an entry point is found, the attacker can run arbitrary code on the victim's computer, and can use this code to download account and password information. A similar situation occurs in a malware attack, where victims are tricked into running malicious code on their computers.

The best defense against these types of threat is *defense in depth*: try to prevent attacks, but also prepare to mitigate their effects should they succeed. This attack scenario motivates big-key cryptography in the *Bounded Retrieval Model* (BRM) [Dzi06, DLW06, CDD+07, ADW09, ADN+10, BKR16]. In this model, the adversary has an absolute bound ℓ on the number of bits it can exfiltrate from the compromised system.

The previous construction of public-key encryption (PKE) in the BRM required the ability to choose a highly structured secret key and public key together. In order to use incompressible encodings as the basis for leakage-resilience, we require encryption schemes that can accept an arbitrary (large) string as their secret key. Furthermore, we need security to hold as long as the key is incompressible.

9.1 Definitions

Definition 9.1 (β-incompressible distributions)**.** *Let $\mathcal{D}$ be a class of distributions over strings (parameterized by a security parameter λ). Consider the following "compression experiment"* $\mathsf{CompExp}_{\mathcal{A}}(1^\lambda)$ *with an adversary* $\mathcal{A} = (\mathcal{A}.\mathsf{Compress}, \mathcal{A}.\mathsf{Expand})$*:*

- $c \overset{\$}{\leftarrow} \mathcal{D}(\lambda)$.
- $w \leftarrow \mathcal{A}.\mathsf{Compress}(c)$.
- $c' \leftarrow \mathcal{A}.\mathsf{Expand}(w)$.
- *Output 1 if* $c = c'$ *and* $|w| \leq \beta(\lambda, |c|)$

We say that $\mathcal{D}$ is β-incompressible if for all PPT $\mathcal{A}$ we have $\Pr[\mathsf{CompExp}_{\mathcal{A}}(1^\lambda) = 1] = \mathrm{negl}(\lambda)$.

9.2 Encoding-Friendly Public-Key Encryption

An encoding-friendly encryption scheme is a tuple of algorithms $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$.

- The key generation algorithm accepts an arbitrary string sk as input, and returns $(pk, \hat{sk})$, where pk is the public key and $\hat{sk}$ is auxiliary state information.
- The encryption algorithm Enc accepts the public key pk and a message m as input, and returns a ciphertext c.
- The decryption algorithm Dec accepts $(pk, \hat{sk}, sk)$ and a ciphertext c as input, and returns a plaintext m.

In our case, we think of sk as potentially being many gigabytes or terabytes. We will require that pk as well as the run-time of Enc and Dec are bounded by $\mathrm{poly}(\lambda, \log|\mathsf{sk}|)$. In the case of Dec, this holds in the RAM model and means that decryption cannot even read the entire key sk. We will also require that $\hat{sk}$ is of size $o(|\mathsf{sk}|)$ so as not to add too much storage overhead.

9.2.1 The β-Leakage $\mathcal{D}$-IND-CPA Game

To define security (for encryption schemes) in this model, we modify the standard IND-CPA game in two ways. First, we let sk, the input to Keygen be chosen from a distribution $\mathcal{D}(\lambda)$, that may not be uniform. Second, we add an additional *leakage phase* to the IND-CPA game (this is the standard modification for leakage resilience).

Definition 9.2 (β-Leakage $\mathcal{D}$-IND-CPA Game)**.** *The game is played between a challenger and an adversary, composed of two algorithms:* $\mathcal{A} = (\mathcal{A}.\mathsf{Leak}, \mathcal{A}.\mathsf{CPA})$*:*

1. *The challenger selects a secret key from* $\mathcal{D}$*:* $sk \xleftarrow{\$} \mathcal{D}(\lambda)$*.*
2. *The challenger generates a public key and auxiliary:* $(pk, \hat{sk}) \leftarrow \mathsf{Keygen}(sk)$*.*
3. *The adversary executes* $\sigma \leftarrow \mathcal{A}.\mathsf{Leak}(pk, \hat{sk}, sk)$ *and returns the leakage string,* σ*.*
4. *The challenger selects a random bit* $b \xleftarrow{\$} \{0,1\}$*.*
5. *The adversary* $\mathcal{A}.\mathsf{CPA}(pk, \sigma)$ *selects two messages* m_0, m_1*, gets back* $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$*, and outputs a bit* b^**.*

The adversary wins the β*-leakage* $\mathcal{D}$*-IND-CPA game if* $|\sigma| \leq \beta(\lambda, |sk|)$ *and* $b^* = b$*.*

We say the adversary has advantage p in this game if it wins with probability $\frac{1}{2} + p$.

Definition 9.3 ($\mathcal{D}$-IND-CPA Security with β-Leakage)**.** *An encoding-friendly encryption scheme* $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *is* $\mathcal{D}$*-IND-CPA secure with* β*-Leakage if there exists a negligible function* ε *such that for every adversary* $\mathcal{A}$*, the advantage of* $\mathcal{A}$ *in the* β*-leakage* $\mathcal{D}$*-IND-CPA game is at most* $\varepsilon(\lambda)$*.*

9.3 Laconic OT

We build our encoding-friendly PKE scheme based on *Lacononic Oblivious Transfer (LOT)* [CDG+17]. For completeness, we present the definition of LOT here.

The following definition appears as Def. 3.1 in [CDG+17]:

Definition 9.4 (Laconic OT)**.** *A laconic OT (LOT) scheme syntactically consists of four algorithms:* crsGen*,* Hash*,* Send*, and* $\mathsf{Receive}$*.*

- $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda)$*. It takes as input the security parameter* λ *and outputs a common reference string.*
- $(\mathsf{digest}, \hat{D}) \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$*. It takes as input a common reference string* crs *and a database* $D \in \{0,1\}^*$ *and outputs a digest* digest *of the database and a state* $\hat{D}$*.*
- $e \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, L, m_0, m_1)$*. It takes as input a common reference string* crs*, a digest* digest*, a database location* $L \in \mathbb{N}$ *and two messages* m_0 *and* m_1 *of length* λ*, and outputs a ciphertext* e*.*
- $m \leftarrow \mathsf{Receive}^{D,\hat{D}}(\mathsf{crs}, e, L)$*. This is a RAM algorithm with random read access to* $(D, \hat{D})$*. It takes as input a common reference string* crs*, a ciphertext* e *and a database location* $L \in \mathbb{N}$*. It outputs a message* m*.*

We require the following properties of an LOT scheme $(\mathsf{crsGen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$*:*

- ***Correctness:*** *We require that it holds for any database D of size at most $M = \mathrm{poly}(\lambda)$ for any polynomial function $\mathrm{poly}(\cdot)$, any memory location $L \in [M]$ and any pair of messages $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$ that*

$$\Pr\left[m = m_{D[L]} \;\middle|\; \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda) \\ (\mathsf{digest}, \hat{D}) \leftarrow \mathsf{Hash}(\mathsf{crs}, D) \\ e \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, L, m_0, m_1) \\ m \leftarrow \mathsf{Receive}^{D,\hat{D}}(\mathsf{crs}, e, L) \end{array} \right] = 1 \ ,$$

  where the probability is taken over the random choices made by crsGen *and* Send*.*

- ***Sender Privacy Against Semi-Honest Receivers:*** *There exists a PPT simulator* LOTSim *such that the following holds. For any database D of size at most $M = \mathrm{poly}(\lambda)$, for any polynomial function $\mathrm{poly}(\cdot)$, any memory location $L \in [M]$ and any pair of messages $(m_0, m_1) \in \{0,1\}^\lambda \times \{0,1\}^\lambda$, let $\mathsf{crs} \leftarrow crsGen(1^\lambda)$ and $(\mathsf{digest}, \hat{D}) \leftarrow \mathsf{Hash}(\mathsf{crs}, D)$. Then it holds that*

$$(\mathsf{crs}, \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, L, m_0, m_1)) \overset{c}{\approx} (\mathsf{crs}, \mathsf{LOTSim}(\mathsf{crs}, D, L, m_{D[L]})) \ .$$

- ***Efficiency Requirement:*** *The length of* digest *is a fixed polynomial in λ independent of the size of the database; we will assume for simplicity that $|\mathsf{digest}| = \lambda$. Moreover, the algorithm* Hash *runs in time $|D| \cdot \mathrm{poly}(\log|D|, \lambda)$,* Send *and* $\mathsf{Receive}$ *run in time $\mathrm{poly}(\log|D|, \lambda)$. Lastly, the length of $\hat{D}$ is $o(|D|) + \mathrm{poly}(\lambda)$.*

The definition of [CDG+17] was not explicit about the additional state $\hat{D}$ having length $o(|D|) + \mathrm{poly}(\lambda)$. However, we note that this is achieved in their construction (with a small modification). Their construction starts with a 2-to-1 LOT scheme (where the hash is only 1/2 the size of the data) and uses it in a Merkle-Tree to get the above construction. The state $\hat{D}$ consists of the internal nodes of the Merkle Tree. In their construction, this would be of size $O(|D|)$. However, if we instead use a λ-to-1 LOT to build a Merkle Tree with degree λ then the size of the internal nodes in $\hat{D}$ will be $O(|D|/\lambda) = o(|D|)$, while preserving all other efficiency measures.

9.4 Encoding-Friendly PKE

9.4.1 Construction

Let $(\mathsf{crsGen}, \mathsf{Hash}, \mathsf{Send}, \mathsf{Receive})$ be an LOT scheme. Let $d = \mathrm{poly}(\lambda)$ be a parameter that we define later depending on the parameters of the encoding scheme. We construct an encryption scheme $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

- $\mathsf{Keygen}(sk)$:
  1. Executes $\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\lambda)$.
  2. Executes $(\mathsf{digest}, \hat{sk}) \leftarrow \mathsf{Hash}(\mathsf{crs}, sk)$.
  3. Outputs $pk = (\mathsf{crs}, \mathsf{digest})$ and $\hat{sk}$.

- We define a helper function: $\mathsf{keybits}(r)$ parses r as a set of indices $i_1, \ldots, i_d$ in the range $1, \ldots, |sk|$.
- $\mathsf{Enc}_{\mathsf{crs},\mathsf{digest}}(m; r)$:
  1. Parses r as $r_0, r_1, \ldots, r_{d-1}$.
  2. Let $m^{(1)}, \ldots, m^{(d)}$ be a d-out-of-d secret sharing of m (e.g., $m^{(1)} = r_1, \ldots, m^{(d-1)} = r_{d-1}, m^{(d)} = m \oplus r_1 \oplus \cdots \oplus r_{d-1}$).
  3. Computes, for $i \in [d]$
  $$c_i^0 \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{keybits}(r_0)_i, m^{(i)}, 0)$$
  and
  $$c_i^1 \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{keybits}(r_0)_i, 0, m^{(i)}) \ .$$
  4. Outputs $(r_0, c_1^0, c_1^1, \ldots, c_d^0, c_d^1)$.
- $\mathsf{Dec}_{\mathsf{crs},\hat{sk},sk}(r_0, c_1^0, c_1^1, \ldots, c_d^0, c_d^1)$:
  1. For $i \in [d]$:
     (a) Denote $k_i = sk_{\mathsf{keybits}(r_0)_i}$ the bit of sk indexed by $\mathsf{keybits}(r_0)_i$.
     (b) Computes
     $$m_i^{(i)} \leftarrow \mathsf{Receive}^{sk,\hat{sk}}(\mathsf{crs}, c^{k_i}, \mathsf{keybits}(r_0)_i) \ .$$
  2. Outputs $m^{(0)} \oplus \cdots \oplus m^{(d)}$ (the secret-share reconstruction).

9.5 Security Analysis

Our main theorem in this section is the security of our PKE scheme:

Theorem 9.5 (IND-CPA Security)**.** *If $\mathcal{D}$ is a class of β-incompressible distributions, then our PKE scheme is IND-CPA secure with $\mathcal{D}$-distributed keys and β' leakage, where $\beta'(\lambda, k) = \beta(\lambda, k) - o(k)$.*

The heart of our security reduction is a reconstruction procedure that, given an IND-CPA adversary $\mathcal{A} = (\mathcal{A}.\mathsf{Leak}, \mathcal{A}.\mathsf{CPA})$, can reconstruct most of the secret key (c.f. Algorithm 1). We use this to construct a compressor/decompressor for $\mathcal{D}$. The full compressor routine (c.f. Algorithm 2) consists of running the leakage adversary $\mathcal{A}.\mathsf{Leak}$, then using the core reconstruction procedure to simulate the *decompressor* in order to check which bits of secret key were successfully reconstructed. The final output of the compressor is consists of the leakage concatenated with the bits of the secret key that were not reconstructed.

The decompressor routine (c.f. Algorithm 3) runs the core reconstruction to recover most of the secret key, and uses the remaining bits from the compressor to recover the rest.

Informally, the reconstruction works in the following way: given "good" leakage (for which the adversary $\mathcal{A}.\mathsf{CPA}$ has a non-negligible distinguishing advantage), the repeatedly fixes the challenge randomness, then "tests" the adversary $\mathcal{A}.\mathsf{CPA}$ to see whether it still has a distinguishing advantage.

The challenge randomness includes the set of secret-key indices that are required to decrypt the message. In every case in which there is a distinguishing advantage, the reconstruction procedure

Algorithm 1 Core Reconstruction

```
 1: procedure $\textsc{Reconstruct}(p, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma)$
 2:                         ▷ Assumes adversary's advantage in Fixed-Prefix IND-CPA is at least $p$
 3:     Initialize $sk^* \leftarrow (\bot, \ldots, \bot)$                    ▷ For all $i$, $\mathsf{sk}^*_i \in \{0, 1, \bot\}$.
 4:     for all $i \in \{1, \ldots, N\}$ do
 5:         Let $r^{(i)} = (r^{(i)}_0, \ldots, r^{(i)}_d)$ be uniform random strings.
 6:         if $\textsc{IsGoodR}(p/2, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma, r^{(i)}, \bot)$ = true then
 7:                 ▷ $\mathcal{A}$ has advantage $\geq p/8$ w.h.p. in $(r_{\mathsf{pre}}, r^{(i)})$-Fixed-Response IND-CPA game
 8:             for all $j \in \{1, \ldots, d\}$ do
 9:                 if $\textsc{IsGoodR}(p/8, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma, r^{(i)}, j)$ = true then
10:                     Set $sk^*_{\mathsf{keybits}(r^{(i)})_j} \leftarrow 1$
11:                 else
12:                     Set $sk^*_{\mathsf{keybits}(r^{(i)})_j} \leftarrow 0$
13:                 end if
14:             end for
15:         end if
16:     end for
17:     return $sk^*$.
18: end procedure
19: function $\textsc{IsGoodR}(p, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma, r, j)$
20:     Denote $r'_{\mathsf{pre}} = (\mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma)$.
21:     Let $k = \lambda \cdot 12 \cdot \frac{2+p}{p^2}$.
22:     for all $i \in \{1, \ldots, k\}$ do
23:         if $j = \bot$ then
24:             Run steps 4 and 5 of the $(r'_{\mathsf{pre}}, r)$-Fixed-Response IND-CPA Game
                (c.f. Definition 9.9). Note that since the prefix of the game is
                already fixed, we don't need to recover $r_{\mathsf{pre}}$ to run steps 4 and 5 of
                the game; $r'_{\mathsf{pre}}$ is sufficient.
25:         else
26:             Run steps 4 and 5 of the $(r'_{\mathsf{pre}}, r, j)$-Bad IND-CPA Game (c.f. Definition 9.10)
27:         end if
28:     end for
29:     Let $\#win$ be the number of wins.
30:     if $\#win \geq (\frac{1}{2} + \frac{1}{2}p)k$ then
31:         return true
32:     else
33:         return false
34:     end if
35: end function
36: function $\textsc{GenTestEncryption}(\mathsf{crs}, \mathsf{digest}, r, j, m)$
37:     Parse $r = (r_0, \ldots, r_d)$
38:     Execute steps 1 to 3 of the encryption algorithm, computing $c^0_1, \ldots, c^0_d$ and $c^1_1, \ldots, c^1_d$
39:     Let $s^* \xleftarrow{\$} \{0, 1\}^\lambda$ be uniformly random and independent of all previous randomness.
40:     Compute $c^*_j \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{keybits}(r_0)_j, s^*, 0)$
41:     return $(r_0, c^0_1, c^1_1, \ldots, c^0_{j-1}, c^1_{j-1}, c^*_j, c^1_j, c^0_{j+1}, c^1_{j+1}, \ldots, c^0_d, c^1_d)$     ▷ Enc. with $c^*_j$ replacing $c^0_j$
42: end function
```

Algorithm 2 Compressor: $\mathcal{A}'.\mathsf{Compress}$

```
procedure $\mathcal{A}'.\mathsf{Compress}_p(sk)$
    for all $i \in \{1, \ldots, \lceil \lambda/p \rceil\}$ do
        $(\mathsf{crs}, \mathsf{digest}, \hat{sk}) \leftarrow \mathsf{Keygen}(sk)$
        $\sigma \leftarrow \mathcal{A}.\mathsf{Leak}(\mathsf{crs}, \mathsf{digest}, \hat{sk}, \mathsf{sk})$
        if IsGood$(p/2, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma) =$ true then
            Run $sk^* \leftarrow$ Reconstruct$(p/8, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma)$
            Denote $sk^-$ the concatenated bits of $sk$ for which $sk^*$ has $\perp$.
            Output $(\mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma, sk^-)$ and exit.
        end if
    end for
    Output $\perp$ and abort                                   ▷ We didn't find a good output
end procedure

function IsGood$(p, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma)$
                    ▷ Returns true w.h.p if adv. $\geq p$ and false w.h.p. if adv. $\leq p/4$
    Set $k = \lambda \cdot 12 \cdot \frac{2+p}{p^2}$.
    Run step 4 of the IND-CPA game $k$ times                   ▷ with i.i.d. randomness
    Let $\#win$ be the number of wins.
    if $\#win \geq (\frac{1}{2} + \frac{1}{2}p)k$ then
        return true
    else
        return false
    end if
end function
```

Algorithm 3 Decompressor: $\mathcal{A}'.\mathsf{Expand}$

```
procedure $\mathcal{A}'.\mathsf{Expand}_p(\mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma, sk^-)$
    Run $sk^* \leftarrow$ Reconstruct$(p/8, \mathsf{crs}, \mathsf{digest}, \hat{sk}, \sigma)$      ▷ Use the same randomness as $\mathcal{A}'.\mathsf{Compress}_p$
    Compute $sk$ by replacing all $\perp$ bits in $sk^*$ with the next bit of $sk^-$.
    Output $sk$.
end procedure
```

iterates over the associated indices, in each iteration replacing a corresponding share of the message with uniform randomness. It then tests whether the distinguishing advantage is maintained. This allows it to determine the value of the corresponding secret-key bit (since if it tampers with the LOT message with the correct bit, the honest-sender security of the LOT implies that the adversary cannot have any distinguishing advantage).

Before we dive into the formal proof, we will introduce some helpful definitions and claims.

9.5.1 Dispersers

In proving security of our scheme, we will need the following definition:

Definition 9.6 ((K, η)-disperser)**.** *A bipartite graph $G = (V_1, V_2, E)$ is a (K, η)-disperser iff for every set $S \subseteq V_1$ of size at least K the set of its neighbors $N(S) \subseteq V_2$ is of size at least $\eta \cdot |V_2|$.*

We use the following theorem, which shows that a random bipartite graph is a good disperser with overwhelming probability:

Theorem 9.7. *Let $G_\lambda = (V_1, V_2, E)$ be a sequence of random d-left-regular bipartite graphs, with $d \geq (2\lambda + 4)/\eta$, $|V_1| = N_\lambda$, $|V_2| = M_\lambda$, such that for each vertex $v \in V_1$, d neighbors in V_2 are chosen uniformly and independently at random.*

For every $\alpha > 16(1 - \eta)M/(dN) > 2^{-\lambda}$, the probability that G_λ is not a $(\alpha \cdot N, \eta)$ disperser is at most $2^{-\lambda}$.

The proof is essentially the same as in [Sip88] (with slightly different parameters). We include the proof in Section 10 for completeness.

9.5.2 IND-CPA Variants

In our analysis, we will consider three variants of the IND-CPA game. In the first variant, the random coins for a prefix of the standard IND-CPA game are fixed in advance.

Definition 9.8 (r_{pre}-Fixed-Prefix IND-CPA Game)**.** *The r_{pre}-Fixed-Prefix IND-CPA game behaves exactly like the standard IND-CPA game of Definition 9.2, except that the random coins for steps 1 to 3 are taken from r_{pre} (i.e., they are fixed in advance). In steps 4 and 5 fresh randomness is used.*

In the second variant, we not only fix the random coins for a prefix of the IND-CPA Game, but also the randomness used in the response to the CPA query (we assume w.l.o.g that $\mathcal{A}.\mathsf{CPA}$ makes only a single query):

Definition 9.9 ((r_{pre}, r^*)-Fixed-Response IND-CPA Game)**.** *The (r_{pre}, r^*)-Fixed-Response IND-CPA game behaves exactly like the r_{pre}-Fixed-Prefix IND-CPA game of Definition 9.8, except that the random coins used to select the indices $i_1, \ldots, i_d$ in the challenge ciphertext in step step 5 are taken from r^*. The rest of the randomness used in step 5 is fresh.*

Finally, in the third variant we fix the randomness for the prefix and CPA response, and also tamper with the response (making the encryption in some cases unreadable):

Definition 9.10 ($(r_{\mathsf{pre}}, r^*, j)$-Bad IND-CPA Game)**.** *The $(r_{\mathsf{pre}}, r^*, j)$-Bad IND-CPA game behaves exactly like the (r_{pre}, r^*)-Fixed-Response IND-CPA game of Definition 9.9, except that in the challenge ciphertext in step step 5 is created by having the c_0^j component encrypt a random and independent value (as in* $\textsc{GenTestEncryption}(\mathsf{crs}, \mathsf{digest}, r, j, m_b)$*; c.f. Line 36 in Algorithm 1) instead of the j'th share.*

Claim 9.10.1. *For all $p \in (0, \frac{1}{2})$ and $p' < p$, if $\mathcal{A}$ has advantage p in the IND-CPA Game, then when r_{pre} is chosen uniformly at random, the probability that $\mathcal{A}$ has advantage p' in the r_{pre}-Fixed-Prefix IND-CPA Game is at least $2(p - p')$.*

Proof. Let $R = (R_{\mathsf{pre}}, R_{\mathsf{post}})$ denote the randomness used in the IND-CPA Game, where R_{pre} is the randomness used in steps 1 to 3.

Denote $\mathcal{R}_{\mathsf{good}}$ be the set of prefixes such that for all $r_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}$ it holds that $\mathcal{A}$ has advantage p'.

Then

$$\begin{aligned}\frac{1}{2} + p &\leq \Pr[\mathcal{A} \text{ wins the IND-CPA Game}] \\ &= \Pr[R_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}] \Pr[\mathcal{A} \text{ wins the IND-CPA} | R_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}] \\ &\quad + (1 - \Pr[R_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}]) \Pr[\mathcal{A} \text{ wins the IND-CPA} | R_{\mathsf{pre}} \notin \mathcal{R}_{\mathsf{good}}]\end{aligned}$$

Since any probability is less than 1 and $\Pr[\mathcal{A} \text{ wins the IND-CPA} | R_{\mathsf{pre}} \notin \mathcal{R}_{\mathsf{good}}] \leq \frac{1}{2} + p'$,

$$\begin{aligned}&\leq \Pr[R_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}] + (1 - \Pr[R_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}])(\frac{1}{2} + p') \\ &= \Pr[R_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}](\frac{1}{2} - p') + (\frac{1}{2} + p')\end{aligned}$$

Rearranging, and since $p' < \frac{1}{2}$,

$$\Pr[R_{\mathsf{pre}} \in \mathcal{R}_{\mathsf{good}}] \geq \frac{p - p'}{\frac{1}{2} - p'} \geq 2(p - p')$$

□

The same proof essentially holds for the following Claim:

Claim 9.10.2. *For all $p \in (0, \frac{1}{2})$ and $p' < p$, if $\mathcal{A}$ has advantage p in the r_{pre}-Fixed-Prefix IND-CPA Game, then when r^* is chosen uniformly at random, the probability that $\mathcal{A}$ has advantage p' in the (r_{pre}, r^*)-Fixed-Response IND-CPA Game is at least $2(p - p')$.*

To show that the decompressor works, we prove that our "CPA tampering test" works:

Lemma 9.11. *For all $p \in (0, \frac{1}{2})$, the following holds with overwhelming probability over the choice of a random (r_{pre}, r^*). If $\mathcal{A}$ has advantage p in the (r_{pre}, r^*)-Fixed-Response IND-CPA game then for all $j \in [d]$:*

1. *If $sk_{\mathsf{keybits}(r^*)_j} = 0$ (where sk is the one chosen in the IND-CPA game prefix using r_{pre} as randomness) then $\mathcal{A}$'s advantage in the $(r_{\mathsf{pre}}, r^*, j)$-Bad IND-CPA game is negligible.*
2. *If $sk_{\mathsf{keybits}(r^*)_j} = 1$ then $\mathcal{A}$'s advantage in the $(r_{\mathsf{pre}}, r^*, j)$-Bad IND-CPA game is negligibly close to p.*

Proof. First, note that in the $(r_{\mathsf{pre}}, r^*, j)$-Bad IND-CPA game, the adversary plays the part of a semi-honest receiver in the LOT scheme. This is because the challenger honestly generates crs and digest, and executes Send honestly when answering a CPA query.

Thus, we can use the LOT scheme's *Sender Privacy Against Semi-Honest Receivers* property to claim that there exists a simulator LOTSim such that for every adversary $\mathcal{A}$, every key sk and every location $L \in [|sk|]$ $\mathcal{A}$ cannot distinguish between $(\mathsf{crs}, \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, L, m_0, m_1))$ and $(\mathsf{crs}, \mathsf{LOTSim}(\mathsf{crs}, sk, L, m_{sk_L}))$.

Recall that $\mathsf{Enc}(pk, m)$ computes a d-out-of-d sharing of m, and for each share $m^{(j)}$ sends two corresponding LOT messages. The j^{th} pair of messages is $c_j^0 \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{keybits}(r^*)_j, m, 0)$ and $c_j^1 \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{keybits}(r^*)_j, 0, m)$. In $\textsc{GenTestEncryption}()$, c_j^0 replaced with $c_j^* \leftarrow \mathsf{Send}(\mathsf{crs}, \mathsf{digest}, \mathsf{keybits}(r^*)_j, s^*, 0)$, and the response to the CPA query contains (c_j^*, c_j^1) in the j^{th} position.

To show Item 1, we note that in this case, $sk_{\mathsf{keybits}(r^*)_j} = 0$, hence (c_j^*, c_j^1) is indistinguishable from $(\mathsf{LOTSim}(\mathsf{crs}, sk, \mathsf{keybits}(r^*)_j, s^*), \mathsf{LOTSim}(\mathsf{crs}, sk, \mathsf{keybits}(r^*)_j, 0))$. Since the simulated pair contains no information about the j^{th} share of m_b, and Enc uses a d-out-of-d secret sharing, in the simulated case the adversary has no information about m_b, and hence must have negligible advantage in the CPA game.

Similarly, to show Item 2, observe that in this case $sk_{\mathsf{keybits}(r^*)_j} = 1$, hence (c_j^*, c_j^1) is indistinguishable from $(\mathsf{LOTSim}(\mathsf{crs}, sk, sk_{\mathsf{keybits}(r^*)_j}, 0), \mathsf{LOTSim}(\mathsf{crs}, sk, sk_{\mathsf{keybits}(r^*)_j}, m_b))$. For the same reason, this pair is indistinguishable from the pair (c_j^0, c_j^1), hence the adversary's advantage when given (c_j^*, c_j^1) must be statistically close to its advantage when given (c_j^0, c_j^1)—which, by the lemma's condition, is p. □

Lemma 9.12 (Correctness)**.** *The output of the decompressor will be correct except with negligible probability.*

Proof. The proof is in two parts:

1. First, we claim that the probability that $\mathcal{A}'.\mathsf{Compress}_p$ fails (outputs $\perp$) is negligible. Observe that in each iteration of the loop in Line 2 of Algorithm 2, $\mathcal{A}'.\mathsf{Compress}_p$ is choosing a random prefix r_{pre}, running the r_{pre}-Fixed-Prefix IND-CPA game (in $\textsc{IsGood}$), and estimating the adversary's advantage in that game by counting the number of wins.

   For any $p \in (0, \frac{1}{2})$, if the adversary has advantage at least p in the r_{pre}-Fixed-Prefix IND-CPA game, then the expected number of wins in $\textsc{IsGood}$ is $\mu = (\frac{1}{2} + p)k$. Since all the executions are independent, by the Chernoff bound, for all $\delta \in (0, 1)$ the probability that the actual number of wins is less than $(1 - \delta)\mu$ is bounded by $2e^{-\mu\delta^2/3}$. Setting $\delta = \frac{1}{2}\frac{p}{1+2p}$, such that $(1 - \delta)\mu = (\frac{1}{2} + p/2)$, and noting that with these settings $2e^{-\mu\delta^2/3} < 2e^{-\lambda}$ we conclude that the probability that $\textsc{IsGood}(p, \dots)$ returns **false** is at most $2e^{-\lambda}$ in this case.

   $\mathcal{A}'.\mathsf{Compress}_p$ calls $\textsc{IsGood}(p/2, \dots)$ By Claim 9.10.1, the probability that for a randomly-chosen prefix r_{pre} the adversary has advantage $p/2$ in the r_{pre}-Fixed-Prefix IND-CPA game is at least p, thus the probability that $\mathcal{A}'.\mathsf{Compress}$ reaches Line 11 (fails) is at most $2e^{-\lambda} + (1 - p)^{\lambda/p} < 3e^{-\lambda}$ hence is negligible in λ.

2. If $\mathcal{A}'.\mathsf{Compress}$ succeeds, we want to show that every bit computed by Algorithm 1 is correct. (If this is the case, then the decompression will also be correct, since $\mathcal{A}'.\mathsf{Compress}$ sends the bits that were not successfully computed by the reconstruction procedure.

   Observe that $\textsc{Reconstruct}(p, \dots)$ enters the loop at Line 8 only if the call to $\textsc{IsGoodR}(p/2, \dots)$ on Line 6 returns **true**. This call to $\textsc{IsGoodR}$ tests the advantage of the adversary in the $(r_{\mathsf{pre}}, r^{(i)})$-Fixed-Response IND-CPA game. By the same Chernoff argument as above, the probability that IsGoodR returns **true** if the adversary's advantage is less than $p/8$ is negligible.

Suppose that the adversary's advantage is at least $p/8$ in the $(r_{\mathsf{pre}}, r^{(i)})$-Fixed-Response IND-CPA game. Consider the call to IsGoodR in Line 9. In this case, IsGoodR tests the advantage of the adversary in the $(r_{\mathsf{pre}}, r^{(i)}, j)$-Bad IND-CPA game. By Lemma 9.11, if $sk_{\mathsf{keybits}(r^{(i)})_j} = 0$, the adversary's advantage in the $(r_{\mathsf{pre}}, r^{(i)}, j)$-Bad IND-CPA game is negligible (and in particular, less than $p/32$). Thus, the probability that IsGoodR returns **true** is negligible, and we will write the correct bit at Line 12. On the other hand, if $sk_{\mathsf{keybits}(r^{(i)})_j} = 1$, by Lemma 9.11 the adversary's advantage is at least $p/8$, hence IsGoodR returns **true** except with negligible probability, and we will write the correct bit at Line 10. Since these are the only two lines at which bits of sk^* are written, we will always write the correct bits.

□

Lemma 9.13 (Compression)**.** *If $\mathcal{A}$.Leak outputs state of size $\beta(\lambda, |sk|)$, then the output of $\mathcal{A}'$.Compress is of size $\beta(\lambda, |sk|) + o(|sk|)$ except with negligible probability.*

Proof. Consider the following bipartite graph $G = (V_1, V_2, E)$: V_1 contains a vertex for every choice of random challenge $r_0^{(i)}$ used Reconstruct(). V_2 contains a vertex for every bit of sk. There is an edge between $r_0^{(i)} \in V_1$ and $j \in V_2$ iff $j \in \mathsf{keybits}(r_0^{(i)})$.

Call a vertex in V_1 *good* if it passed the IsGoodR test on Line 6 of Algorithm 1. Observer that if a vertex v is good, then Algorithm 1 will set all d bits corresponding to $N(v)$ in sk^*.

Note that IsGoodR is called with the parameter p passed to Algorithm 1, which in turn is called with $p/8$, where p is the advantage of the adversary in the IND-CPA game.

Reconstruct$(p/8, \dots)$ is only called if IsGood$(p/2, \dots)$ returns **true** (in Line 5 of Algorithm 2), which, except with negligible probability, implies that the adversary has advantage at least $p/8$ in the r_{pre}-Fixed-Prefix IND-CPA game.

Given that the adversary has advantage at least $p/8$ in the r_{pre}-Fixed-Prefix IND-CPA game, by Claim 9.10.2 the probability that it has advantage $p/16$ in the (r_{pre}, r^*)-Fixed-Response IND-CPA game (for a randomly chosen r^*) is at least $p/8$. Thus, with overwhelming probability there will be at least $(p/16) \cdot N$ good vertices.

We set N such that $N > 256(1-\eta)|sk|/(pd)$ and $d > (2\lambda + 4)/\eta$, satisfying the conditions of Theorem 9.7. Thus, except with negligible probability, G is a $((p/16) \cdot N, \eta)$-disperser. This means that the neighbors of "good" vertices comprise at least $(1-\eta)|sk|$ bits of the secret key. Thus, $|sk^-| < \eta|sk| = o(|sk|)$. □

9.5.3 Proof of Theorem 9.5

Proof. All that remains is to put the pieces together. By Lemma 9.12, all bits filled in sk^* are the correct bits of sk. The compressed key consists of $(\mathsf{crs}, \mathsf{digest}, \sigma, |sk^-|)$. The crs size depends only on λ, digest can be made $o(|sk|)$, and by Lemma 9.13, $|sk^-| = o(|sk|)$. By our assumption about the adversary, $|\sigma| < \beta'(\lambda, |sk|)$. Thus, the total compressed size is at most $\beta'(\lambda, |sk|) + o(|sk|)$.

□

References

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, Heidelberg, August 2009.

[ADN$^+$10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 113–134. Springer, Heidelberg, May / June 2010.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, Heidelberg, August 2009.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996.

[AKPS19] Benedikt Auerbach, Eike Kiltz, Bertram Poettering, and Stefan Schoenen. Lossy trapdoor permutations with improved lossiness. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 230–250. Springer, Heidelberg, March 2019.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 57–74. Springer, Heidelberg, August 2013.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, Heidelberg, August 2018.

[BKR16] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 373–402. Springer, Heidelberg, August 2016.

[BLP$^+$13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584. ACM Press, June 2013.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018,* *Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564. Springer, Heidelberg, April / May 2018.

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *Approximation, randomization, and combinatorial optimization*, volume 2764 of *Lecture Notes in Comput. Sci.*, pages 200–215. Springer, Berlin, 2003.

[CDD$^+$07] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 479–498. Springer, Heidelberg, February 2007.

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65. Springer, Heidelberg, August 2017.

[CFMJ19] Ethan Cecchetti, Ben Fisch, Ian Miers, and Ari Juels. PIEs: Public incompressible encodings for decentralized storage. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 1351–1367. ACM Press, November 2019.

[CGM19] Yilei Chen, Nicholas Genise, and Pratyay Mukherjee. Approximate trapdoors for lattices and smaller hash-and-sign signatures. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 3–32. Springer, Heidelberg, December 2019.

[CHK$^+$16] Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, January 2016.

[DGO19] Ivan Damgård, Chaya Ganesh, and Claudio Orlandi. Proofs of replicated storage without timing assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 355–380. Springer, Heidelberg, August 2019.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, Heidelberg, February 2001.

[DJMW12] Yevgeniy Dodis, Abhishek Jain, Tal Moran, and Daniel Wichs. Counterexamples to hardness amplification beyond negligible. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 476–493. Springer, Heidelberg, March 2012.

[DLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 225–244. Springer, Heidelberg, March 2006.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DVW09] Yevgeniy Dodis, Salil P. Vadhan, and Daniel Wichs. Proofs of retrievability via hardness amplification. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 109–127. Springer, Heidelberg, March 2009.

[Dzi06] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 207–224. Springer, Heidelberg, March 2006.

[FGK$^+$13] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. *Journal of Cryptology*, 26(1):39–74, January 2013.

[Fis18] Ben Fisch. Tight proofs of space and replication. Cryptology ePrint Archive, Report 2018/702, 2018. https://eprint.iacr.org/2018/702.

[GKPV10] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 230–240. Tsinghua University Press, January 2010.

[GLW20] Rachit Garg, George Lu, and Brent Waters. New techniques in replica encodings with client setup. Cryptology ePrint Archive, Report 2020/617, 2020. https://eprint.iacr.org/2020/617.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, June 2011.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 169–186. Springer, Heidelberg, May 2007.

[JK07] Ari Juels and Burton S. Kaliski Jr. Pors: proofs of retrievability for large files. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007: 14th Conference on Computer and Communications Security*, pages 584–597. ACM Press, October 2007.

[JP11] Abhishek Jain and Krzysztof Pietrzak. Parallel repetition for leakage resilience amplification revisited. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 58–69. Springer, Heidelberg, March 2011.

[KOS10] Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 295–313. Springer, Heidelberg, August 2010.

[Lab17a] Protocol Labs. Filecoin: A decentralized storage network. 2017.

[Lab17b] Protocol Labs. Proof of replication. 2017.

[LW10] Allison B. Lewko and Brent Waters. On the insecurity of parallel repetition for leakage resilience. In *51st Annual Symposium on Foundations of Computer Science*, pages 521–530. IEEE Computer Society Press, October 2010.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718. Springer, Heidelberg, April 2012.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, Heidelberg, February 2004.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, Heidelberg, August 2003.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, Heidelberg, May 1999.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342. ACM Press, May / June 2009.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166. Springer, Heidelberg, March 2006.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196. ACM Press, May 2008.

[RBC07] Daniel Ramsbrock, Robin Berthier, and Michel Cukier. Profiling attacker behavior following SSH compromises. In *The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, 25-28 June 2007, Edinburgh, UK, Proceedings*, pages 119–124. IEEE Computer Society, 2007.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, Heidelberg, May 2011.

[Sip88] Michael Sipser. Expanders, randomness, or time versus space. *J. Comput. Syst. Sci.*, 36(3):379–383, 1988.

[SW08] Hovav Shacham and Brent Waters. Compact proofs of retrievability. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 90–107. Springer, Heidelberg, December 2008.

[vJO$^+$12] Marten van Dijk, Ari Juels, Alina Oprea, Ronald L. Rivest, Emil Stefanov, and Nikos Triandopoulos. Hourglass schemes: how to prove that cloud files are encrypted. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012: 19th Conference on Computer and Communications Security*, pages 265–280. ACM Press, October 2012.

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS 2013: 4th Innovations in Theoretical Computer Science*, pages 111–126. Association for Computing Machinery, January 2013.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, November 1982.

[Zha16] Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 479–508. Springer, Heidelberg, August 2016.

10 Random Graphs Are Good Dispersers

We recall Theorem 9.7:

Theorem 9.7. *Let $G_\lambda = (V_1, V_2, E)$ be a sequence of random d-left-regular bipartite graphs, with $d \geq (2\lambda + 4)/\eta$, $|V_1| = N_\lambda$, $|V_2| = M_\lambda$, such that for each vertex $v \in V_1$, d neighbors in V_2 are chosen uniformly and independently at random.*

For every $\alpha > 16(1-\eta)M/(dN) > 2^{-\lambda}$, the probability that G_λ is not a $(\alpha \cdot N, \eta)$ disperser is at most $2^{-\lambda}$.

Proof. G is a $(\alpha \cdot N, \eta)$-disperser if for every pair of sets $S \subset V_1$ and $T \subset V_2$ such that $|S| = \alpha N$ and $|T| = \eta M$ it holds that S has at least one neighbor in T.

For a specific S and T, we can bound the probability that this does not occur by observing that each of the vertices in S has d independently-selected neighbors in V_2. Denote $z = 2\lambda + 4$ (so $d = z/\eta$). The probability that no vertex in S has a neighbor in T is at bounded by

$$\left(1 - \frac{|T|}{M}\right)^{d|S|} \leq (1-\eta)^{d\alpha N} \leq ((1-\eta)^{1/\eta})^{z\alpha N} \leq e^{-z\alpha N} \leq 2^{-z\alpha N}$$

To complete the proof, we take the union bound over all applicable pairs of sets S, T:

$$\Pr\left[G \text{ is not a disperser}\right] \leq \binom{N}{\alpha N}\binom{M}{\eta M} \cdot 2^{-z\alpha N}$$

Using that, for all $n > 1, \varepsilon \in (0,1)$: $\binom{n}{\varepsilon n} \leq 2^{nH(\varepsilon)}$,

$$\leq 2^{N \cdot H(\alpha) + M - z\alpha N}$$

Using $H(x) \leq x \log(e/x)$,

$$\leq 2^{-\alpha N(z - \log(e/\alpha)) + M}$$

$$\leq 2^{-\alpha N(z - \lambda - 2) + M}$$

Defining $w = \lambda + 2 = z - \lambda - 2$,

$$\leq 2^{-\alpha N w + M}$$

Using $\alpha > 16(1-\eta)M/dN$,

$$\leq 2^{-16(1-\eta)Mw/d + M}$$

Since $w/d = \eta/2$,

$$\leq 2^{-16(1-\eta)M\eta/2 + M} = 2^{-M(16(1-\eta)\eta/2 - 1)}$$

Since $(1-\eta)\eta \leq 1/4$,

$$\leq 2^{-M} \leq 2^{-\lambda}$$

To bound the probability that Consider a set $S \subset V_1$ of size of size ηM □