

Leakage-Resilient Incompressible Cryptography: Constructions and Barriers*

Kaartik Bhushan[†] Rishab Goyal[‡] Venkata Koppula[§]
Varun Narayanan[¶] Manoj Prabhakaran^{||} Mahesh Sreekumar Rajasree^{**}

Abstract

We introduce Leakage-Resilient Incompressible cryptography, which simultaneously addresses two variants of side-channel attacks that have been tackled in theoretical cryptography. Leakage-resilience seeks to provide security against an adversary who learns a part of the secret-key and the entire ciphertext or signature; conversely, incompressible cryptography provides security against an adversary who learns the entire secret-key, but only a part of the ciphertext or signature. However, constructions in either of these security models can fail against an attack in the other model. In this work, we define a new model of security that subsumes both leakage-resilient cryptography and incompressible cryptography, and we present several non-trivial positive and negative results.

On the positive side, first we present a transformation from incompressible symmetric-key encryption (SKE) to leakage-resilient incompressible SKE in the information-theoretic setting. Next, as one of our main results, we construct a leakage-resilient incompressible public-key encryption (PKE), combining an incompressible SKE and a new primitive that we call leakage-resilient non-committing key encapsulation mechanism (LR-NC-KEM). While an information-theoretic incompressible SKE suitable for use in both these constructions already exists in the literature (Dziembowski, CRYPTO 2006), we present a new construction achieving ciphertext-rate $\frac{1}{2}$ and compression-rate $\frac{1}{2}$, using an appropriate notion of invertible extractors; this leads to corresponding improvements in the final parameters we obtain in these constructions. We also construct a leakage-resilient incompressible signature scheme in the classical random oracle model, assuming (DCR or LWE) and the Decisional Linear assumption, with signature-rate and compression-rate $1 - o(1)$ and leakage-rate 1.

On the negative side, we show barriers to significantly improving the parameters we obtain, by showing impossibility of basing the security of such improved schemes on blackbox reductions.

Apart from the general framework and the specific results we obtain, some of the intermediate tools that we define and instantiate, like LR-NC-KEM and invertible extractors, may be of independent interest.

*A preliminary version of this work appeared in the proceedings of ASIACRYPT 2024. This manuscript is the full version.

[†]CNRS, IRIF, Université Paris Cité. Work done while the author was at IIT Bombay. kbhushan@irif.fr

[‡]University of Wisconsin-Madison. rishab@cs.wisc.edu

[§]Indian Institute of Technology Delhi. kvenkata@cse.iitd.ac.in

[¶]Chennai Mathematical Institute. Work done while the author was at UCLA. varunnkv@gmail.com

^{||}Indian Institute of Technology Bombay. mp@cse.iitb.ac.in

^{**}University of Warsaw. Work done while the author was at IIT Delhi. srmahesh1994@gmail.com

Contents

1	Introduction	3
1.1	Our Contributions	5
1.1.1	Lower Bounds:	5
1.1.2	Positive Results	6
2	Technical Overview	6
2.1	Impossibility Results	6
2.2	Positive Results	9
3	Preliminaries	12
3.1	Extractors	12
3.2	Incompressible Public Key Encryption	13
3.3	Incompressible Secret Key Encryption	15
3.4	Incompressible Encoding	17
3.5	Garbling Scheme	19
3.6	Reconstructable Probabilistic Encoding	19
3.7	Hash Proof Systems	21
3.8	Leakage-Resilient Signatures	24
3.9	Simulatable Attacks	26
3.10	Probability	28
4	Leakage-Resilient Incompressible and Non-Committing Schemes: Definitions	29
4.1	Leakage-Resilient Incompressible Encryption	29
4.2	Leakage-Resilient Non-Committing Key Encapsulation Mechanism	31
4.3	Leakage-Resilient Incompressible Signatures	33
5	Leakage-Resilient Incompressible SKE against Unbounded Adversaries	34
5.1	Information-Theoretic Incompressible SKE with Ciphertext-Rate $\frac{1}{2}$	35
5.2	Generic Transformation	38
6	Leakage-Resilient NC-KEM from Hash Proof Systems	45
7	Leakage-Resilient Incompressible PKE	54
8	Leakage-Resilient Incompressible PKE from Leakage-Resilient Incompressible SKE	59
9	Bit-Leakage-Resilient Incompressible PKE from any Incompressible PKE	65
10	Leakage-Resilient Incompressible Signatures	68
11	Impossibility Results for Rate-$1 - o(1)$ Incompressible Encryption Schemes	71
11.1	Impossibility Result for Incompressible PKE with Short Keys	72
11.2	Impossibility Result for Leakage-Resilient Incompressible SKE	78
12	Declaration on the Use of AI Tools	86

1 Introduction

Traditionally, cryptography relies on each honest user having an ideally secure computation platform. If a user’s computing system does not provide an ideally secure platform, the user is essentially considered to be corrupted. However, this severely limits the security guarantees cryptography can provide, as computing systems are seldom ideal. In particular, existence of various practical side-channel and key-infiltration attacks routinely undermine security of numerous deployed cryptosystems.

From a theoretical perspective, this seems to be a hopelessly bleak situation, especially when all the users rely on non-ideal computing systems. However, a long line of research in cryptography has studied the problem of providing full-fledged security guarantees even in non-ideal settings. Our work is inspired by two such popular formulations considered today – leakage resilience, and incompressible cryptography.

Leakage-resilient cryptography. As demonstrated by several practical side-channel attacks (such as [29] presenting one of the earlier ones), an adversary may be able to extract partial information (*leakage*) about the secret keys of an, otherwise, honest user. Since traditional cryptographic schemes provide *no guarantees at all* once such leakage takes place, the area of *leakage-resilient cryptography* [1] was developed to protect from such side-channel attacks. It is a very rich area of cryptography that has received a lot of attention over the last two decades.

One of the prominent examples is (leakage-resilient) public-key encryption. It allows an adversary to learn the public-key pk as well as a leakage on the secret key sk in the form of $f_{\text{leak}}(\text{sk})$. Here f_{leak} is any adversarially chosen function, with the only constraint on f_{leak} that its output length is limited. This ensures that leakage-resilient public-key encryption is not trivially impossible, as we still require semantic security of the encryption system to hold even in presence of the secret key leakage.

Incompressible cryptography. Side-channel attacks are not the only means by which an adversary can learn some information about the secret keys. Key infiltration attacks, where the adversary is able to learn the entire secret key, are another common security threat. In such scenarios, it is unclear whether one could ask for any meaningful notion of security, but prior works have demonstrated that one may still ask for security of past communication. Consider encryption¹ as an example: suppose an adversary who knows that, at some point in the future, it will learn the secret key. Such an adversary can store as much of the prior (encrypted) communication as possible, and plan to decrypt the stored ciphertexts once it has the secret key. A mitigating factor in such a scenario is the vast amounts of ciphertexts such an adversary will need to store (anticipating key compromise in the future). Alternately, consider a scenario where the ciphertexts need to be “exfiltrated” from a partially secure system via malware, large amounts of communication could be detected and truncated. However, if the adversary can compress the ciphertexts (and later recover some digest of the message when the key is obtained), then even with the limits on storage or exfiltration bandwidth, secrecy of the messages will be compromised.

Again, this concern can be addressed by enhancing the cryptographic guarantees of the cryptosystems used. An early variant of this problem was introduced by Rivest [32] (in a somewhat different context); it was explicitly studied by Dziembowzki [17] in the secret-key setting, and more recently studied in the public-key setting by Guan, Wichs, and Zhandry [24]. Guan *et al.* called this *incompressible encryption*, and we adopt the same terminology in this work. Informally, incompressible public-key encryption states that no attacker can learn any information about plaintext from any arbitrarily compressed ciphertext, even if it obtains the secret key, as long as the ciphertext is compressed before receiving the key. That is, even given secret key sk , an attacker cannot learn anything about an encrypted message from $f_{\text{comp}}(\text{ct})$,

¹In the case of key-agreement, the threat due to compromise of long-terms keys is addressed using ephemeral keys, to yield forward security. However, this does not apply to scenarios like encrypted e-mails and disk encryption, involving a long-term decryption key.

where f_{comp} is any adversarially chosen function with a limited output size (which disallows storing the entire ciphertext).

Leakage resilience and incompressibility are duals. leakage-resilient encryption and incompressible encryption can be viewed as *duals* of each other. In the former, the adversary gets partial information about the secret key and the full ciphertext. In the latter, it gets the full secret key, but can only retain partial information about the ciphertext. Indeed, connections between these two domains have been studied, and techniques from one domain have led to progress in the other. However, quite apart from the similarity of techniques, we note that the attacks underlying both these settings can simultaneously take place: *When the ciphertexts are being compressed for storage/exfiltration, leakage on the key may already be available to the adversary.* We ask:

Can such combined threats be tackled and if so, how efficiently?

To address this, we initiate the study of *leakage-resilient incompressible cryptography*. While the central focus of this work is on encryption (public-key and symmetric-key), we also study signatures in this setting. We briefly recall that a leakage-resilient signature scheme says that an adversary must not be able to forge a signature, even after receiving leakage on the signing key. While incompressible signatures prevent replay attacks, where the attacker must not be able to forge a signature, even if it receives an arbitrary (yet bounded) leakage on any number of signatures. We formally explain this further in the technical overview.

Measuring efficacy of leakage-resilient incompressible cryptography. Inspired by prior works [8, 16, 1, 14, 31, 13, 21, 5, 24, 7, 25, 22] on leakage resilience and incompressible cryptography, we consider a natural combination of known efficiency metrics for cryptosystems in the leakage-resilient incompressible setting.

Let us start by reviewing the commonly used efficiency metrics for encryption. For leakage-resilient encryption, a common approach is to measure the amount of leakage that is permitted without losing semantic security. It is formally captured as the *leakage-rate*, which is the ratio of total number of bits that can be leaked divided by the length, i.e. $\frac{|f_{\text{leak}}(\text{sk})|}{|\text{sk}|}$. For incompressible encryption, a well-known metric is the ciphertext-rate. It measures the maximum amount of message that can be packed in a single ciphertext, and formally defined as the ratio of the message length and the ciphertext size, i.e. $\frac{|m|}{|\text{ct}|}$. In our work, we also study another efficiency metric for incompressible encryption, that we call *compression-rate*. It is defined as the ratio of the total number of bits about a ciphertext that can be stored divided by the ciphertext length, i.e. $\frac{|f_{\text{comp}}(\text{ct})|}{|\text{ct}|}$.

There are three main metrics – ciphertext-rate, compression-rate, and leakage-rate – that we want to optimize while designing leakage-resilient incompressible encryption schemes. Ideally, we want to design encryption schemes where all three rates are close to 1: this would give a scheme that is as efficient as possible in terms of the ciphertext size, while semantic security will hold even when an attacker learns the secret key and the ciphertext almost entirely. Thus, the central question that we study in this work is:

Q1. *Can we build leakage-resilient incompressible PKE with ciphertext-rate, compression-rate, and leakage-rate approaching 1?*

We already have encryption schemes that can tolerate as much as $1 - o(1)$ fraction of the leakage (i.e., leakage-rate is almost 1) [1, 5, 26, 6, 12]. Moreover, we also have incompressible encryption schemes with ciphertext-rate and compression-rate of 1 [24, 7, 25, 22]. Thus, it is quite natural to ask if we can design a unified encryption scheme that achieves all these rates simultaneously.

Furthermore, towards our broader goal of studying other forms of leakage-resilient incompressible cryptosystems, we also study a similar question for signature schemes:

Q2. *Can we build leakage-resilient incompressible signature schemes with $1 - o(1)$ leakage-rate and optimal communication (message + signature size)?*

1.1 Our Contributions

Our first main contribution is a pair of black-box barriers. We show that leakage-resilient incompressible SKE cannot have ciphertext-rate and compression-rate close to 1 while leaving too little residual uncertainty in the secret key after leakage, if security is to be proved via a black-box reduction to a secure cryptographic game [20, 35]. We also prove a short-key barrier for ordinary incompressible PKE with ciphertext-rate and compression-rate close to 1. In light of these barriers, we ask a natural question: *What ciphertext-rate, compression-rate, and leakage-rate can we provably achieve for leakage-resilient incompressible encryption?* Our next contribution is to design encryption schemes from standard cryptographic assumptions that achieve balanced rates. As an additional contribution, we construct a leakage-resilient incompressible signature scheme in the classical random oracle model under (DCR or LWE) and the Decisional Linear assumption. Below we summarize our main results in two main categories – lower bounds and positive results.

1.1.1 Lower Bounds:

We first prove that leakage-resilient incompressible SKE cannot simultaneously have ciphertext-rate and compression-rate close to 1 while leaving too little residual uncertainty in the secret key after leakage. Formally, we prove the following.

Theorem 1. *Let $\text{SKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ be an SKE scheme with deterministic decryption and almost perfect correctness. Fix a parameter regime with message length $\ell_{\text{msg}} = \omega(\log \lambda)$, ciphertext length ℓ_{ct} , secret key length ℓ_{sk} , state size ℓ_{st} , and leakage length ℓ_{leak} . Suppose the leakage class contains an efficiently computable function $f : SK \rightarrow \{0, 1\}^{\ell_{\text{leak}}}$ whose fibers have size at most $2^{\ell_{\text{sk}} - \ell_{\text{leak}}}$. If there exists $\eta = \omega(\log \lambda)$ such that*

$$(\ell_{\text{sk}} - \ell_{\text{leak}}) + \ell_{\text{ct}} - \ell_{\text{st}} \leq \ell_{\text{msg}} - \eta,$$

then there is no secure cryptographic game $\mathcal{G}$ such that a black-box reduction proves leakage-resilient incompressible SKE security of SKE from the security of $\mathcal{G}$.

Thus, in the rate $1 - o(1)$ ciphertext and compression regime, any scheme avoiding this black-box barrier must retain roughly ℓ_{msg} bits of uncertainty in the secret key after leakage. We note that if we restrict the adversary’s leakage functions to bit-leakage functions, then one can still achieve ciphertext-rate and compression-rate $1 - o(1)$ with leakage-rate $1/2 - o(1)$; we discuss this formally in Section 9. The proof of the above theorem is complemented by a new lower bound that we prove for incompressible PKE.

Incompressible encryption with ciphertext-rate 1. As mentioned earlier, we have constructions for incompressible PKE with ciphertext-rate 1 [24, 7, 22]. However, all constructions with provable security in the standard model have *large* secret keys, i.e. the key size grows with the message size. Guan *et al.* [24] conjectured that any ciphertext-rate $1 - o(1)$ and compression-rate $1 - o(1)$ incompressible PKE scheme with ‘short’ key size cannot be proven secure using a black-box reduction to a secure cryptographic game [35]. In this work, we formally prove this conjecture. Formally, we prove:

Theorem 2. *Let $\text{PKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ be an incompressible PKE scheme for arbitrary message length ℓ_{msg} with ciphertext length $\ell_{\text{ct}} = \ell_{\text{msg}} + o(\ell_{\text{msg}})$, secret key length ℓ_{sk} , state size $\ell_{\text{st}} = \ell_{\text{msg}} - o(\ell_{\text{msg}})$ and possesses almost perfect correctness and deterministic decryption. Then, if $\ell_{\text{sk}} \leq \ell_{\text{msg}}(1 - \Omega(1))$, there is no secure cryptographic game $\mathcal{G}$ such that there is a black-box reduction that derives the security of PKE from the security of $\mathcal{G}$.*

More generally, our theorem rules out any parameter regime satisfying $\ell_{\text{sk}} + \ell_{\text{ct}} - \ell_{\text{st}} \leq \ell_{\text{msg}} - \omega(\log \lambda)$. For example, if $\ell_{\text{ct}} = (1 + \epsilon)\ell_{\text{msg}} + o(\ell_{\text{msg}})$ and $\ell_{\text{st}} = \ell_{\text{msg}} - o(\ell_{\text{msg}})$, then avoiding the barrier requires $\ell_{\text{sk}} \geq (1 - \epsilon)\ell_{\text{msg}} - o(\ell_{\text{msg}})$.

1.1.2 Positive Results

We show that any information-theoretic incompressible secret-key encryption scheme with uniform key generation can be transformed into an information-theoretic leakage-resilient incompressible SKE scheme. Naturally, given the information-theoretic promise, this scheme has large secret keys. When instantiated using the incompressible SKE scheme of Dziembowski [17], the resulting leakage-resilient incompressible SKE scheme has ciphertext-rate $\frac{1}{3}$ and inherits the compression-rate of the underlying scheme.

Further, we present an alternate information-theoretic incompressible SKE scheme with ciphertext-rate and compression-rate both equal to $\frac{1}{2}$ (but using a larger secret key). Using this to instantiate our leakage-resilient incompressible SKE, we obtain the same two rates together with leakage-rate $1 - o(1)$. Thus, we achieve the balanced rate pair $(\frac{1}{2}, \frac{1}{2})$ in the information-theoretic setting.

Theorem 3. *There exists a leakage-resilient incompressible SKE scheme with negligible correctness error, ciphertext-rate $\frac{1}{2}$, compression-rate $\frac{1}{2}$, leakage-rate $1 - o(1)$, and security against unbounded adversaries for output-length-bounded leakage functions.*

Next, we move on to leakage-resilient incompressible public-key encryption. Following a recent result by Goyal *et al.* [22], which lifts incompressible SKE to incompressible PKE, we note that we can generically lift any leakage-resilient incompressible SKE to leakage-resilient incompressible PKE. But this yields a $o(1)$ ciphertext-rate. In this work, we optimize this significantly to design a leakage-resilient incompressible PKE scheme with ciphertext-rate $\frac{1}{2}$.

Theorem 4. *Assuming the hardness of the DDH problem, there exists a leakage-resilient incompressible PKE scheme with ciphertext-rate $\frac{1}{2}$, compression-rate $\frac{1}{2}$, and leakage-rate $1 - o(1)$.*

For this construction, we introduce a new cryptographic primitive called a *leakage-resilient non-committing key encapsulation mechanism* (LR-NC-KEM), which may be of independent interest (see Section 6). All of our results surrounding incompressible encryption, together with existing results are included in Table 1 and Table 2.

Leakage-resilient incompressible signatures. Finally, we construct a leakage-resilient incompressible signature scheme in the classical random oracle model (see Section 10), following the template of Guan *et al.* [24]. We combine the adaptive random oracle encoding of Moran and Wichs [30] with the full-state leakage-resilient signatures of Boyle, Segev, and Wichs [4]. Formally, we show that:

Theorem 5. *Assume (DCR or LWE) and the Decisional Linear assumption. Then there exists a leakage-resilient incompressible signature scheme in the classical random oracle model with signature-rate and compression-rate $1 - o(1)$ and leakage-rate 1, against arbitrary efficient leakage functions of the signing key with access to the same oracle.*

2 Technical Overview

This section provides a high-level summary of the paper’s results, both positive and negative.

2.1 Impossibility Results

In this work, we introduce two black-box separation results, demonstrating that specific incompressible encryption primitives cannot be proven secure under any standard assumptions using a black-box reduction. We accomplish this by leveraging the notion of simulatable attacks introduced by Wichs [35]. A simulatable attack against the security of a primitive involves two entities: an inefficient adversary $\mathcal{A}$ that achieves a high probability of winning the security game, and an efficient simulator Sim capable of mimicking $\mathcal{A}$. Notably, we require that Sim can convincingly emulate $\mathcal{A}$ for any efficient reduction R with oracle access to Sim ,

Primitive	Ciphertext-Rate	Compression-Rate	secret key Size	Note
IT. SKE	$\frac{1}{3}$	1	$\ell_{\text{msg}} + d$	[17]
IT. SKE	$\frac{1}{2}$	$\frac{1}{2}$	$4 \cdot \max(\ell_{\text{st}}, \ell_{\text{msg}}) + O(\lambda)$	Section 5.1
SKE	$1 - o(1)$	$1 - o(1)$	$\max(\ell_{\text{st}}, \ell_{\text{msg}})(1 + o(1))$	[7]
PKE	$1 - o(1)$	$1 - o(1)$	$\ell_{\text{msg}}(1 + o(1)) \cdot \text{poly}(\lambda)$	[7]
PKE	$1 - o(1)$	$1 - o(1)$	$\ell_{\text{msg}}(1 - \Omega(1))$	Section 11.1 (Barrier)

Table 1: Constructions for Regular Incompressible Encryption. In the first row, $d = O(\log(\ell_{\text{msg}} + \lambda)(\log(\ell_{\text{st}} + \ell_{\text{msg}} + \lambda) + \lambda))$; see Theorem 20. In the information-theoretic setting, we show an incompressible SKE scheme with balanced ciphertext-rate and compression-rate $\frac{1}{2}$. In the computational setting, prior works showed incompressible SKE/PKE with ciphertext-rate 1 and large secret keys. In the last row, we record the black-box barrier for rate-1 incompressible PKE with short secret keys.

Primitive	Ciphertext-Rate	Compression-Rate	Leakage	Leakage Function	Reference
IT. SKE	$\frac{1}{2}$	$\frac{1}{2}$	$1 - o(1)$	General	Section 5
PKE	$\frac{1}{2}$	$\frac{1}{2}$	$1 - o(1)$	General	Section 7
SKE	$1 - o(1)$	$1 - o(1)$	small residual	General	Section 11.2 (Barrier)
PKE/SKE	$1 - o(1)$	$1 - o(1)$	$\frac{1}{2} - o(1)$	Bits	Section 9

Table 2: Constructions for Leakage-Resilient Incompressible Encryption. In the first two rows, we discuss parameters that can be achieved for leakage-resilient incompressible encryption. The third row records the SKE black-box barrier when leakage leaves too little residual key uncertainty. Finally, in the last row, we present parameters achievable when leakage is restricted to subsets of bits of the secret key.

i.e., $|\Pr[R^{\mathcal{A}}(1^\lambda) = 1] - \Pr[R^{\text{Sim}}(1^\lambda) = 1]| = \text{negl}(\lambda)$.

Simulatable Attack on Ciphertext Rate-1 Incompressible PKE with Small Keys.

Assuming that the reader is familiar with the definition of the incompressible security game (see Section 3.2), we describe our inefficient reduction $\mathcal{A}$, which has two hardcoded functions, g and h , as follows:

1. On input a public key pk , $\mathcal{A}_0$ returns a pair of messages $g(\text{pk}) = (m_0, m_1)$ along with $\text{aux} = g(\text{pk})$.
2. On input a ciphertext ct^* and auxiliary information aux , $\mathcal{A}_1$ returns $h(\text{ct}^*)$.
3. On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, $\mathcal{A}_2$ checks whether $\text{aux} = g(\text{pk})$. If not, it returns $\perp$. Otherwise, it computes $C = \{\text{ct} \mid h(\text{ct}) = \text{st}\}$ and $M = \{m \mid m = \text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct}), \text{ct} \in C\}$. It checks whether $\{m_0, m_1\} \cap M = \emptyset$ or $\{m_0, m_1\} \subseteq M$. If so, it returns $\perp$. Otherwise, if $m_0 \in M$, it returns 0. Else, it returns 1.

Now, let's understand why $\mathcal{A}$ wins the incompressible security game with high probability. In the security game, the challenger faithfully generates the challenge ciphertext ct^* , an encryption of m_b where b is the random challenge bit. This implies that $m_b \in M$, and the only case where $\mathcal{A}$ loses is if $m_{1-b} \in M$. Since h is a random function, the preimage $h^{-1}(\text{st})$ has size close to its average value, namely about $2^{|\text{ct}^*| - |\text{st}|}$. In the rate $1 - o(1)$ regime, this is only $2^{o(|m|)}$, so the corresponding decryptable set is tiny compared to the full message space. Hence the probability that the other random message also lands in M is about $2^{-(|m| - o(|m|))}$, which is negligible.

Now, let's present our simulator Sim which maintains two databases Q_0, Q_1 :

1. On input a public key pk , Sim first checks whether an entry $(\text{pk}, (m_0, m_1))$ is in Q_0 . If so, it outputs $(m_0, m_1, \text{aux} = (m_0, m_1))$. If not, it randomly generates a pair of messages (m_0, m_1) , stores $(\text{pk}, (m_0, m_1))$ in Q_0 , and outputs $(m_0, m_1, \text{aux} = (m_0, m_1))$.

2. On input a ciphertext ct^* and auxiliary information aux , Sim checks whether an entry (ct^*, st) is in Q_1 . If so, it outputs st . If not, it randomly generates st , stores (ct^*, st) in Q_1 , and outputs st .
3. On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, Sim checks whether $(\text{pk}, (m_0, m_1))$ is present in Q_0 . If not, it returns $\perp$. Otherwise, it computes $C = \{\text{ct} \mid (\text{ct}, \text{st}) \in Q_1\}$ and $M = \{m \mid m = \text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct}), \text{ct} \in C\}$. It checks whether $\{m_0, m_1\} \cap M = \emptyset$ or $\{m_0, m_1\} \subseteq M$. If so, it returns $\perp$. Otherwise, if $m_0 \in M$, it returns 0. Else, it returns 1.

The differences between $\mathcal{A}$ and Sim are that Sim generates random responses to the queries it receives and maintains a log of them. In the third type of query, it searches its database Q_1 to construct the set C . These differences lead to the following bad events that the reduction may use to distinguish $\mathcal{A}$ from Sim :

1. On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, Sim may have early aborted because $(\text{pk}, (m_0, m_1))$ wasn't present in Q_0 . However, for the same input, $\mathcal{A}$ must have returned a bit b . This event happens when the reduction was able to guess the value of $h(\text{ct}^*)$ without querying its oracle. Using simple guessing probability, one can show that this event is negligible in the size of the message.
2. On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, Sim may have early aborted because $(m_0, m_1) \cap M = \emptyset$ whereas, for the same input, $\mathcal{A}$ must have returned a bit b . This event happens when the reduction was able to guess the value of $h(\text{ct}^*)$ without querying its oracle such that ct^* decrypts to either m_0 or m_1 . Simple guessing probability cannot be applied directly because the reduction may cleverly guess a *bad* secret key that decrypts the majority of the ciphertext to one of m_0 or m_1 . However, using the fact that the secret key size is small, we argue that the number of such messages is negligible for every secret key. So, for randomly chosen messages m_0 and m_1 , the messages are *good*, in the sense that around $2^{|\text{ct}|}/2^{|m|}$ ciphertexts decrypt to m_0 or m_1 . With this guarantee, we can apply simple guessing probability bounds to show that this event happening is negligible.
3. On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, Sim may have returned a bit b whereas, for the same input, $\mathcal{A}$ must have returned $\perp$. We show that this scenario is very similar to the previous one and again use guessing probability bounds.

Since the above events occur with negligible probability, there does not exist any efficient reduction that can distinguish between $\mathcal{A}$ and Sim . Therefore, we obtain a simulatable attack against the security of a rate 1 incompressible scheme with small keys.

Simulatable Attack on Rate 1 Leakage-Resilient Incompressible SKE. Assuming that the reader is familiar with the definition of the leakage-resilient incompressible security game (see Section 4.1), we describe our inefficient reduction $\mathcal{A}$, which has three hardcoded functions, f , g , and h , as follows:

1. On input a secret key sk , $\mathcal{A}_0$ returns the truncation of sk to $|\text{sk}| - o(|\text{sk}|)$ bits.
2. On input a leakage value z , $\mathcal{A}_1$ returns a pair of messages $(m_0, m_1) = g(z)$ and auxiliary information $\text{aux} = (z, m_0, m_1)$.
3. On input a ciphertext ct^* and auxiliary information aux , $\mathcal{A}_2$ returns $h(\text{ct}^*)$.
4. On input $(\text{sk}, \text{aux} = (z, m_0, m_1), \text{st})$, $\mathcal{A}_3$ checks whether $z = f(\text{sk})$ and $(m_0, m_1) = g(z)$. If not, it returns $\perp$. Otherwise, it computes $C = \{\text{ct} \mid h(\text{ct}) = \text{st}\}$ and $M = \{m \mid m = \text{IncSKE.Dec}(\text{sk}, \text{ct}), \text{ct} \in C\}$. It checks whether $\{m_0, m_1\} \cap M = \emptyset$ or $\{m_0, m_1\} \subseteq M$. If so, it returns $\perp$. Otherwise, if $m_0 \in M$, it returns 0. Else, it returns 1.

Similar to the argument presented in the previous section, we can show that $\mathcal{A}$ wins the leakage-resilient incompressible game with very high probability. Now, let's present our simulator Sim which maintains two databases Q_1, Q_2 :

1. On input a secret key sk , Sim returns the same fixed leakage value $z = f(\text{sk})$ as the inefficient adversary.
2. On input a leakage value z , Sim first checks whether an entry $(z, (m_0, m_1))$ is in Q_1 . If so, it outputs $(m_0, m_1, \text{aux} = (z, m_0, m_1))$. If not, it randomly generates a pair of messages (m_0, m_1) , stores $(z, (m_0, m_1))$ in Q_1 , and outputs $(m_0, m_1, \text{aux} = (z, m_0, m_1))$.
3. On input a ciphertext ct^*, aux , Sim checks whether an entry (ct^*, st) is in Q_2 . If so, it outputs st . If not, it randomly generates st , stores (ct^*, st) in Q_2 , and outputs st .
4. On input $(\text{sk}, \text{aux} = (z, m_0, m_1), \text{st})$, Sim checks whether $z = f(\text{sk})$ and $(z, (m_0, m_1))$ is present in Q_1 . If not, it returns $\perp$. Otherwise, it computes $C = \{\text{ct} \mid (\text{ct}, \text{st}) \in Q_2\}$ and $M = \{m \mid m = \text{IncSKE.Dec}(\text{sk}, \text{ct}), \text{ct} \in C\}$. It checks whether $\{m_0, m_1\} \cap M = \emptyset$ or $\{m_0, m_1\} \subseteq M$. If so, it returns $\perp$. Otherwise, if $m_0 \in M$, it returns 0. Else, it returns 1.

Similar to the previous section, note that the distinctions between $\mathcal{A}$ and Sim lie in the fact that Sim generates random responses to the queries it receives and keeps a log of them. In the fourth type of query, it searches its database Q_2 to construct the set C . These differences give rise to bad events similar to those in the previous section. We address most of them in a similar manner, with the exception that, rather than relying on small key information, we leverage the advantage of substantial leakage of the secret key to argue indistinguishability.

Comparison with [35]. While our techniques for the above separation results are inspired by prior work [35], there are key technical differences which required new non-trivial ideas. At a high level, one of the main differences is that prior separation results [35] were for cryptosystems such as one-way functions or entropy-generators/condensers. Abstractly, all these cryptographic objects were just different types of one-way functions with some special properties (such as correlated input security, leakage-resilience etc). Therefore, the objects were defined w.r.t. only a ‘public’ function key, and there was not a concept of a ‘secret’ key. Informally, thus meant that the primitives for which impossibilities existed did not have any real *correctness* property, but it was efficiency/succinctness and security. This meant that to show the separation result, they just had to design an inefficient (simulatable) attacker on the security of the system by leveraging the desired level of efficiency/succinctness for the object.

Unfortunately, it is unclear how this simple strategy can be directly extended to the encryption setting, since now there is also a ‘secret’ key which brings in a new (and much needed) correctness guarantee. Thus, our inefficient (simulatable) attacker must find a way to leverage both efficiency/succinctness and correctness. Clearly, if we do not use correctness while designing our inefficient (simulatable) attack, then we cannot prove the separation result since one could trivially design an “*incorrect*” encryption scheme that achieves optimal efficiency/succinctness as well as security. Thus, one of our key technical contributions in coming up with the above separation result is that we had to find a new approach to also leverage correctness along with succinctness. We believe our approach of generalizing the [35] black-box separation techniques to simultaneously leverage correctness and succinctness together to break security will have more applications in the future.

2.2 Positive Results

In this work, we present the first leakage-resilient incompressible encryption schemes in both the secret-key and public-key settings, as well as a construction of leakage-resilient incompressible signatures in the classical random oracle model under (DCR or LWE) and the Decisional Linear assumption.

LR Incompressible SKE secure against unbounded adversaries. At a high level, our leakage-resilient incompressible SKE scheme uses an incompressible SKE scheme for encryption, but applies some preprocessing of the secret key to protect against leakage. Specifically, in our scheme, the secret key is represented by a large random string sk . To encrypt a message m , a

random seed s is sampled, and a nearly uniform string inc.sk is derived by applying the extractor to sk using the seed s . Subsequently, inc.sk serves as the secret key for the incompressible SKE scheme, which encrypts m to produce inc.ct . The final ciphertext is $(s, \text{inc.ct})$. To decrypt, inc.sk is recomputed from sk and s , and then the decoder of the incompressible SKE scheme is invoked on $(\text{inc.sk}, \text{inc.ct})$.

To provide the intuition behind the leakage-resilient incompressible security of our scheme, we draw a parallel to the role of the extractor in leakage-resilient encryption schemes: even if a substantial portion of the secret key sk is leaked to the adversary, if sk retains sufficient entropy after the leakage $f(\text{sk})$, the resulting string inc.sk obtained by extraction is statistically close to uniform. Consequently, an unbounded adversary cannot learn any information about the secret key used in the inner incompressible SKE scheme. We should, however, exercise caution while arguing security against an adversary who eventually observes the leakage $f(\text{sk})$, the compression of the ciphertext chosen adaptively based on that leakage, and the secret key sk . For instance, in the security game, an adversary can force the challenger to commit to the secret key and message using hash functions of its choice while computing the leakage and compression, making it impossible to efficiently and retroactively resample a secret key consistent with those commitments. In our proof, we carry out this resampling using a computationally unbounded program to make the secret key independent of the inner incompressible SKE scheme, and then reduce the security to the security of the inner incompressible SKE scheme. This necessitates an incompressible SKE scheme that guarantees security against unbounded adversaries.

Our construction inherits its compression rate from the inner incompressible SKE scheme. Hence, instantiating it with the scheme proposed by Dziembowski [17] gives a ciphertext rate $\frac{1}{3}$. To overcome this, we build our own incompressible SKE scheme with unbounded security and ciphertext rate $\frac{1}{2}$.

Information-Theoretic Incompressible SKE with Ciphertext-Rate $\frac{1}{2}$. In Dziembowski's scheme, the secret key sk serves as a seed for a randomness extractor. To encrypt a message m , a large random string R is sampled and applied to the extractor (with seed sk) to produce k . This is then used in a *one-time-pad* operation on the message m to produce the ciphertext $(R, k \oplus m)$. We tweak this construction using a seeded strong invertible extractor to get rid of the second component of the ciphertext, namely $k \oplus m$. The secret key for our construction is the seed s for a strong invertible extractor Ext , and a random string k of the same length as the message (which coincides with the output length of the invertible extractor). The ciphertext of a message m is sampled as $\text{ct} = \text{Ext}_s^{-1}(m \oplus k)$, where Ext^{-1} is the inverter for the extractor that samples a random R such that $\text{Ext}_s(R) = m \oplus k$. To decode, the decryptor computes $\text{Ext}_s(\text{ct}) = m \oplus k$ using the seed s in the secret key, and then recovers m using the key k . To argue security, we observe that, by the property of the inverter, and the fact that k is random, $(R, s, m \oplus k)$ is distributed identically as in the experiment in which R and s are sampled uniformly, and $m \oplus k$ is extracted from R using s and Ext . After this step, our security argument follows that of the scheme proposed in [17].

LR Incompressible PKE from LR Non-Committing Encryption. We present a hybrid construction for LR incompressible PKE that combines an incompressible SKE with a leakage-resilient non-committing key encapsulation mechanism (NC-KEM). Essentially, a leakage-resilient NC-KEM is a stronger variant of NC-KEM which can handle leakage from the secret key. We formally define this primitive in Section 4.2 and present a construction based on hash proof systems in Section 6.

The public key pk and secret key sk of the PKE scheme will be the public and secret keys of the NC-KEM scheme. Using the NC-KEM, a fresh SKE key inc.sk is generated by computing $(\text{inc.sk}, \text{ncc.ct}) \leftarrow \text{Encaps}(\text{pk})$. The message m is then encrypted using inc.sk to produce $\text{inc.ct} \leftarrow \text{IncSKE.Enc}(\text{inc.sk}, m)$, and the final ciphertext is $\text{ct} = (\text{ncc.ct}, \text{inc.ct})$. To argue security, we first switch the NC-KEM to simulation mode. The resulting reduction to the base incompressible SKE runs the NC-KEM simulators. Consequently, if all simulators are PPT, then computational incompressible SKE suffices. If even one simulator is possibly

inefficient, then the reduction may also be inefficient, and so we instead need an incompressible SKE secure against unbounded adversaries. For more details, see Section 7.

LR Incompressible PKE from LR Incompressible SKE. We give a high level description of how to bootstrap any LR incompressible SKE scheme along with any regular PKE scheme to an LR incompressible PKE scheme (against general leakage) using *deferred encryption* technique.

The public key of the leakage-resilient incompressible scheme consists of a set of $2n$ public keys, denoted as $\{\text{pk}_{i,b}\}_{i,b}$. On the other hand, the secret key of the scheme includes the secret key k of a leakage-resilient incompressible SKE scheme along with n PKE secret keys corresponding to the bits of k , i.e., $\{\text{sk}_{i,k_i}\}_i$. To encrypt a message m , the encryptor garbles the encryption circuit of the leakage-resilient incompressible SKE scheme with the message m and hardcoded randomness r , i.e., $\text{IncSKE.Enc}(\cdot, m; r)$. This garbling process yields a garbled circuit $\hat{C}$ along with $2n$ labels, given by $\{\text{lab}_{i,b}\}_{i,b}$. Subsequently, each label $\text{lab}_{i,b}$ is encrypted under its corresponding PKE public key $\text{pk}_{i,b}$ to generate a set of ciphertexts, denoted as $\{\text{ct}_{i,b}\}_{i,b}$. The ciphertext consists of the garbled circuit $\hat{C}$ along with the set of ciphertexts $\{\text{ct}_{i,b}\}_{i,b}$. For decryption, the decryptor simply decrypts $\{\text{ct}_{i,k_i}\}_i$ using the corresponding secret keys $\{\text{sk}_{i,k_i}\}_i$. This enables the decryptor to acquire the labels $\{\text{lab}_{i,k_i}\}_i$, which can be utilized to evaluate the garbled circuit $\hat{C}$ and derive an incompressible SKE ciphertext inc.ct . By utilizing the incompressible SKE key k , the decryptor can perform the final decryption to obtain the original message m .

The above approach postpones the actual encryption of the message m until the decryption phase. That is, the actual inc.ct is generated only when the secret key is acquired. Using this fact, in the security game, we are able to equivocate the challenge ciphertext to appear as an encryption under any key k during the second phase. It's important to emphasize that we need to be careful to make sure that the leaking of the secret keys $\{\text{sk}_{i,k_i}\}_i$ does not cause problems for the equivocation process. This aspect is appropriately addressed in our proof. For completeness, we provide our LR incompressible PKE construction in Section 8.

LR Incompressible PKE against Bits-Leakage from Incompressible PKE. We present a transformation that converts any incompressible public-key encryption (PKE) scheme into an LR incompressible PKE secure against leakage functions that output a subset of the secret key bits. This is achieved using a reconstructable probabilistic encoding (RPE). An ℓ -RPE is an encoding where, for any input x and subset S with $|S| \leq \ell$, the partial encoding $\text{Enc}(x)_S$ is perfectly indistinguishable from the uniform distribution. Additionally, there exists a reconstructing algorithm that, given x , a partial codeword $\tilde{w}$, and subset S with $|\tilde{w}| = |S| \leq \ell$, outputs a complete codeword w from the distribution $\text{Enc}(x)$ such that $w_S = \tilde{w}$.

The transformation is straightforward. The public key of the scheme is the public key inc.pk of an incompressible PKE scheme, while the secret key is the RPE of the secret key $\text{RPE.Enc}(\text{inc.sk})$. Any adversary capable of winning the LR incompressible game for this scheme can break the underlying incompressible PKE scheme. This is because the reduction can respond to queries with random values and later reconstruct a valid encoding for inc.sk during the second phase using the reconstructing algorithm of the RPE scheme. For more detail, see Section 9.

LR Incompressible Signatures. We define leakage-resilient incompressible signatures by adding the leakage-resilience property to the security definition of incompressible signatures as mentioned in [24]. The authors of that paper constructed incompressible signatures using a standard signature scheme along with an incompressible encoding scheme (defined in [30]). In the classical random oracle model, we combine the adaptive encoding of Moran and Wichs [30] with the full-state leakage-resilient signature scheme of Boyle, Segev, and Wichs [4]. A signature contains the codeword c and a BSW signature on c . A successful attack either supplies a codeword that was never signed, giving a BSW forgery, or reproduces a previously signed codeword, contradicting encoding incompressibility. We choose the internal BSW parameter to grow logarithmically with the codeword length; this yields negligible relative authentication overhead

and leakage-rate 1 under our fixed- λ rate definitions. The assumptions are (DCR or LWE) for the encoding and Decisional Linear for BSW. For more detail, see Section 10.

3 Preliminaries

Throughout this paper, we will use λ to denote the security parameter and $\text{negl}(\cdot)$ to denote a negligible function in the input. Unless stated otherwise, all length parameters, entropy parameters, and size parameters are assumed to be polynomially bounded in λ , and all algorithms are assumed to run in time polynomial in λ . We will use the shorthand notation PPT for “probabilistic polynomial time”. We use the notation $\mathcal{A}^{\mathcal{B}}$ to denote that the algorithm $\mathcal{A}$ can issue multiple queries to its oracle $\mathcal{B}$ and receives the outputs of $\mathcal{B}$ executed on these queries.

For any finite set X , $x \leftarrow X$ denotes the process of picking an element x from X uniformly at random. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from the distribution $\mathcal{D}$. We will use the notation $\approx_\epsilon$ and $\equiv$ to denote that two distributions are ϵ -close and identical respectively. For any natural number $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \dots, n\}$. For any two binary strings x and y , $x||y$ denotes the concatenation of x and y . For any binary string x , x_i denotes the i^{th} bit of x .

For any function h and a set X that is a subset of the domain of h , we use the shorthand notation $h(X)$ to denote the set $\{h(x)\}_{x \in X}$. For any function F whose first argument is interpreted as a seed, and any seed s , we write $F_s(x) := F(s, x)$. We use symbols such as ℓ_{msg} , ℓ_{st} , ℓ_{sk} , ℓ_{ct} , ℓ_{sign} , ℓ_{leak} , and ℓ_k both for abstract size parameters and for the corresponding parameter-dependent size functions; when their arguments are omitted, they are inferred from the surrounding scheme and experiment. We also fix a canonical binary encoding for every finite space appearing in the paper and identify elements with their encodings whenever convenient. All asymptotic rate definitions in the subsequent subsections are interpreted when the displayed limits exist; all rate calculations in this paper are for families where these limits exist.

3.1 Extractors

Definition 6. For two random variables X and Y taking values in $\mathcal{U}$, their statistical distance is $\Delta(X, Y) = \max_{T \subseteq \mathcal{U}} |\Pr[X \in T] - \Pr[Y \in T]|$. We say that X and Y are ϵ -close if $\Delta(X, Y) \leq \epsilon$.

Definition 7. An extractor $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ is a strong (k, ϵ) -extractor if for every distribution X on $\{0, 1\}^n$ such that $H_\infty(X) \geq k$, the distributions $(S, \text{Ext}_S(X))$ and (U_d, U_m) are ϵ -close, where $S \leftarrow U_d$. Here, U_r denotes the uniform distribution on $\{0, 1\}^r$ and the min-entropy satisfies

$$H_\infty(X) = \min_x \left\{ \log \frac{1}{\Pr[X = x]} \right\}.$$

We will also use the average conditional min-entropy variant of extractors.

Definition 8. Let X and Z be jointly distributed random variables, where X takes values in $\{0, 1\}^n$. The average conditional min-entropy of X conditioned on Z is

$$\tilde{H}_\infty(X | Z) = -\log \left(\mathbb{E}_{z \leftarrow Z} \left[\max_x \Pr[X = x | Z = z] \right] \right).$$

Lemma 9 ([15]). *Let X and Z be jointly distributed random variables. If Z is supported on a set of size at most 2^t , then $\tilde{H}_\infty(X | Z) \geq H_\infty(X) - t$.*

Definition 10. An extractor $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ is a strong average-case (k, ϵ) -extractor if for every pair of jointly distributed random variables (X, Z) such that X takes values in $\{0, 1\}^n$ and $\tilde{H}_\infty(X | Z) \geq k$, the distributions $(Z, S, \text{Ext}_S(X))$ and (Z, U_d, U_m) are ϵ -close, where $S \leftarrow U_d$.

Remark 11. Any strong extractor is also a strong average-case extractor with a small loss in the entropy parameter and error. In particular, if $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ is a strong (k, ϵ) -extractor, then for every $\delta > 0$, it is a strong average-case $(k + \log(1/\delta), \epsilon + \delta)$ -extractor.

We use the following result from Chapter 6 of [34] which describes many randomness extractors.

Theorem 12 (Corollary 6.40, [34]). *For every $n \in \mathbb{N}, k \in [0, n]$ and $\epsilon > 0$, there is an explicit strong (k, ϵ) -extractor $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ with $m = k - O(\log(1/\epsilon))$ and $d = O(\log k \cdot \log(n/\epsilon))$.*

We set $\epsilon = 2^{-\lambda}$ for our applications. We can obtain the following corollary by substituting this value in the above theorem.

Corollary 13. *Let $\lambda \in \mathbb{N}$. For every $n \in \mathbb{N}$ and $k \in [0, n]$, there is an explicit strong $(k, 2^{-\lambda})$ -extractor $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ with $m = k - O(\lambda)$ and*

$$d = O(\log k \cdot (\log n + \lambda)).$$

In particular, when n and k are polynomially bounded in λ , this seed length is $\tilde{O}(\lambda)$.

The following average-case version follows from Corollary 13 and Remark 11.

Corollary 14. *Let $\lambda \in \mathbb{N}$. For every $n \in \mathbb{N}$ and $k \in [0, n]$, there is an explicit strong average-case $(k, 2^{-\lambda})$ -extractor $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ with $m = k - O(\lambda)$ and*

$$d = O(\log(k + \lambda) \cdot (\log n + \lambda)).$$

In particular, when n and k are polynomially bounded in λ , this seed length is $\tilde{O}(\lambda)$.

3.2 Incompressible Public Key Encryption

Let $\text{PKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ be a public key encryption scheme with message space $\{\mathcal{M}_{\ell_{\text{msg}}}\}_{\ell_{\text{msg}}}$, public-key space $\{\mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, secret-key space $\{\mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, and ciphertext space $\{\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, satisfying the following syntax.

- **Setup** $(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm takes as input the security parameter λ , message size ℓ_{msg} and state size ℓ_{st} (equivalently, the adversary's long-term storage bound), all in unary, and outputs a public key pk and a secret key sk .
- **Enc** (pk, m) : The encryption algorithm is a randomized algorithm that takes as input a public key pk , a message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ and outputs a ciphertext $\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$.
- **Dec** $(\text{pk}, \text{sk}, \text{ct})$: The decryption algorithm is deterministic, it takes as input a public key pk , a secret key sk , a ciphertext $\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ and outputs $y \in \mathcal{M}_{\ell_{\text{msg}}} \cup \{\perp\}$.

Remark. In the above syntax, the decryption algorithm takes both the public key and the secret key as input. Equivalently, one could absorb the public key into the secret key and use the more traditional formulation where decryption takes only the secret key. We keep them separate because our separation results are stated with respect to the size of the secret key.

Correctness. An encryption scheme satisfies correctness if for all $\lambda \in \mathbb{N}$, $\ell_{\text{msg}} \in \mathbb{N}$, $\ell_{\text{st}} \in \mathbb{N}$, and $m \in \mathcal{M}_{\ell_{\text{msg}}}$,

$$\Pr [\text{Dec}(\text{pk}, \text{sk}, \text{Enc}(\text{pk}, m)) = m : (\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})] \geq 1 - \text{negl}(\lambda),$$

where the probability is over the randomness of **Setup** and **Enc**.

Security. Consider the following experiment with a two-stage interactive adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Incompressible PKE Experiment $\text{Expt}_{\mathcal{A}, \text{PKE}}^{\text{IncompPKE}}(\lambda)$:

- **First Phase:**

- $\mathcal{A}_1$ on input 1^λ , outputs the message size $1^{\ell_{\text{msg}}}$ and the state size $1^{\ell_{\text{st}}}$. The challenger computes $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
- $\mathcal{A}_1$ on input pk , outputs the challenge messages $m_0, m_1 \in \mathcal{M}_{\ell_{\text{msg}}}$ and auxiliary information aux .
- The challenger chooses $b \leftarrow \{0, 1\}$, computes a ciphertext $\text{ct}^* \leftarrow \text{Enc}(\text{pk}, m_b)$ and sends it to $\mathcal{A}_1$.
- $\mathcal{A}_1$ computes a state st such that $|\text{st}| \leq \ell_{\text{st}}$ and sends it to the challenger.

- **Second Phase:**

- $\mathcal{A}_2$ receives $(\text{pk}, \text{sk}, \text{aux}, \text{st})$ and outputs bit b' . $\mathcal{A}$ wins the experiment if $b = b'$.

Definition 15. A PKE scheme is said to be incompressible if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left| \Pr[\text{Expt}_{\mathcal{A}, \text{PKE}}^{\text{IncompPKE}}(\lambda) = 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

The scheme is said to satisfy information-theoretic incompressibility security if the above guarantee holds for all adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ (including unbounded adversaries).

Rates. Let $\ell_{\text{ct}} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ denote the ciphertext size as a function of λ , ℓ_{msg} , and ℓ_{st} . Clearly, $\ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) \geq \ell_{\text{msg}}$ is necessary for correct decryption. Moreover, if $\ell_{\text{st}} \geq \ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, then the adversary can store the entire ciphertext, so the security notion is only meaningful in the regime $\ell_{\text{st}} < \ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$. We define two rates — the ciphertext rate and the compression rate.

- ciphertext rate: By convention, we measure the ciphertext rate along the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$:

$$\text{CiphRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{msg}}}{\ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}.$$

- compression rate: For fixed λ and ℓ_{msg} , the compression rate is

$$\text{CompRate}(\lambda, \ell_{\text{msg}}) = \lim_{\ell_{\text{st}} \rightarrow \infty} \frac{\ell_{\text{st}}}{\ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}.$$

Depending on the number of ciphertexts that decrypt to a particular message under a given public/secret key pair, we categorise messages as good or bad. Intuitively, we would expect each message to possess an equal proportion of the ciphertexts. If the proportion significantly exceeds the average, we label it as ‘bad’. Additionally, we single out public/secret key pairs as bad when they are associated with too many bad messages. The proof of the impossibility results will heavily rely on these definitions.

Definition 16. Let $\lambda \in \mathbb{N}$, $\ell_{\text{msg}} \in \mathbb{N}$, $\ell_{\text{st}} \in \mathbb{N}$, $\alpha \in (0, 1)$ and $(\text{pk}, \text{sk}) \in \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \times \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. A message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ is said to be α -**bad** with respect to (pk, sk) if

$$\Pr_{\text{ct} \leftarrow \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}} [\text{Dec}(\text{pk}, \text{sk}, \text{ct}) = m] > 1/2^{\alpha \cdot \ell_{\text{msg}}}.$$

A message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ is said to be α -**good** with respect to $\text{pk} \in \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ if it is not α -bad with respect to any (pk, sk) where $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. A public/secret key pair $(\text{pk}, \text{sk}) \in \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \times \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ is said to be α -**bad** if

$$\Pr_{m \leftarrow \mathcal{M}_{\ell_{\text{msg}}}} [m \text{ is } \alpha\text{-bad with respect to } (\text{pk}, \text{sk})] > 1/2^{(1-\alpha) \cdot \ell_{\text{msg}}}.$$

Guan *et al.* [24] gave the first ciphertext-rate-1 and compression-rate-1 scheme using indistinguishable obfuscators (iO) [18, 33, 19]. Later, Branco *et al.* [7] gave a construction from standard assumptions.

Theorem 17 ([7]). *Assuming either the hardness of LWE together with DCR, or the hardness of DDH together with DCR, there exists an incompressible PKE scheme with ciphertext-rate 1 and compression-rate 1.*

3.3 Incompressible Secret Key Encryption

Let $\text{SKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ be a secret key encryption scheme with message space $\{\mathcal{M}_{\ell_{\text{msg}}}\}_{\ell_{\text{msg}}}$, secret key space $\{\mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, and ciphertext space $\{\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. It consists of the following PPT algorithms.

- **Setup** $(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm takes as input the security parameter λ , the message length ℓ_{msg} , and state size ℓ_{st} and outputs a secret key $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$.
- **Enc**(sk, m) : The encryption algorithm is a randomized algorithm that takes as input a secret key sk and a message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ and outputs a ciphertext $\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$.
- **Dec**(sk, ct) : The decryption algorithm is deterministic, it takes as input a secret key sk and a ciphertext ct and outputs either a message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ or $\perp$.

Correctness. For correctness, we require that for all $\lambda \in \mathbb{N}$, $\ell_{\text{msg}} \in \mathbb{N}$, $\ell_{\text{st}} \in \mathbb{N}$, and $m \in \mathcal{M}_{\ell_{\text{msg}}}$,

$$\Pr [\text{Dec}(\text{sk}, \text{Enc}(\text{sk}, m)) = m : \text{sk} \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})] \geq 1 - \text{negl}(\lambda),$$

where the probability is over the randomness of **Setup** and **Enc**.

Incompressible SKE Security. Consider the following experiment with a two-stage interactive adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Incompressible SKE Experiment $\text{Expt}_{\mathcal{A}, \text{SKE}}^{\text{IncompSKE}}(\lambda)$:

- **First Phase:**
 - $\mathcal{A}_1$ on input 1^λ , outputs the message length $1^{\ell_{\text{msg}}}$ and the state size $1^{\ell_{\text{st}}}$. The challenger runs $\text{sk} \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
 - $\mathcal{A}_1$ outputs the challenge messages m_0, m_1 , along with auxiliary information aux .
 - The challenger randomly chooses $b \in \{0, 1\}$ and computes a ciphertext $\text{ct}^* = \text{Enc}(\text{sk}, m_b)$ and sends it to $\mathcal{A}_1$.
 - $\mathcal{A}_1$ computes a state st such that $|\text{st}| \leq \ell_{\text{st}}$ and sends it to the challenger.
- **Second Phase:**
 - $\mathcal{A}_2$ receives $(\text{sk}, \text{aux}, \text{st})$ and outputs b' . $\mathcal{A}$ wins the experiment if $b = b'$.

Definition 18. An SKE scheme is said to be incompressible secure if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left| \Pr[\text{Expt}_{\mathcal{A}, \text{SKE}}^{\text{IncompSKE}}(\lambda) = 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

The scheme is said to satisfy information-theoretic incompressibility security if the above guarantee holds for all adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ (including unbounded adversaries).

Rates. Let $\ell_{\text{ct}} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ denote the ciphertext size as a function of λ , ℓ_{msg} , and ℓ_{st} . Clearly, $\ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) \geq \ell_{\text{msg}}$ is necessary for correct decryption. Moreover, if $\ell_{\text{st}} \geq \ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, then the adversary can store the entire ciphertext, so the security notion is only meaningful in the regime $\ell_{\text{st}} < \ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$. We define two rates — the ciphertext rate and the compression rate.

- ciphertext rate: By convention, we measure the ciphertext rate for fixed λ and along the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$:

$$\text{CiphRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{msg}}}{\ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}.$$

- compression rate: For fixed λ and ℓ_{msg} , the compression rate is

$$\text{CompRate}(\lambda, \ell_{\text{msg}}) = \lim_{\ell_{\text{st}} \rightarrow \infty} \frac{\ell_{\text{st}}}{\ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}.$$

Depending on the number of ciphertexts that decrypt to a particular message under a given secret key, we categorise messages as good or bad. Intuitively, we would expect each message to possess an equal proportion of the ciphertexts. If the proportion significantly exceeds the average, we label it as ‘bad’. Additionally, we single out secret keys as bad when they are associated with too many bad messages. The proof of the impossibility results will heavily rely on these definitions.

Definition 19. Let $\lambda \in \mathbb{N}, \ell_{\text{msg}} \in \mathbb{N}, \ell_{\text{st}} \in \mathbb{N}, \alpha \in (0, 1)$ and $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. A message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ is said to be α -**bad** with respect to sk if

$$\Pr_{\text{ct} \leftarrow \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}} [\text{Dec}(\text{sk}, \text{ct}) = m] > 1/2^{(\alpha) \cdot \ell_{\text{msg}}}.$$

A message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ is said to be α -**good** if it is not α -bad with respect to any $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. A secret key $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ is said to be α -**bad** if

$$\Pr_{m \leftarrow \mathcal{M}_{\ell_{\text{msg}}}} [m \text{ is } \alpha\text{-bad with respect to } \text{sk}] > 1/2^{(1-\alpha) \cdot \ell_{\text{msg}}}.$$

Theorem 20 ([17]). *There exists an information-theoretic incompressible SKE scheme for arbitrary message length that is secure against unbounded adversaries with the following parameters and rates: secret key length $\ell_{\text{sk}} = \ell_{\text{msg}} + d$, where*

$$d = O(\log(\ell_{\text{msg}} + \lambda) \cdot (\log(\ell_{\text{st}} + \ell_{\text{msg}} + \lambda) + \lambda)),$$

ciphertext-rate $\frac{1}{3}$, and compression-rate 1.

Remark. The scheme in Theorem 20 is obtained from the incompressible SKE construction of Dziembowski [17], instantiated with the seeded average-case extractor from Corollary 14. We briefly recall the construction and the parameter calculation.

Let $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^{\ell_{\text{msg}}}$ be a strong average-case extractor. The secret key is $\text{sk} = (s, t)$, where $s \in \{0, 1\}^d$ is the extractor seed and $t \in \{0, 1\}^{\ell_{\text{msg}}}$ is a one-time pad mask. To encrypt a message $\text{msg} \in \{0, 1\}^{\ell_{\text{msg}}}$, sample $R \leftarrow \{0, 1\}^n$ uniformly and output $\text{ct} = (R, \text{Ext}_s(R) \oplus \text{msg} \oplus t)$. Decryption on input $\text{sk} = (s, t)$ and $\text{ct} = (R, X)$ outputs $X \oplus \text{Ext}_s(R) \oplus t$.

We instantiate the parameters as follows. Let $\epsilon = 2^{-\lambda}$ and let ℓ_{st} denote the adversary’s storage bound. We use a strong average-case extractor $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^{\ell_{\text{msg}}}$ that extracts ℓ_{msg} bits from average conditional min-entropy $k = \ell_{\text{msg}} + O(\lambda)$ with error ϵ . We set $n = \ell_{\text{st}} + k$. Then, for $R \leftarrow \{0, 1\}^n$ and any stored state st of length at most ℓ_{st} , Lemma 9

gives $\tilde{H}_\infty(R \mid \text{st}) \geq n - \ell_{\text{st}} = k$. Hence, by the strong average-case extractor guarantee, $\text{Ext}_s(R)$ is statistically close to uniform even given the stored state st and the seed s .

The ciphertext consists of $R \in \{0, 1\}^n$ and one additional ℓ_{msg} -bit string. Therefore, $|\text{ct}| = n + \ell_{\text{msg}} = \ell_{\text{st}} + k + \ell_{\text{msg}} = \ell_{\text{st}} + 2\ell_{\text{msg}} + O(\lambda)$. To evaluate the ciphertext rate, we use the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$. In this regime, $|\text{ct}| = 3\ell_{\text{msg}} + O(\lambda)$, and hence $\ell_{\text{msg}}/|\text{ct}| = 1/3 + o(1)$. Thus, according to our convention, the ciphertext rate is $1/3$. On the other hand, for the compression rate, we fix λ and ℓ_{msg} and let $\ell_{\text{st}} \rightarrow \infty$. Since $\ell_{\text{st}}/|\text{ct}| = \ell_{\text{st}}/(\ell_{\text{st}} + 2\ell_{\text{msg}} + O(\lambda)) = 1 - o(1)$ in this regime, the compression rate is 1.

Finally, the secret key length is $|\text{sk}| = d + \ell_{\text{msg}}$. By Corollary 14,

$$d = O(\log(k + \lambda) \cdot (\log n + \lambda)).$$

Thus $|\text{sk}| = \ell_{\text{msg}} + O(\log(k + \lambda)(\log n + \lambda))$; when n and k are polynomially bounded in λ , this specializes to $|\text{sk}| = \ell_{\text{msg}} + \tilde{O}(\lambda)$.

Theorem 21 ([7]). *Assuming hardness of LWE problem or DCR problem, there exists an incompressible SKE scheme with ciphertext-rate 1 and compression-rate 1.*

3.4 Incompressible Encoding

We use a slightly modified version of the definition from [24]. In our formulation, the state size is provided as an input to the encoding algorithm rather than being specified separately in the incompressibility definition. This is only a syntactic change and better fits our framework. The notion of incompressible encodings was introduced by Moran and Wichs [30].

Definition 22 (Incompressible Encodings). An incompressible encoding scheme for message space $\{0, 1\}^{\ell_{\text{msg}}}$, codeword space $\{0, 1\}^{\ell_{\text{code}}}$, and state space $\{0, 1\}^{\ell_{\text{st}}}$ is a pair of PPT algorithms $\text{Code} = (\text{Enc}, \text{Dec})$ with the following syntax:

- $\text{Enc}(1^\lambda, 1^{\ell_{\text{st}}}, m)$: The encoding algorithm takes as input the security parameter 1^λ , the state size $1^{\ell_{\text{st}}}$, and a message m , and outputs a codeword c .
- $\text{Dec}(c)$: The decoding algorithm takes as input a codeword c and outputs a message m .

We require correctness and the following incompressibility property.

Correctness. For all security parameters $\lambda \in \mathbb{N}$, all state lengths $\ell_{\text{st}} \in \mathbb{N}$, all message lengths $\ell_{\text{msg}} \in \mathbb{N}$, and all messages $m \in \{0, 1\}^{\ell_{\text{msg}}}$,

$$\Pr[\text{Dec}(\text{Enc}(1^\lambda, 1^{\ell_{\text{st}}}, m)) = m] \geq 1 - \text{negl}(\lambda).$$

where the probability is over the randomness of the encoding algorithm.

Security. Consider the following experiment for a two-stage interactive adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

Codeword Compression Experiment $\text{Expt}_{\mathcal{A}, \text{Code}}^{\text{IncompCode}}(\lambda)$:

1. **First phase:**

- On input the security parameter 1^λ , the adversary $\mathcal{A}_1$ outputs a state size $1^{\ell_{\text{st}}}$, a message m , and auxiliary input aux .
- The challenger samples a codeword $c \leftarrow \text{Enc}(1^\lambda, 1^{\ell_{\text{st}}}, m)$ and sends c to $\mathcal{A}_1$.
- The adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.

2. **Second phase:**

- The adversary $\mathcal{A}_2$ receives $(\text{st}, m, \text{aux})$ and outputs a codeword c' .

- The experiment outputs 1 if and only if $c' = c$.

Definition 23 (Incompressibility). An incompressible encoding scheme Code is said to be incompressible if, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$\Pr[\text{Expt}_{\mathcal{A}, \text{Code}}^{\text{Incomp}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

Rates. Let $\ell_{\text{code}} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ denote the codeword size as a function of λ , ℓ_{msg} , and ℓ_{st} . We define two rates — the encoding rate and the compression rate.

- encoding rate: By convention, we measure the encoding rate for fixed λ and in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$:

$$\lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{msg}}}{\ell_{\text{code}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}.$$

- compression rate: For fixed λ and ℓ_{msg} , the compression rate is

$$\lim_{\ell_{\text{st}} \rightarrow \infty} \frac{\ell_{\text{st}}}{\ell_{\text{code}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}.$$

CRS and random oracle models. The preceding syntax and experiment describe the plain model. In the CRS variant, the parameters are announced before the challenger samples the encoding's public string crs . Both encoding algorithms and both adversarial stages receive this string, whose input is suppressed. Selective security requires (m, aux) to be fixed before the CRS is revealed. The string remains available in the second phase and is not charged to the stored state. Correctness probabilities include its sampling.

In the classical random oracle model, all algorithms and adversarial stages have consistent access throughout the experiment, including parameter and message selection, to one uniform random function $\text{RO} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$. The encoding uses inputs $\langle \text{MW}, r, j \rangle$, where $r \in \{0, 1\}^\lambda$ is a fresh nonce and j is a block index, with an injective encoding of the tuple. This is the oracle of Moran and Wichs [30], with an unrestricted index field so that the length functions also make sense in our fixed- λ rate limits. For polynomially bounded lengths and sufficiently large λ , it is just a relabeling of their fixed-length oracle inputs; the unused inputs provide independent oracle answers. The oracle is not reset between phases; no query transcript is given to the second stage except through its prescribed inputs. Its randomness is included in all probabilities. Messages and auxiliary inputs may depend on prior oracle queries. These are model variants of the plain-model definition above.

Moran and Wichs [30] give incompressible encodings under either DCR or LWE, selectively secure in the CRS model and adaptively secure, with their composability guarantee, in the random oracle model. We use the latter construction. Their separate selection and compression stages also cover our interactive first-stage syntax: include the selection-stage working state, and the target message, in the auxiliary input before the codeword is sampled, and let the expansion algorithm ignore any extra fields. This does not increase the compressed-state length and does not expose any information chosen after the challenge. More explicitly, their Theorems 7.1 and 5.3 give polynomially bounded block parameters $r(\lambda)$, $d(\lambda)$ and $h(\lambda) = e(\lambda) - \ell(\lambda) > 0$ such that a padded $Br(\lambda)$ -bit message has a codeword of length $Bd(\lambda) + a(\lambda) + \lambda$ and is incompressible against storage of at most $Bh(\lambda) - \lambda$ bits. Here $a(\lambda)$ accounts for the encoding's public key and is polynomially bounded. Their good surjective lossy functions satisfy

$$\frac{r(\lambda)}{d(\lambda)} = 1 - o(1), \quad \frac{h(\lambda)}{d(\lambda)} = 1 - o(1).$$

For our syntax, choose

$$B = \max \left\{ \left\lceil \frac{\ell_{\text{msg}}}{r(\lambda)} \right\rceil, \left\lceil \frac{\ell_{\text{st}} + \lambda}{h(\lambda)} \right\rceil \right\},$$

pad the message to $Br(\lambda)$ bits, and include a deterministic self-delimiting header specifying $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$. The decoder removes the padding. Thus, writing $C = \ell_{\text{code}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$,

$$C = Bd(\lambda) + a(\lambda) + \lambda + O(\log(\lambda + \ell_{\text{msg}} + \ell_{\text{st}} + 2)). \quad (1)$$

All fields have fixed lengths for a given parameter triple. The header and padding are fixed before the challenge codeword; stripping the header reduces exact reconstruction to that of the native encoding without increasing stored state. Security therefore follows from the adaptive RO guarantee. The encoding-rate is $\min\{r(\lambda), h(\lambda)\}/d(\lambda)$, and the compression-rate is $h(\lambda)/d(\lambda)$, both $1 - o(1)$ as $\lambda \rightarrow \infty$. These calculations take the length limits at fixed λ , exactly as above. Finite exceptional security parameters can be handled by using a fixed larger block parameter.

3.5 Garbling Scheme

A garbling scheme $\text{GC} = (\text{Garble}, \text{Eval})$ for a class of circuits $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$, where every circuit $C \in \mathcal{C}_n$ has input length n , consists of the following algorithms.

- $\text{Garble}(1^\lambda, C \in \mathcal{C}_n)$: The garbling algorithm takes as input the security parameter 1^λ and a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$, and outputs a garbled circuit $\hat{C}$ and a set of labels $\{\text{lab}_{i,b}\}_{i \in [n], b \in \{0,1\}}$.
- $\text{Eval}(\hat{C}, \{\text{lab}_i\}_{i \in [n]})$: The evaluation algorithm takes as input a garbled circuit $\hat{C}$ and one label lab_i for each input wire $i \in [n]$, and outputs a string $y \in \{0, 1\}^m$.

Correctness. For correctness, we require perfect correctness: for all λ , $C \in \mathcal{C}_n$, and $x \in \{0, 1\}^n$,

$$\Pr \left[\text{Eval}(\hat{C}, \{\text{lab}_{i,x_i}\}_{i \in [n]}) = C(x) : (\hat{C}, \{\text{lab}_{i,b}\}_{i \in [n], b \in \{0,1\}}) \leftarrow \text{Garble}(1^\lambda, C) \right] = 1$$

where the probability is over the randomness of the **Garble** algorithm.

Security. For security, we define simulation based security [36, 3].

Definition 24. A garbling scheme $\text{GC} = (\text{Garble}, \text{Eval})$ for a class of circuits $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ is said to be secure if there exists a PPT algorithm Sim such that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that for all λ , the following holds.

$$\left| \Pr \left[\mathcal{A}_2(\hat{C}, \{\text{lab}_i\}_{i \in [n]}, aux) = 1 : \begin{array}{l} (C, x, aux) \leftarrow \mathcal{A}_1(1^\lambda), \\ (\hat{C}, \{\text{lab}_i\}_{i \in [n]}) \leftarrow \text{Sim}(1^\lambda, 1^n, 1^{|C|}, C(x)) \end{array} \right] \right. \\ \left. - \Pr \left[\mathcal{A}_2(\hat{C}, \{\text{lab}_{i,x_i}\}_{i \in [n]}, aux) = 1 : \begin{array}{l} (C, x, aux) \leftarrow \mathcal{A}_1(1^\lambda), \\ (\hat{C}, \{\text{lab}_{i,b}\}_{i \in [n], b \in \{0,1\}}) \leftarrow \text{Garble}(1^\lambda, C) \end{array} \right] \right| \leq \text{negl}(\lambda).$$

3.6 Reconstructable Probabilistic Encoding

We recall the notion of reconstructable probabilistic encoding (RPE) [10, 11, 2]. We use normalized Hamming distance throughout this subsection. For $u, v \in \{0, 1\}^n$, let $\Delta(u, v) := \frac{1}{n} \|u - v\|_1$. A (k, n, ρ, ℓ) binary reconstructable probabilistic encoding scheme consists of algorithms $(\text{Enc}, \text{Dec}, \text{Recon})$ satisfying the following properties.

- **Correctness.** The encoder Enc is a probabilistic algorithm that, on input $x \in \{0, 1\}^k$, outputs $w \in \{0, 1\}^n$. The decoder Dec is a deterministic algorithm that, on input $w' \in \{0, 1\}^n$, outputs an element of $\{0, 1\}^k$.

For every $x \in \{0, 1\}^k$, if $w \leftarrow \text{Enc}(x)$, then for every $w' \in \{0, 1\}^n$ satisfying $\Delta(w, w') \leq \rho$, we have $\text{Dec}(w') = x$.

In particular, when $\rho = 0$, this says that every valid output of $\text{Enc}(x)$ is decoded back to x .

- **Uniformity.** For every $x \in \{0, 1\}^k$ and every subset $S \subseteq [n]$ such that $|S| \leq \lfloor \ell n \rfloor$, the random variable $\text{Enc}(x)_S$ is uniformly distributed over $\{0, 1\}^{|S|}$.
- **Reconstructability.** The reconstruction algorithm **Recon** is a probabilistic algorithm that takes as input $x \in \{0, 1\}^k$, a set $S \subseteq [n]$ with $|S| \leq \lfloor \ell n \rfloor$, and a string $\tilde{w} \in \{0, 1\}^{|S|}$. It outputs a string $w \in \{0, 1\}^n$ sampled from the conditional distribution of $\text{Enc}(x)$ given that $w_S = \tilde{w}$.

By uniformity and reconstructability, we obtain the following immediate corollary.

Corollary 25. *For any (k, n, ρ, ℓ) binary RPE and any $x \in \{0, 1\}^k$, the following two distributions are identical:*

$$\left(S, w, \tilde{w} \left| \begin{array}{l} S \leftarrow \{T \subseteq [n] : |T| = \lfloor \ell n \rfloor\} \\ w \leftarrow \text{Enc}(x) \\ \tilde{w} = w_S \end{array} \right. \right) \equiv \left(S, w, \tilde{w} \left| \begin{array}{l} S \leftarrow \{T \subseteq [n] : |T| = \lfloor \ell n \rfloor\} \\ \tilde{w} \leftarrow \{0, 1\}^{\lfloor \ell n \rfloor} \\ w \leftarrow \text{Recon}(x, S, \tilde{w}) \end{array} \right. \right).$$

Consequently, the corresponding marginal distributions over $(w, \tilde{w})$ are also identical.

Proof. Fix any set $S \subseteq [n]$ of size $\lfloor \ell n \rfloor$. By the uniformity property, $\text{Enc}(x)_S$ is uniformly distributed over $\{0, 1\}^{|S|}$. Therefore, sampling $w \leftarrow \text{Enc}(x)$ and setting $\tilde{w} = w_S$ is equivalent to first sampling $\tilde{w} \leftarrow \{0, 1\}^{|S|}$ uniformly and then sampling w from the conditional distribution of $\text{Enc}(x)$ given $w_S = \tilde{w}$. This is exactly the distribution sampled by $\text{Recon}(x, S, \tilde{w})$. Averaging over the uniformly sampled set S gives the claimed joint distributional identity. The marginal identity over $(w, \tilde{w})$ follows by applying the projection $(S, w, \tilde{w}) \mapsto (w, \tilde{w})$ to both sides. $\square$

For our purposes, we do not require error correction. Hence we will only use the case $\rho = 0$.

Theorem 26 ([2, Lemma 2]). *If there exists a binary linear $[n, k, d]$ code, then there exists a $(k, n, 0, (d-1)/n)$ binary reconstructable probabilistic encoding.*

The RPE encoder obtained in Theorem 26 is not the original error-correcting-code encoder. Rather, the construction uses the linear structure of the code, in particular suitable generator and parity-check matrices, to sample uniformly from an affine subspace associated with the message. Thus, the theorem should be viewed as a way of converting a linear code with large distance into a locally uniform probabilistic encoding.

We instantiate Theorem 26 using binary linear codes with relative distance close to $1/2$ and rate close to the Gilbert–Varshamov bound.

Corollary 27 (Specialized consequence of the Ta-Shma-code parameter analysis of [27, Section 8.2]). *Let $\beta > 0$ be an arbitrary constant. Then there exists a constant $c_\beta > 0$ and thresholds $k_\beta \in \mathbb{N}$ and $\varepsilon_\beta > 0$, depending only on β , such that for every $k \geq k_\beta$ and $0 < \varepsilon \leq \varepsilon_\beta$ there exists a binary linear $[N, k, d]$ code satisfying*

$$N \leq \frac{k}{c_\beta \varepsilon^{2+\beta}} \quad \text{and} \quad d \geq \left(\frac{1}{2} - \frac{\varepsilon}{2} \right) N.$$

For fixed β , a single deterministic algorithm constructs a generator matrix from k and rational ε in time polynomial in k , $1/\varepsilon$, and the bit length of ε .

Proof. We make the parameter dependence in [27, Sections 8.1–8.2] explicit. Choose a power of two $s \geq \max\{128, 40/\beta\}$ and set $\alpha = 1/s$. These are constants depending only on β . The base-code bias, the graph degrees, and the graph approximation parameter in that construction then depend only on β , not on ε or k . The base code has dimension k and length $O_\beta(k)$; its construction and the graph construction of [27, Lemma B.1] therefore require only a threshold $k \geq k_\beta$ independent of ε . The restriction in [27, Equation (9)] is satisfied whenever

$$0 < \varepsilon \leq \varepsilon_\beta := 2^{-4s^5 \log_2 s}.$$

Section 8.2 chooses the walk length for an arbitrary such ε , including the rounding needed between levels of the code cascade. By [27, Claim 8.9], the resulting linear map has bias at most ε on every nonzero input and length $O_\beta(k\varepsilon^{-(2+40/s)})$. It is injective because $\varepsilon < 1$, and hence has dimension exactly k . Since $40/s \leq \beta$, this gives the displayed bounds with a constant c_β independent of k and ε .

The total walk length is $O_\beta(\log(1/\varepsilon))$ and all branching degrees are constants. We can therefore enumerate the output coordinates and evaluate the linear map on the k basis vectors in time polynomial in k and $1/\varepsilon$. The integer walk lengths can be selected by exact comparisons after raising the rational inequalities to fixed integer powers. This gives the asserted uniform generator-matrix algorithm; no error-correcting decoder is needed here. $\square$

Corollary 28. *Let $\beta > 0$ be an arbitrary constant. Then there exists a constant $c_\beta > 0$ and thresholds k_β and ε_β , depending only on β , such that for every $k \geq k_\beta$ and $0 < \varepsilon \leq \varepsilon_\beta$ there exists a binary RPE with parameters $(k, N, 0, \ell)$ satisfying*

$$N \leq \frac{k}{c_\beta \varepsilon^{2+\beta}} \quad \text{and} \quad \ell \geq \frac{1}{2} - \frac{\varepsilon}{2} - \frac{1}{N}.$$

Its description is constructible uniformly with the same polynomial parameter dependence, and its encoding, decoding, and reconstruction algorithms run in time polynomial in k and N .

Proof. By Corollary 27, uniformly for $k \geq k_\beta$ and $0 < \varepsilon \leq \varepsilon_\beta$, there exists a binary linear $[N, k, d]$ code such that

$$N \leq \frac{k}{c_\beta \varepsilon^{2+\beta}} \quad \text{and} \quad d \geq \left(\frac{1}{2} - \frac{\varepsilon}{2}\right) N.$$

Set $d_0 := \lceil (1/2 - \varepsilon/2)N \rceil \leq d$. Applying Theorem 26 and using the smaller, efficiently computable leakage fraction $(d_0 - 1)/N$, we obtain a binary RPE with parameters $(k, N, 0, (d_0 - 1)/N)$. Therefore,

$$\ell = \frac{d_0 - 1}{N} \geq \frac{1}{2} - \frac{\varepsilon}{2} - \frac{1}{N},$$

as required.

For completeness, let $G \in \mathbb{F}_2^{k \times N}$ be the full-row-rank generator matrix furnished above. Encode x by sampling w uniformly subject to $Gw = x$, and decode by computing Gw . If $|S| < d$, the columns outside S still have rank k : otherwise some nonzero row combination of G would be a codeword supported on S , contradicting its minimum distance. Consequently, every prescribed w_S admits exactly $2^{N-|S|-k}$ completions. This proves uniformity, and sampling uniformly from the affine solution space with both $Gw = x$ and $w_S = \tilde{w}$ gives exact reconstruction. Gaussian elimination implements all three algorithms in polynomial time. $\square$

3.7 Hash Proof Systems

Let $\mathcal{L} \subseteq X$ be an NP language with witness relation $R_{\mathcal{L}}$, and let $Y \subseteq X \setminus \mathcal{L}$ be a designated set of non-language instances. We use a formulation in which the relevant distributions over language and non-language instances are sampled explicitly.

A hash proof system for $\mathcal{L}$ is a tuple of PPT algorithms

$$\text{HPS} = (\text{Gen}, \text{SampL}, \text{SampY}, \text{KeyGen}, \text{Encap}, \text{Decap})$$

with the following syntax.

- **Gen**($1^\lambda, 1^{\ell_k}$): on input the security parameter 1^λ and the key-entropy parameter 1^{ℓ_k} , the parameter-generation algorithm outputs public parameters pp together with a trapdoor $\text{td}_{\mathcal{L}}$. The public parameters implicitly determine a finite key space $\mathcal{K}_{\text{pp}}$, a set $X_{\lambda, \ell_k} \subseteq X$, a language $\mathcal{L}_{\lambda, \ell_k} \subseteq X_{\lambda, \ell_k}$, and a designated set of non-language instances $Y_{\lambda, \ell_k} \subseteq X_{\lambda, \ell_k} \setminus \mathcal{L}_{\lambda, \ell_k}$.
- **SampL**(pp): outputs a pair (x, w) such that $(x, w) \in R_{\mathcal{L}}$ and $x \in \mathcal{L}_{\lambda, \ell_k}$.

- $\text{SampY}(\text{pp}, \text{td}_{\mathcal{L}})$: outputs an element $x \in Y_{\lambda, \ell_k}$.
- $\text{KeyGen}(\text{pp})$: outputs a public key pk and a secret key sk .
- $\text{Encap}(\text{pk}, x, w)$: is a randomized algorithm that takes as input a public key pk and a pair $(x, w) \in R_{\mathcal{L}}$ with $x \in X_{\lambda, \ell_k}$, and outputs a key $k \in \mathcal{K}_{\text{pp}}$.
- $\text{Decap}(\text{sk}, x)$: takes as input a secret key sk and an element $x \in X_{\lambda, \ell_k}$, and outputs a key $k \in \mathcal{K}_{\text{pp}}$.

Correctness. There exists a negligible function negl such that for every $\lambda \in \mathbb{N}$ and every $\ell_k \in \mathbb{N}$, if $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_k})$, $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp})$, and $(x, w) \leftarrow \text{SampL}(\text{pp})$, then

$$\Pr[\text{Decap}(\text{sk}, x) = \text{Encap}(\text{pk}, x, w)] \geq 1 - \text{negl}(\lambda),$$

where the probability is taken over the randomness of all the algorithms.

Language indistinguishability. For every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$ and all $\ell_k \in \mathbb{N}$,

$$\left| \Pr \left[\mathcal{A}(\text{pp}, x) = 1 : \begin{array}{l} (\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_k}), \\ (x, w) \leftarrow \text{SampL}(\text{pp}) \end{array} \right] - \Pr \left[\mathcal{A}(\text{pp}, x) = 1 : \begin{array}{l} (\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_k}), \\ x \leftarrow \text{SampY}(\text{pp}, \text{td}_{\mathcal{L}}) \end{array} \right] \right| \leq \text{negl}(\lambda).$$

Smoothness. For every (possibly unbounded) adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$ and all $\ell_k \in \mathbb{N}$,

$$\left| \Pr \left[\mathcal{A}(\text{pp}, \text{pk}, x, k) = 1 : \begin{array}{l} (\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_k}) \\ (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp}) \\ x \leftarrow \text{SampY}(\text{pp}, \text{td}_{\mathcal{L}}) \\ k := \text{Decap}(\text{sk}, x) \end{array} \right] - \Pr \left[\mathcal{A}(\text{pp}, \text{pk}, x, k) = 1 : \begin{array}{l} (\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_k}) \\ (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{pp}) \\ x \leftarrow \text{SampY}(\text{pp}, \text{td}_{\mathcal{L}}) \\ k \leftarrow \mathcal{K}_{\text{pp}} \end{array} \right] \right| \leq \text{negl}(\lambda).$$

For our application, we use the DDH-based programmable hash proof system of Branco *et al.* [7] only as an ordinary hash proof system. Since we do not need programmability here, we can omit the outer balancing hash used in their construction. With a suitable choice of parameters, this yields a language $\mathcal{L} \subseteq X$ and a hash proof system whose public-parameter dependent key space has entropy at least ℓ_k while instances in X_{λ, ℓ_k} have size $o(\ell_k) \cdot \text{poly}(\lambda)$.

Theorem 29 (Adapted from [7]). *Assuming the hardness of the DDH problem, for every polynomially bounded $\ell_k \in \mathbb{N}$ and every constant $\alpha \in (0, 1/2)$, there exists a language $\mathcal{L} \subseteq X$ together with a hash proof system for $\mathcal{L}$ whose public parameters determine a finite key space $\mathcal{K}_{\text{pp}}$ such that $\log |\mathcal{K}_{\text{pp}}| \geq \ell_k$, smoothness is perfectly uniform over $\mathcal{K}_{\text{pp}}$, and every instance $x \in X_{\lambda, \ell_k}$ has size $O(\ell_k^{2\alpha} \cdot b_{\mathcal{G}})$, where p is the order of the underlying DDH group and $b_{\mathcal{G}} = \text{poly}(\lambda)$ is the bit length of its fixed-length injective group-element encoding. The secret-key length is*

$$|\text{sk}| = \ell_k + O(\ell_k^{1-\alpha} + \ell_k^\alpha \log p).$$

In particular, for fixed λ (and hence fixed group), $|x| = o(\ell_k)$ and $|\text{sk}| = \ell_k + o(\ell_k)$. In a joint regime, the former conclusion holds if $b_{\mathcal{G}} = o(\ell_k^{1-2\alpha})$, while the latter holds if $\log p = o(\ell_k^{1-\alpha})$.

Proof. We derive the claim from the DDH-based hash proof system underlying Construction 2 of Branco *et al.* [7], but *without* applying the outer balancing hash function HC .

Fix parameters $\ell, s \in \mathbb{N}$ and a cyclic group $\mathcal{G} = \langle g \rangle$ of prime order p . For $z \in \mathbb{Z}_p$, write $[z] := g^z \in \mathcal{G}$, and extend this notation entrywise to vectors and matrices over $\mathbb{Z}_p$. For a matrix B over $\mathbb{Z}_p$ and a bracketed vector or matrix $[C]$ of compatible dimensions, write $B[C]$ as shorthand for $[BC]$. Similarly, for a bracketed matrix $[C]$ and a matrix D over $\mathbb{Z}_p$ of compatible dimensions, write $[C]D$ as shorthand for $[CD]$.

Let $h \leftarrow \mathbb{Z}_p^\ell \setminus 0$ and set $\mathbf{pp} := (\mathcal{G}, p, [h], \ell, s)$, $\mathbf{td}_{\mathcal{L}} := h$, and $\mathcal{K}_{\mathbf{pp}} := \mathcal{G}^{s \times (\ell-1)}$. Define

$$\mathcal{L} = \{[hw^\top] : w \in \mathbb{Z}_p^{\ell-1}\} \subseteq \mathcal{G}^{\ell \times (\ell-1)}$$

and

$$Y = \left\{ [E] \in \mathcal{G}^{\ell \times (\ell-1)} : \begin{pmatrix} h & E \end{pmatrix} \text{ is invertible over } \mathbb{Z}_p \right\}.$$

The hash proof system is defined as follows: **Gen** samples $\mathbf{pp}$ as above and sets $\mathbf{td}_{\mathcal{L}} := h$, **SampL** samples $w \leftarrow \mathbb{Z}_p^{\ell-1}$ and outputs $([hw^\top], w)$, and **SampY** uses $\mathbf{td}_{\mathcal{L}} = h$ to sample E uniformly subject to $\begin{pmatrix} h & E \end{pmatrix}$ being invertible, and outputs $[E]$, as in Branco *et al.* [7]. For example, find a matrix B whose columns complement h to a basis, sample $u \leftarrow \mathbb{Z}_p^{\ell-1}$ and a uniform invertible $T \in \mathbb{Z}_p^{(\ell-1) \times (\ell-1)}$, and set $E = hu^\top + BT$. Every admissible E has a unique such decomposition, which proves uniformity. Basis completion and sampling an invertible matrix are efficient by linear algebra and standard finite-field sampling. The remaining algorithms are:

- **KeyGen** samples $A \leftarrow \mathbb{Z}_p^{s \times \ell}$ uniformly at random, sets $\mathbf{sk} := A$, and $\mathbf{pk} := A[h] \in \mathcal{G}^s$.
- On input $(\mathbf{pk}, x = [hw^\top], w)$ with witness $w \in \mathbb{Z}_p^{\ell-1}$, define

$$\mathbf{Encap}(\mathbf{pk}, x, w) := \mathbf{pk} \cdot w^\top = A[hw^\top] \in \mathcal{G}^{s \times (\ell-1)}.$$

- On input $(\mathbf{sk} = A, x = [E])$, define

$$\mathbf{Decap}(A, [E]) := A[E] \in \mathcal{G}^{s \times (\ell-1)}.$$

We now verify the required properties.

Correctness. Let $x = [hw^\top] \in \mathcal{L}$. Then

$$\mathbf{Decap}(A, x) = A[hw^\top] = (A[h])w^\top = \mathbf{pk} \cdot w^\top = \mathbf{Encap}(\mathbf{pk}, x, w).$$

Hence correctness holds perfectly.

Language indistinguishability. This is exactly the DDH-based language used in the construction of Branco *et al.* [7]. They prove that samples from $\mathcal{L}$ are computationally indistinguishable from samples from Y under the DDH assumption. Since we only removed the post-processing map HC from the output of encapsulation/decapsulation, the language and its sampling procedures are unchanged, and the same indistinguishability proof applies.

Smoothness. Fix any $x = [E] \in Y$. By definition of Y , the matrix $M := \begin{pmatrix} h & E \end{pmatrix} \in \mathbb{Z}_p^{\ell \times \ell}$ is invertible. Consider one row $a \in \mathbb{Z}_p^\ell$ of A . The map $a \mapsto (ah, aE)$ is a bijection from $\mathbb{Z}_p^\ell$ to $\mathbb{Z}_p \times \mathbb{Z}_p^{\ell-1}$ because M is invertible. Therefore, conditioning on the value of ah leaves aE uniformly distributed over $\mathbb{Z}_p^{\ell-1}$. Applying this row by row, we obtain that conditioned on $\mathbf{pk} = A[h]$, the matrix $A[E] \in \mathcal{G}^{s \times (\ell-1)} = \mathcal{K}_{\mathbf{pp}}$ is uniform. Thus, for every fixed $x \in Y$, the value $\mathbf{Decap}(\mathbf{sk}, x)$ is exactly uniform over $\mathcal{K}_{\mathbf{pp}}$ given $\mathbf{pk}$ and x , which proves statistical smoothness against unbounded adversaries.

Parameter choice. Use the fixed-length injective group-element encoding of length b_G . Then $|x| = \ell(\ell - 1)b_G$, and entrywise encoding of the raw key gives an efficient injective encoding of length $n_{\text{src}} = s(\ell - 1)b_G$. The usual case $b_G = O(\log p)$ recovers bounds stated in terms of $\log p$, but is not needed here. Let $\kappa := \ell_k$ denote the target key-entropy parameter. For sufficiently large κ , choose

$$\ell = \lceil \kappa^\alpha \rceil, \quad s = \left\lceil \frac{\kappa}{(\ell - 1) \log p} \right\rceil.$$

Then, $\log |\mathcal{K}_{\text{pp}}| = s(\ell - 1) \log p \geq \kappa$. Moreover, $|x| = \ell(\ell - 1)b_G = O(\ell^2 b_G) = O(\kappa^{2\alpha} b_G)$. Encode the scalar secret key A by packing its $s\ell$ entries, in a fixed order, as the base- p digits of one integer in $[0, p^{s\ell})$. Its binary representation has length $\lceil s\ell \log p \rceil$, with leading zeros as needed. Packing and unpacking use polynomial-time integer arithmetic, not discrete logarithms, and the algorithms above use the unpacked matrix. Thus

$$|\text{sk}| = \lceil s\ell \log p \rceil \leq \frac{\kappa \ell}{\ell - 1} + \ell \log p + 1 = \kappa + O(\kappa^{1-\alpha} + \kappa^\alpha \log p).$$

Since $\alpha < 1/2$, the asserted fixed- λ estimates follow. The displayed joint-regime conditions follow by requiring respectively $\kappa^{2\alpha} b_G = o(\kappa)$ and $\kappa^\alpha \log p = o(\kappa)$.

Hence, under DDH, there exists a language $\mathcal{L} \subseteq X$ together with a hash proof system for $\mathcal{L}$ whose public parameters determine a finite key space $\mathcal{K}_{\text{pp}}$ with $\log |\mathcal{K}_{\text{pp}}| \geq \ell_k$, whose smoothness is perfectly uniform over $\mathcal{K}_{\text{pp}}$, and whose instances in X_{λ, ℓ_k} have size $O(\ell_k^{2\alpha} b_G)$, with the secret-key bound and the fixed- λ and joint-regime consequences stated in the theorem. $\square$

3.8 Leakage-Resilient Signatures

Leakage-resilient signatures were studied by Katz and Vaikuntanathan [28]. We use the stronger full-state bounded-leakage construction of Boyle, Segev, and Wichs (BSW) [4]. Their guarantee permits leakage from key-generation and signing randomness, even interleaved with signing queries. Our experiment only needs leakage from the signing key before the signing phase, and is covered by that guarantee. We use a message-recovering formulation: verification returns the signed message. The conversion from ordinary signatures is given below. As in BSW, public verification data are also available to the signer; we suppress this public input to **Sign**. Only the secret signing material is counted in ℓ_{sk} .

Definition 30 (Leakage-Resilient Signatures). A leakage-resilient signature scheme family **Sig** with message spaces $\{\mathcal{M}_{\ell_{\text{msg}}}\}_{\ell_{\text{msg}}}$, signing-key spaces $\{\mathcal{SK}_{\lambda, \ell_{\text{msg}}}\}_{\lambda, \ell_{\text{msg}}}$, verification-key spaces $\{\mathcal{VK}_{\lambda, \ell_{\text{msg}}}\}_{\lambda, \ell_{\text{msg}}}$, signature spaces $\{\Sigma_{\lambda, \ell_{\text{msg}}}\}_{\lambda, \ell_{\text{msg}}}$, and leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}}\}_{\lambda, \ell_{\text{msg}}}$ is a tuple of PPT algorithms (**Gen**, **Sign**, **Verify**) with the following syntax:

- **Gen**($1^\lambda, 1^{\ell_{\text{msg}}}$) : The key-generation algorithm takes as input the security parameter 1^λ and the message length $1^{\ell_{\text{msg}}}$, and outputs a verification key $\text{vk} \in \mathcal{VK}_{\lambda, \ell_{\text{msg}}}$ and a signing key $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}}$.
- **Sign**(sk, m) : The signing algorithm takes as input the signing key sk and a message $m \in \mathcal{M}_{\ell_{\text{msg}}}$, and outputs a signature $\sigma \in \Sigma_{\lambda, \ell_{\text{msg}}}$.
- **Verify**(vk, σ) : The verification algorithm takes as input the verification key vk and a signature $\sigma \in \Sigma_{\lambda, \ell_{\text{msg}}}$, and outputs either a message $m' \in \mathcal{M}_{\ell_{\text{msg}}}$ or $\perp$.

Correctness. For all security parameters $\lambda \in \mathbb{N}$, all message lengths $\ell_{\text{msg}} \in \mathbb{N}$, and all messages $m \in \mathcal{M}_{\ell_{\text{msg}}}$, if $(\text{vk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_{\text{msg}}})$ and $\sigma \leftarrow \text{Sign}(\text{sk}, m)$, then

$$\Pr[\text{Verify}(\text{vk}, \sigma) = m] \geq 1 - \text{negl}(\lambda).$$

where the probability is over the randomness of **Gen** and **Sign**.

The concrete BSW algorithms below are oracle-free. In the classical random-oracle composition of Section 10, adversarial stages and leakage computations also access the global oracle. The proof there simulates this independent oracle, including calls inside leakage computations, within BSW's standard-model experiment.

Security. Let $\ell_{\text{leak}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. For a message length ℓ_{msg} , consider the following experiment for an adversary $\mathcal{A}$:

Leakage-Resilient Unforgeability Experiment $\text{Expt}_{\mathcal{A}, \text{Sig}}^{\text{LeakSig}}(\lambda, \ell_{\text{msg}}, \ell_{\text{leak}})$:

1. **Setup:** The challenger samples $(\text{vk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_{\text{msg}}})$ and sends vk to $\mathcal{A}$.
2. **Leakage phase:**
 - For polynomially many rounds, the adversary $\mathcal{A}$ adaptively submits a leakage function $f_i \in \mathcal{F}_{\lambda, \ell_{\text{msg}}}$.
 - The challenger returns the leakage value $\ell_i := f_i(\text{sk})$, provided that the total number of leaked bits satisfies $\sum_i |\ell_i| \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}})$.
3. **Signing phase:**
 - For polynomially many rounds, the adversary $\mathcal{A}$ adaptively submits a message $m_i \in \mathcal{M}_{\ell_{\text{msg}}}$.
 - The challenger computes $\sigma_i \leftarrow \text{Sign}(\text{sk}, m_i)$ and sends σ_i to $\mathcal{A}$.
4. **Forgery:**
 - The adversary $\mathcal{A}$ outputs a signature σ . Let $m := \text{Verify}(\text{vk}, \sigma)$.
 - The experiment outputs 1 if and only if $m \neq \perp$ and m was not queried during the signing phase.

Definition 31 (ℓ_{leak} -Leakage-Resilient Unforgeability). A signature scheme family Sig is said to be ℓ_{leak} -leakage-resilient unforgeable with respect to the leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}}\}_{\lambda, \ell_{\text{msg}}}$ if, for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda, \ell_{\text{msg}} \in \mathbb{N}$,

$$\Pr[\text{Expt}_{\mathcal{A}, \text{Sig}}^{\text{LeakSig}}(\lambda, \ell_{\text{msg}}, \ell_{\text{leak}}) = 1] \leq \text{negl}(\lambda).$$

Rates. For any family of such schemes in which $\ell_{\text{msg}} = \ell_{\text{msg}}(\lambda)$ is a chosen message-length regime, and $\ell_{\text{sign}} = \ell_{\text{sign}}(\lambda, \ell_{\text{msg}})$, $\ell_{\text{leak}} = \ell_{\text{leak}}(\lambda, \ell_{\text{msg}})$, and $\ell_{\text{sk}} = \ell_{\text{sk}}(\lambda, \ell_{\text{msg}})$ are the corresponding parameter-dependent lengths, we define two rates — the signature rate and the leakage rate.

- signature rate: For fixed λ

$$\text{SignRate} = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{msg}}}{\ell_{\text{sign}}(\lambda, \ell_{\text{msg}})}.$$

- leakage rate: For fixed λ ,

$$\text{LeakRate} = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{leak}}(\lambda, \ell_{\text{msg}})}{\ell_{\text{sk}}(\lambda, \ell_{\text{msg}})}.$$

The BSW instantiation. We use the generic bit-string construction of [4, Section 5, Theorem 5.1], instantiated under the Decisional Linear (DLIN) assumption. Its internal security parameter is denoted by n , distinct from λ and from any message length. Its secret key is a uniform string $x \in \{0, 1\}^{\mu(n)}$. For a public-coin second-preimage-resistant function with output length $\kappa(n)$, its permitted total leakage is $\mu(n) - \kappa(n) - \omega(\log n)$ bits, including full-state leakage. Fix a polynomially bounded output length $\kappa(n)$ and set

$$\mu(n) = n(\kappa(n) + n), \quad L_{\text{BSW}}(n) = \mu(n) - \kappa(n) - n = \mu(n)(1 - 1/n). \quad (2)$$

The entropy margin n is $\omega(\log n)$.

We spell out the ingredients to make the assumption explicit. BSW uses public-coin second-preimage-resistant functions, public-coin admissible hash functions, tagged lossy encryption with obviously samplable public keys, and a statistical witness-indistinguishable argument for NP with an obviously samplable reference string. Under DLIN these are available as described in [4, Sections 2.3–2.6 and 4]: public-coin collision-resistant hashing from discrete logarithms supplies both the second-preimage-resistant function and the admissible-hash construction; the lossy encryption follows from decisional-linear lossy trapdoor functions; and the required argument follows from the DLIN instantiation of Groth, Ostrovsky, and Sahai [23]. The public hash for second-preimage resistance can have a fixed polynomial output length $\kappa(n)$ and the polynomial input length $\mu(n)$ above. No separate hash or proof-system assumption is added. We use this generic bit-string instantiation so that secret-key length and leakage are both measured in bits.

In particular, BSW’s admissible hash is a deterministic, public function $H : \{0,1\}^* \rightarrow \{0,1\}^{\tau(n)}$. It is part of the verification key and maps an arbitrary-length message to the tag of the encryption inside the signature. Its admissibility is used in BSW’s adaptive unforgeability proof; it is not a random oracle. A BSW signature consists of that ciphertext and an argument, both of length bounded by a fixed polynomial $p(n)$, independently of the message length. The argument’s statement contains $H(m)$ rather than m . Evaluation of H still takes time polynomial in $n + |m|$. We absorb framing and padding into $p(n)$ and use fixed-length authentication strings. Public hash descriptions, encryption keys, and argument reference strings belong to the verification key and need not be repeated in each signature.

Message recovery. Write $(\text{BSW.Gen}, \text{BSW.Sign}, \text{BSW.Verify})$ for these ordinary signatures, where verification takes the message as a separate input. The message-recovering version signs c by outputting (c, σ) , where $\sigma \leftarrow \text{BSW.Sign}(x, c)$; verification returns c exactly when $\text{BSW.Verify}(\text{vk}, c, \sigma) = 1$, and returns $\perp$ otherwise. Its signature length on a C -bit message is $C + p(n)$. To prove leakage-resilient unforgeability, forward leakage queries unchanged, answer a query c by appending c to the signature returned by the ordinary signing oracle, and turn a successful message-recovering forgery (c, σ) into the ordinary forgery (c, σ) . The recovered message is new exactly when the ordinary forgery message is new. The secret key, leakage budget, and success probability are unchanged. Thus ordinary, rather than strong, unforgeability suffices. Section 10 chooses n as a function of λ and C and verifies the resulting rates.

3.9 Simulatable Attacks

We take the following definitions from [35] with some minor modifications. In particular, we return to Wichs’s general convention that each cryptographic game $\mathcal{G}$ comes with a baseline function $c_{\mathcal{G}} : \mathbb{N} \rightarrow [0, 1]$. Decisional games are the special case $c_{\mathcal{G}}(\lambda) = 1/2$.

Definition 32 (Cryptographic Game). A cryptographic game $\mathcal{G}$ is defined by a (possibly inefficient) random system $\mathcal{C}$ called the challenger, together with a baseline function $c_{\mathcal{G}} : \mathbb{N} \rightarrow [0, 1]$. When given as input the security parameter λ in unary, the challenger $\mathcal{C}$ interacts with some attacker $\mathcal{A}$ possibly for multiple rounds and outputs a bit b , which is taken to be the output of the game $\mathcal{G}$. We use the shorthand $b = \langle \mathcal{A}(1^\lambda), \mathcal{C}(1^\lambda) \rangle$ to denote this process. The advantage of $\mathcal{A}$ in the game $\mathcal{G}$ is given by $\text{Adv}_{\mathcal{G}}^{\mathcal{A}}(\lambda) = \Pr[\langle \mathcal{A}(1^\lambda), \mathcal{C}(1^\lambda) \rangle = 1] - c_{\mathcal{G}}(\lambda)$. A cryptographic game $\mathcal{G}$ is secure if for every PPT adversary $\mathcal{A}$, there exists a negligible function μ such that $\text{Adv}_{\mathcal{G}}^{\mathcal{A}}(\lambda) \leq \mu(\lambda)$ for all sufficiently large λ .

We say that an attacker $\mathcal{A}$ successfully breaks the security of $\mathcal{G}$ if $\text{Adv}_{\mathcal{G}}^{\mathcal{A}}(\lambda)$ is positive and non-negligible in λ . This is a one-sided security notion: only positive non-negligible advantage constitutes a break. Below, we give a more general definition that captures other kinds of cryptographic scenarios.

Definition 33 (Cryptographic Property). Let $\mathcal{A}$ denote the set of all adversaries, efficient or inefficient. A cryptographic property $\mathcal{C}$ is defined as a function $\mathcal{C} : \mathcal{A} \times \mathbb{N} \rightarrow [0, 1]$ such that for an attacker $\mathcal{A} \in \mathcal{A}$ and security parameter $\lambda \in \mathbb{N}$, $\mathcal{C}(\mathcal{A}, \lambda) = \text{Adv}_{\mathcal{C}}^{\mathcal{A}}(\lambda)$ denotes the advantage of $\mathcal{A}$ in breaking the property. We say that $\mathcal{C}$ is secure if for all PPT attackers $\mathcal{A}$, the advantage $\mathcal{C}(\mathcal{A}, \lambda) = \text{Adv}_{\mathcal{C}}^{\mathcal{A}}(\lambda)$ is negligible in λ .

Most proofs in cryptography have the form of a black-box reduction, and this is what we define below.

Definition 34 (Black-Box Reduction). Let $\mathcal{C}$ be a cryptographic property and let $\mathcal{G}$ be a cryptographic game. A black-box reduction deriving the security of $\mathcal{C}$ from that of $\mathcal{G}$ is an oracle PPT machine $\mathcal{R}^{(\cdot)}$ for which there exist a polynomial $p(\cdot)$ and an integer λ_0 such that, for every $\lambda \geq \lambda_0$ and every possibly inefficient non-uniform attacker $\mathcal{A}_\lambda$ satisfying $\text{Adv}_{\mathcal{C}}^{\mathcal{A}_\lambda}(\lambda) \geq 1/2$, we have

$$\text{Adv}_{\mathcal{G}}^{\mathcal{R}^{\mathcal{A}_\lambda}}(\lambda) \geq \frac{1}{p(\lambda)}.$$

We next recall the simulatable-attack paradigm of Wichs [35], which is a general method for proving black-box separation results. The idea is to construct an inefficient but successful attacker against a cryptographic property $\mathcal{C}$, together with an efficient stateful simulator Sim such that no oracle machine can distinguish oracle access to the inefficient attacker from oracle access to Sim . The simulator itself need not be a valid attacker against $\mathcal{C}$, since it may keep state or coordinate between oracle queries in a way that a valid attacker for $\mathcal{C}$ is not allowed to do.

In Wichs's original formulation, the inefficient attack is point-wise in the auxiliary choice: for every security parameter λ and every $h \in \mathcal{H}_\lambda$, the attacker $\mathcal{A}_{\lambda,h}$ breaks $\mathcal{C}$. In our applications, we use the following average-case variant, where this holds for all but a negligible fraction of choices of $h \leftarrow \mathcal{H}_\lambda$.

Definition 35 (Average-case Simulatable Attack). A simulatable attack on a cryptographic property $\mathcal{C}$ consists of

1. an ensemble of possibly inefficient stateless non-uniform attackers $\{\mathcal{A}_{\lambda,h}\}_{\lambda \in \mathbb{N}, h \in \mathcal{H}_\lambda}$, where $\mathcal{H}_\lambda$ are finite sets, and
2. a stateful PPT simulator Sim .

The auxiliary choice h may be any element of the finite family $\mathcal{H}_\lambda$. In particular, pairs such as (g, h) are covered by taking $\mathcal{H}_\lambda = G_\lambda \times H_\lambda$. We require the following two properties to hold:

- **Inefficient attack:** There exists a negligible function ν_{atk} such that, for every $\lambda \in \mathbb{N}$,

$$\Pr_{h \leftarrow \mathcal{H}_\lambda} \left[\text{Adv}_{\mathcal{C}}^{\mathcal{A}_{\lambda,h}}(\lambda) \geq \frac{1}{2} \right] \geq 1 - \nu_{\text{atk}}(\lambda).$$

- **Simulatability:** There exists a negligible function $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, for every polynomial $q = q(\lambda)$ and every possibly inefficient oracle machine $\mathcal{R}^{(\cdot)}$ making at most $q = q(\lambda)$ queries to its oracle, it holds that

$$\left| \Pr_{h \leftarrow \mathcal{H}_\lambda, \rho_{\mathcal{R}}, \rho_{\mathcal{A}_{\lambda,h}}} [\mathcal{R}^{\mathcal{A}_{\lambda,h}}(1^\lambda) = 1] - \Pr_{\rho_{\mathcal{R}}, \rho_{\text{Sim}}} [\mathcal{R}^{\text{Sim}(1^\lambda)}(1^\lambda) = 1] \right| \leq \text{poly}(q(\lambda)) \cdot \text{negl}(\lambda),$$

where the first probability is over the choice of $h \leftarrow \mathcal{H}_\lambda$, the coins of $\mathcal{R}$, and the coins of $\mathcal{A}_{\lambda,h}$, and the second is over the coins of $\mathcal{R}$ and Sim .

The following theorem is the average-case analogue of Wichs's simulatable-attack theorem. The proof is the same meta-reduction argument, except that we lose a negligible term from the negligible fraction of bad choices of $h \leftarrow \mathcal{H}_\lambda$ for which the inefficient attack may fail.

Theorem 36 (Average-case simulatable attacks, following [35]). *Suppose there exists an average-case simulatable attack against a cryptographic property $\mathcal{C}$. If there is a black-box reduction deriving the security of $\mathcal{C}$ from the security of a cryptographic game $\mathcal{G}$, then $\mathcal{G}$ is not secure. More precisely, there exists a PPT adversary that breaks $\mathcal{G}$ with non-negligible advantage.*

Proof. Let $\mathcal{G}$ be the cryptographic game with challenger $\mathcal{C}$ and baseline function $c_{\mathcal{G}}$. Let $\{\mathcal{A}_{\lambda,h}\}_{\lambda \in \mathbb{N}, h \in \mathcal{H}_{\lambda}}$ and Sim be the average-case simulatable attack against $\mathcal{C}$.

Let $\mathcal{B}^{(\cdot)}$ be the black-box reduction deriving the security of $\mathcal{C}$ from the security of $\mathcal{G}$. By the black-box reduction guarantee, there exist a polynomial $p(\cdot)$ and an integer λ_0 such that, for every $\lambda \geq \lambda_0$ and every possibly inefficient non-uniform attacker $\mathcal{A}$ that breaks $\mathcal{C}$ with the required advantage, we have

$$\text{Adv}_{\mathcal{G}}^{\mathcal{B}^{\mathcal{A}}}(\lambda) = \Pr_{\rho_{\mathcal{B}}, \rho_{\mathcal{A}}, \rho_{\mathcal{C}}} [\langle \mathcal{B}^{\mathcal{A}}(1^{\lambda})(1^{\lambda}), \mathcal{C}(1^{\lambda})(1^{\lambda}) \rangle = 1] - c_{\mathcal{G}}(\lambda) \geq \frac{1}{p(\lambda)}.$$

By the average-case inefficient-attack property, there exists a negligible function ν_{atk} such that, except with probability at most $\nu_{\text{atk}}(\lambda)$ over $h \leftarrow \mathcal{H}_{\lambda}$, we have $\text{Adv}_{\mathcal{C}}^{\mathcal{A}_{\lambda,h}}(\lambda) \geq 1/2$. Let $\text{Good}_{\lambda} = \{h \in \mathcal{H}_{\lambda} : \text{Adv}_{\mathcal{C}}^{\mathcal{A}_{\lambda,h}}(\lambda) \geq \frac{1}{2}\}$. Then $\Pr_{h \leftarrow \mathcal{H}_{\lambda}}[h \in \text{Good}_{\lambda}] \geq 1 - \nu_{\text{atk}}(\lambda)$.

For every $h \in \text{Good}_{\lambda}$, the reduction guarantee gives $\text{Adv}_{\mathcal{G}}^{\mathcal{B}^{\mathcal{A}_{\lambda,h}}}(\lambda) \geq \frac{1}{p(\lambda)}$. For $h \notin \text{Good}_{\lambda}$, we use only the trivial lower bound $\text{Adv}_{\mathcal{G}}^{\mathcal{B}^{\mathcal{A}_{\lambda,h}}}(\lambda) \geq -1$. Also,

$$\mathbb{E}_{h \leftarrow \mathcal{H}_{\lambda}} [\text{Adv}_{\mathcal{G}}^{\mathcal{B}^{\mathcal{A}_{\lambda,h}}}(\lambda)] = \Pr_{h \leftarrow \mathcal{H}_{\lambda}, \rho_{\mathcal{B}}, \rho_{\mathcal{A}_{\lambda,h}}, \rho_{\mathcal{C}}} [\langle \mathcal{B}^{\mathcal{A}_{\lambda,h}}(1^{\lambda})(1^{\lambda}), \mathcal{C}(1^{\lambda})(1^{\lambda}) \rangle = 1] - c_{\mathcal{G}}(\lambda).$$

Therefore,

$$\begin{aligned} \mathbb{E}_{h \leftarrow \mathcal{H}_{\lambda}} [\text{Adv}_{\mathcal{G}}^{\mathcal{B}^{\mathcal{A}_{\lambda,h}}}(\lambda)] &\geq (1 - \nu_{\text{atk}}(\lambda)) \cdot \frac{1}{p(\lambda)} - \nu_{\text{atk}}(\lambda) \cdot 1 \\ &= \frac{1}{p(\lambda)} - \text{negl}(\lambda). \end{aligned}$$

Now define the oracle machine $\mathcal{M}^{(\cdot)}$ that runs $\mathcal{B}^{(\cdot)}(1^{\lambda})$ while interacting with the challenger $\mathcal{C}(1^{\lambda})$, and outputs the final bit output by $\mathcal{C}$. Since $\mathcal{B}$ is PPT, there exists a polynomial $q(\cdot)$ such that $\mathcal{M}$ makes at most $q(\lambda)$ oracle queries. By the simulatability property,

$$\begin{aligned} &\left| \Pr_{h \leftarrow \mathcal{H}_{\lambda}, \rho_{\mathcal{B}}, \rho_{\mathcal{A}_{\lambda,h}}, \rho_{\mathcal{C}}} [\langle \mathcal{B}^{\mathcal{A}_{\lambda,h}}(1^{\lambda})(1^{\lambda}), \mathcal{C}(1^{\lambda})(1^{\lambda}) \rangle = 1] \right. \\ &\quad \left. - \Pr_{\rho_{\mathcal{B}}, \rho_{\text{Sim}}, \rho_{\mathcal{C}}} [\langle \mathcal{B}^{\text{Sim}(1^{\lambda})}(1^{\lambda})(1^{\lambda}), \mathcal{C}(1^{\lambda})(1^{\lambda}) \rangle = 1] \right| \leq \text{poly}(q(\lambda)) \cdot \text{negl}(\lambda). \end{aligned}$$

Since $q(\lambda)$ is polynomially bounded, $\text{poly}(q(\lambda)) \cdot \text{negl}(\lambda)$ is still negligible. Combining the previous two inequalities, we obtain

$$\text{Adv}_{\mathcal{G}}^{\mathcal{B}^{\text{Sim}(1^{\lambda})}}(\lambda) = \Pr_{\rho_{\mathcal{B}}, \rho_{\text{Sim}}, \rho_{\mathcal{C}}} [\langle \mathcal{B}^{\text{Sim}(1^{\lambda})}(1^{\lambda})(1^{\lambda}), \mathcal{C}(1^{\lambda})(1^{\lambda}) \rangle = 1] - c_{\mathcal{G}}(\lambda) \geq \frac{1}{p(\lambda)} - \text{negl}(\lambda).$$

Thus $\mathcal{B}^{\text{Sim}(1^{\lambda})}$ breaks $\mathcal{G}$ with non-negligible advantage. Since both $\mathcal{B}$ and Sim are PPT, the composed adversary $\mathcal{B}^{\text{Sim}(1^{\lambda})}$ is also PPT. Hence $\mathcal{G}$ is not secure. $\square$

3.10 Probability

Choosing a uniformly random function h is equivalent to sampling the value $h(x)$ lazily as queries arrive. This is analogous to throwing balls into bins, and we will use the following standard bound on the maximum load.

Lemma 37. For every $p \geq 2q \lg(q)$, if p balls are thrown independently and uniformly at random into q bins, then the maximum load is smaller than $\frac{e \cdot p}{q}$ with probability at least $1 - \frac{1}{q}$. Moreover, when $p = q$, the probability that the maximum load exceeds $k = 2 \frac{\ln(p)}{\ln \ln(p)}$ is at most $\exp(-\frac{p}{ek!})$.

We next bound the probability that an oracle machine can predict the value of a random function at a point that it never queried.

Theorem 38. Let H denote the set of all functions from $\{0, 1\}^m$ to $\{0, 1\}^n$. Let $\mathcal{A}$ be an unbounded oracle Turing machine with oracle access to a uniformly random function $h \leftarrow H$ via oracle $\mathcal{O}^h$, where $\mathcal{O}^h(x) = h(x)$ for every $x \in \{0, 1\}^m$. Let X be the set of oracle queries made by $\mathcal{A}$ during the execution. Then, for every $q \in \mathbb{N}$,

$$\Pr_{h \leftarrow H} \left[\begin{array}{l} \mathcal{A}^{\mathcal{O}^h(\cdot)}(1^n, 1^m) \rightarrow \{(x_i, z_i)\}_{i \in [q]} \text{ and} \\ \exists i \in [q] \text{ such that } h(x_i) = z_i \text{ and } x_i \notin X \end{array} \right] \leq \frac{q}{2^n}.$$

The next theorem extends the previous one to the case where the adversary outputs sets and attempts to predict the image of some element in one of those sets.

Theorem 39. Let H denote the set of all functions from $\{0, 1\}^m$ to $\{0, 1\}^n$. Let $\mathcal{A}$ be an unbounded oracle Turing machine with oracle access to a uniformly random function $h \leftarrow H$ via oracle $\mathcal{O}^h$, where $\mathcal{O}^h(x) = h(x)$ for every $x \in \{0, 1\}^m$. Let X be the set of oracle queries made by $\mathcal{A}$ during the execution. Then, for every $q, \ell \in \mathbb{N}$,

$$\Pr_{h \leftarrow H} \left[\begin{array}{l} \mathcal{A}^{\mathcal{O}^h(\cdot)}(1^n, 1^m) \rightarrow \{(S_i, z_i)\}_{i \in [q]} \text{ and} \\ \exists i \in [q] \text{ such that } z_i \in h(S_i) \text{ and} \\ S_i \cap X \cap h^{-1}(z_i) = \emptyset \end{array} \right] \leq \frac{q \cdot \ell}{2^n},$$

where $|S_i| \leq \ell$ for all $i \in [q]$.

4 Leakage-Resilient Incompressible and Non-Committing Schemes: Definitions

Throughout this section, we parameterize leakage resilience by a leakage-class family $\mathcal{F}$ of admissible leakage functions. The adversary may make polynomially many leakage queries, subject to a global leakage bound. When $\mathcal{F}$ is sufficiently expressive and closed under adaptive composition, polynomially many leakage queries can be captured by a single leakage query that outputs the entire leakage transcript. However, this equivalence need not hold for more restricted leakage classes, and hence we explicitly allow polynomially many leakage queries.

4.1 Leakage-Resilient Incompressible Encryption

We define leakage-resilient incompressible PKE and SKE by augmenting the corresponding incompressible notions from the preliminaries with pre-challenge leakage on the secret key.

Leakage-Resilient Incompressible PKE. Let $\text{PKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ be a PKE scheme with message space $\{\mathcal{M}_{\ell_{\text{msg}}}\}_{\ell_{\text{msg}}}$, public-key space $\{\mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, secret-key space $\{\mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, ciphertext space $\{\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ and leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. It consists of the following PPT algorithms.

- **Setup** $(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm takes as input the security parameter, the message size, and the state size, all in unary, and outputs a public key $\text{pk} \in \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ and a secret key $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$.

- $\text{Enc}(\text{pk}, m)$: The encryption algorithm is randomized. On input a public key pk and a message $m \in \mathcal{M}_{\ell_{\text{msg}}}$, it outputs a ciphertext $\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$.
- $\text{Dec}(\text{pk}, \text{sk}, \text{ct})$: The decryption algorithm is deterministic. On input a public key pk , a secret key sk , and a ciphertext $\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, it outputs either a message in $\mathcal{M}_{\ell_{\text{msg}}}$ or $\perp$.

Correctness. We require that for all $\lambda \in \mathbb{N}$, $\ell_{\text{msg}} \in \mathbb{N}$, $\ell_{\text{st}} \in \mathbb{N}$, and $m \in \mathcal{M}_{\ell_{\text{msg}}}$,

$$\Pr [\text{Dec}(\text{pk}, \text{sk}, \text{Enc}(\text{pk}, m)) = m : (\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})] \geq 1 - \text{negl}(\lambda),$$

where the probability is over the randomness of Setup and Enc .

Security. Consider the following experiment with a two-stage interactive adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Leakage-Resilient Incompressible PKE Experiment $\text{Expt}_{\mathcal{A}, \text{PKE}}^{\text{LRIncompPKE}}(\lambda, \ell_{\text{leak}})$:

- **First Phase:**
 - $\mathcal{A}_1$ on input 1^λ outputs the message size $1^{\ell_{\text{msg}}}$ and the state size $1^{\ell_{\text{st}}}$. The challenger computes $(\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$ and sends pk to $\mathcal{A}_1$.
 - **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}_1$ is allowed to make polynomially many leakage queries. For the j^{th} query, it submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, where $f_j : \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_j}$. The challenger returns $f_j(\text{sk})$ to $\mathcal{A}_1$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.
 - $\mathcal{A}_1$ outputs the challenge messages $m_0, m_1 \in \mathcal{M}_{\ell_{\text{msg}}}$ and auxiliary information aux .
 - The challenger chooses $b \leftarrow \{0, 1\}$, computes $\text{ct}^* \leftarrow \text{Enc}(\text{pk}, m_b)$, and sends ct^* to $\mathcal{A}_1$.
 - $\mathcal{A}_1$ computes a state st such that $|\text{st}| \leq \ell_{\text{st}}$ and sends it to the challenger.
- **Second Phase:**
 - $\mathcal{A}_2$ receives $(\text{pk}, \text{sk}, \text{aux}, \text{st})$ and outputs a bit b' . The adversary wins if $b = b'$.

Definition 40. Let $\ell_{\text{leak}} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. A PKE scheme is said to be ℓ_{leak} -leakage-resilient incompressible secure with respect to the leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left| \Pr[\text{Expt}_{\mathcal{A}, \text{PKE}}^{\text{LRIncompPKE}}(\lambda, \ell_{\text{leak}}) = 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

Leakage-Resilient Incompressible SKE. Let $\text{SKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ be an SKE scheme with message space $\{\mathcal{M}_{\ell_{\text{msg}}}\}_{\ell_{\text{msg}}}$, secret-key space $\{\mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, ciphertext space $\{\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, and leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. It consists of the following PPT algorithms.

- $\text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm takes as input the security parameter, the message size, and the state size, all in unary, and outputs a secret key $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$.
- $\text{Enc}(\text{sk}, m)$: The encryption algorithm is randomized. On input a secret key sk and a message $m \in \mathcal{M}_{\ell_{\text{msg}}}$, it outputs a ciphertext $\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$.
- $\text{Dec}(\text{sk}, \text{ct})$: The decryption algorithm is deterministic. On input a secret key sk and a ciphertext $\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, it outputs either a message in $\mathcal{M}_{\ell_{\text{msg}}}$ or $\perp$.

Correctness. We require that for all $\lambda \in \mathbb{N}$, $\ell_{\text{msg}} \in \mathbb{N}$, $\ell_{\text{st}} \in \mathbb{N}$, and $m \in \mathcal{M}_{\ell_{\text{msg}}}$,

$$\Pr [\text{Dec}(\text{sk}, \text{Enc}(\text{sk}, m)) = m : \text{sk} \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})] \geq 1 - \text{negl}(\lambda),$$

where the probability is over the randomness of **Setup** and **Enc**.

Security. Consider the following experiment with a two-stage interactive adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Leakage-Resilient Incompressible SKE Experiment $\text{Expt}_{\mathcal{A}, \text{SKE}}^{\text{LRIncompSKE}}(\lambda, \ell_{\text{leak}})$:

- **First Phase:**

- $\mathcal{A}_1$ on input 1^λ outputs the message size $1^{\ell_{\text{msg}}}$ and the state size $1^{\ell_{\text{st}}}$. The challenger runs $\text{sk} \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}_1$ is allowed to make polynomially many leakage queries. For the j^{th} query, it submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, where $f_j : \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_j}$. The challenger returns $f_j(\text{sk})$ to $\mathcal{A}_1$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.
- $\mathcal{A}_1$ outputs the challenge messages $m_0, m_1 \in \mathcal{M}_{\ell_{\text{msg}}}$ and auxiliary information **aux**. We assume without loss of generality that **aux** contains m_0 and m_1 .
- The challenger chooses $b \leftarrow \{0, 1\}$, computes $\text{ct}^* \leftarrow \text{Enc}(\text{sk}, m_b)$, and sends ct^* to $\mathcal{A}_1$.
- $\mathcal{A}_1$ computes a state st such that $|\text{st}| \leq \ell_{\text{st}}$ and sends it to the challenger.

- **Second Phase:**

- $\mathcal{A}_2$ receives $(\text{sk}, \text{aux}, \text{st})$ and outputs a bit b' . The adversary wins if $b = b'$.

Definition 41. Let $\ell_{\text{leak}} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. An SKE scheme is said to be ℓ_{leak} -leakage-resilient incompressible secure with respect to the leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left| \Pr[\text{Expt}_{\mathcal{A}, \text{SKE}}^{\text{LRIncompSKE}}(\lambda, \ell_{\text{leak}}) = 1] - \frac{1}{2} \right| \leq \text{negl}(\lambda).$$

Rates. For leakage-resilient incompressible PKE and SKE, the ciphertext rate and compression rate are inherited unchanged from the corresponding incompressible PKE and SKE notions in the preliminaries section. The additional efficiency measure is the leakage rate. For fixed λ and in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, the leakage rate is

$$\text{LeakRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}{\ell_{\text{sk}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}.$$

4.2 Leakage-Resilient Non-Committing Key Encapsulation Mechanism

An LR-NC-KEM NCE = (**Setup**, **Encaps**, **Decaps**, **Sim**₁, **Sim**₂, **Sim**₃) with secret-key space $\{\mathcal{SK}_{\lambda, \ell_k}\}_{\lambda, \ell_k}$, public-key space $\{\mathcal{PK}_{\lambda, \ell_k}\}_{\lambda, \ell_k}$, ciphertext space $\{\mathcal{C}_{\lambda, \ell_k}\}_{\lambda, \ell_k}$, key space $\{\mathcal{K}_{\lambda, \ell_k}\}_{\lambda, \ell_k}$, and leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_k}\}_{\lambda, \ell_k}$ consists of the following algorithms. The algorithms **Setup**, **Encaps**, **Decaps**, and **Sim**₁ are probabilistic polynomial time, whereas **Sim**₂ and **Sim**₃ may be inefficient.

- **Setup**($1^\lambda, 1^{\ell_k}$) : The setup algorithm takes as input the security parameter 1^λ and encapsulation-key size 1^{ℓ_k} and outputs a public key **pk** and a secret key **sk**.

- **Encaps(pk)** : The encapsulation algorithm takes as input a public key $\mathbf{pk}$, produces a key $k \in \{0, 1\}^{\ell_k}$ and a ciphertext $\mathbf{ct}$.
- **Decaps(pk, sk, ct)** : The decapsulation algorithm takes as input a public key $\mathbf{pk}$, a secret key $\mathbf{sk}$, and a ciphertext $\mathbf{ct}$ and outputs $k \in \{0, 1\}^{\ell_k}$.
- **Sim₁(1^λ, 1^{ℓ_k})** : The first simulator takes as input the security parameter 1^λ and encapsulation-key size 1^{ℓ_k} and outputs a public key $\mathbf{pk}$, a ciphertext $\mathbf{ct}$, and a state $\mathbf{state}$.
- **Sim₂(state, f)** : The second inefficient² simulator takes as input a state $\mathbf{state}$ and a function $f : \mathcal{SK}_{\lambda, \ell_k} \rightarrow \{0, 1\}^{\ell}$, and outputs a string $r \in \{0, 1\}^{\ell}$ and an updated state $\mathbf{state}'$.
- **Sim₃(state, k)** : The third inefficient³ simulator takes as input a state $\mathbf{state}$ and a string $k \in \{0, 1\}^{\ell_k}$ and outputs a secret key $\mathbf{sk}$.

Correctness. For all $\lambda \in \mathbb{N}$, $\ell_k \in \mathbb{N}$, and all $(\mathbf{pk}, \mathbf{sk})$ output by $\text{Setup}(1^\lambda, 1^{\ell_k})$,

$$\Pr[(k, \mathbf{ct}) \leftarrow \text{Encaps}(\mathbf{pk}) : \text{Decaps}(\mathbf{pk}, \mathbf{sk}, \mathbf{ct}) = k] \geq 1 - \text{negl}(\lambda).$$

Security. Consider the following two experiments with an adversary $\mathcal{A}$. We note that the simulated experiment samples $k \leftarrow \{0, 1\}^{\ell_k}$, so the indistinguishability requirement below also captures pseudorandomness of the encapsulated key.

Real World.

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ outputs the key size 1^{ℓ_k}. The challenger computes $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{Setup}(1^\lambda, 1^{\ell_k})$ and sends $\mathbf{pk}$ to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make polynomially many leakage queries. For the j^{th} query, $\mathcal{A}$ submits a function $f_j \in \mathcal{F}_{\lambda, \ell_k}$, where $f_j : \mathcal{SK}_{\lambda, \ell_k} \rightarrow \{0, 1\}^{\ell_j}$. The challenger returns $r_j = f_j(\mathbf{sk})$ to $\mathcal{A}$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_k)$. Otherwise, it aborts.
- **Challenge Phase:** The challenger computes $(k, \mathbf{ct}) \leftarrow \text{Encaps}(\mathbf{pk})$ and returns $(\mathbf{sk}, k, \mathbf{ct})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b .

Simulated World.

- **Initialization Phase:** $\mathcal{A}$ on input 1^λ outputs the key size 1^{ℓ_k}. The challenger computes $(\mathbf{pk}, \mathbf{ct}, \mathbf{state}) \leftarrow \text{Sim}_1(1^\lambda, 1^{\ell_k})$ and sends $\mathbf{pk}$ to $\mathcal{A}$.
- **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}$ is allowed to make polynomially many leakage queries. For the j^{th} query, $\mathcal{A}$ submits a function $f_j \in \mathcal{F}_{\lambda, \ell_k}$, where $f_j : \mathcal{SK}_{\lambda, \ell_k} \rightarrow \{0, 1\}^{\ell_j}$. The challenger computes $(r_j, \mathbf{state}') \leftarrow \text{Sim}_2(\mathbf{state}, f_j)$, sets $\mathbf{state} = \mathbf{state}'$, and returns r_j to $\mathcal{A}$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_k)$. Otherwise, it aborts.
- **Challenge Phase:** The challenger samples $k \leftarrow \{0, 1\}^{\ell_k}$ uniformly, computes $\mathbf{sk} \leftarrow \text{Sim}_3(\mathbf{state}, k)$, and returns $(\mathbf{sk}, k, \mathbf{ct})$ to $\mathcal{A}$.
- **Response Phase:** $\mathcal{A}$ outputs b .

² f may not be an efficiently computable function.

³Even if the function f is efficiently computable, it may not be efficiently invertible. The third simulator must generate a simulated secret key $\mathbf{sk}$ such that the leakage values are consistent with the outputs of the second simulator. See Definition 42.

Let p_{real} and p_{sim} be the probabilities with which $\mathcal{A}$ outputs 0 in the real world and simulated world, respectively.

Definition 42. Let $\ell_{\text{leak}} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. A ℓ_{leak} -LR-NC-KEM scheme is said to be secure with respect to the leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_k}\}_{\lambda, \ell_k}$ if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\text{negl}(\cdot)$ such that, for all $\lambda \in \mathbb{N}$,

$$|p_{\text{real}} - p_{\text{sim}}| \leq \text{negl}(\lambda).$$

Rates. For fixed λ , the leakage rate of an ℓ_{leak} -leakage-resilient non-committing key-encapsulation mechanism is

$$\text{LeakRate}(\lambda) = \lim_{\ell_k \rightarrow \infty} \frac{\ell_{\text{leak}}(\lambda, \ell_k)}{\ell_{\text{sk}}(\lambda, \ell_k)}.$$

4.3 Leakage-Resilient Incompressible Signatures

The construction in Section 10 uses the classical random oracle model. All algorithms, adversarial stages, and permitted leakage computations have consistent access to the one global oracle of Section 3.4, including during parameter selection. Messages may be chosen adaptively after seeing the verification key and making oracle queries. The oracle is not reset, and its query transcript is not an additional second-stage input. Correctness and security probabilities include its randomness. Without this oracle, the same syntax and experiment define the plain-model notion. Public verification data and the parameter triple are suppressed inputs to the signing algorithm; they are not part of the secret signing material.

Definition 43 (Leakage-Resilient Incompressible Signatures). An ℓ_{leak} -leakage-resilient incompressible signature scheme Sig with message space $\{\mathcal{M}_{\ell_{\text{msg}}}\}_{\ell_{\text{msg}}}$, signing-key space $\{\mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, verification-key space $\{\mathcal{VK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, signature space $\{\Sigma_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, and leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ consists of a tuple of PPT algorithms $(\text{Gen}, \text{Sign}, \text{Verify})$. As in the original incompressible-signature definition of GWZ [24], we consider message-recovering signatures, where verification outputs either a message or $\perp$.

- $\text{Gen}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The generation algorithm takes as input the security parameter 1^λ , the message size $1^{\ell_{\text{msg}}}$, and the state size $1^{\ell_{\text{st}}}$, and outputs (vk, sk) .
- $\text{Sign}(\text{sk}, m)$: The signing algorithm takes as input the signing key sk and a message m , and outputs a signature σ .
- $\text{Verify}(\text{vk}, \sigma)$: The verification algorithm takes as input the verification key vk and a signature σ , and outputs a message m' or $\perp$.

Correctness. For all security parameters $\lambda, \ell_{\text{st}} \in \mathbb{N}$, all message lengths ℓ_{msg} , and all messages $m \in \mathcal{M}_{\ell_{\text{msg}}}$, over the randomness of $(\text{vk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$ and $\sigma \leftarrow \text{Sign}(\text{sk}, m)$, it holds that

$$\Pr[\text{Verify}(\text{vk}, \sigma) \neq m] \leq \text{negl}(\lambda).$$

Security. Consider the following experiment for an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Leakage-Resilient Incompressible Signature Experiment $\text{Expt}_{\mathcal{A}, \text{Sig}}^{\text{LeakIncompSig}}(\lambda, \ell_{\text{leak}})$:

1. **Initialization Phase:** On input 1^λ , the adversary $\mathcal{A}_1$ specifies the message size $1^{\ell_{\text{msg}}}$ and the state size $1^{\ell_{\text{st}}}$ in unary. The challenger samples $(\text{vk}, \text{sk}) \leftarrow \text{Gen}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$ and provides vk to $\mathcal{A}_1$.

2. **Pre-Challenge Query Phase:** In this phase, $\mathcal{A}_1$ is allowed to make polynomially many leakage queries. For the j^{th} query, $\mathcal{A}_1$ submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, where $f_j : \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_j}$. The challenger provides $f_j(\text{sk})$ to $\mathcal{A}_1$, as long as $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$. At the end of the last round, $\mathcal{A}_1$ provides auxiliary information aux to the challenger.
3. **Signing Query Phase:** For polynomially many rounds, $\mathcal{A}_1$ requests signatures on messages. In the i^{th} query, it submits a message $m_i \in \{0, 1\}^{\ell_{\text{msg}}}$, and the challenger computes $\sigma_i \leftarrow \text{Sign}(\text{sk}, m_i)$ and sends σ_i back to $\mathcal{A}_1$.
4. **First Response Phase:** $\mathcal{A}_1$ sends a state st such that $|\text{st}| \leq \ell_{\text{st}}$.
5. **Second Response Phase:** $\mathcal{A}_2$ receives the verification key vk , the auxiliary information aux , and the state st , and outputs a signature σ' . If $\text{Verify}(\text{vk}, \sigma')$ outputs $\perp$, then the challenger outputs 0. Otherwise, it outputs 1.

Definition 44 (ℓ_{leak} -Leakage-Resilient Incompressible Signatures). Let $\ell_{\text{leak}} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$. A signature scheme Sig is said to be ℓ_{leak} -leakage-resilient incompressible secure with respect to the leakage-class family $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and all $\lambda \in \mathbb{N}$,

$$\Pr[\text{Expt}_{\mathcal{A}, \text{Sig}}^{\text{LeakIncompSig}}(\lambda, \ell_{\text{leak}}) = 1] \leq \text{negl}(\lambda).$$

Rates. Let $\ell_{\text{sign}} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ denote the signature size as a function of λ , ℓ_{msg} , and ℓ_{st} . We define three rates — the signature rate, the compression rate, and the leakage rate.

- signature rate: By convention, we measure the signature rate for fixed λ and in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$:

$$\text{SignRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{msg}}}{\ell_{\text{sign}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}.$$

- compression rate: For fixed λ and ℓ_{msg} , the compression rate is

$$\text{CompRate}(\lambda, \ell_{\text{msg}}) = \lim_{\ell_{\text{st}} \rightarrow \infty} \frac{\ell_{\text{st}}}{\ell_{\text{sign}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}.$$

- leakage rate: For fixed λ and in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, the leakage rate is

$$\text{LeakRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}{\ell_{\text{sk}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}.$$

5 Leakage-Resilient Incompressible SKE against Unbounded Adversaries

In this section, we present leakage-resilient incompressible SKE schemes in the information-theoretic setting. We first give an incompressible SKE scheme with ciphertext-rate $\frac{1}{2}$ and security against unbounded adversaries. The scheme has negligible correctness error. We then show how to transform any information-theoretic incompressible SKE scheme with uniform key generation into a leakage-resilient incompressible SKE scheme.

Throughout this section, incompressibility is considered in the one-time information-theoretic sense: the adversary receives a single challenge ciphertext under a freshly sampled secret key.

5.1 Information-Theoretic Incompressible SKE with Ciphertext-Rate

$\frac{1}{2}$

In this subsection, we build an incompressible SKE scheme with asymptotic ciphertext-rate $\frac{1}{2}$ and security against unbounded adversaries. The construction uses *invertible seeded extractors*. These are extractors for which, given a seed and an output, one can efficiently sample a random preimage from the source domain. A stronger version of this notion was introduced by Cheraghchi *et al.* [9].

Definition 45. A strong average-case (k, ϵ) -extractor $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^m$ is said to be ν -invertible if there exists a probabilistic algorithm Inv , running in time polynomial in d and n , with the following properties. The algorithm Inv takes as input a seed $s \in \{0, 1\}^d$ and a string $o \in \{0, 1\}^m$, and outputs a string $R \in \{0, 1\}^n$. All probabilities and distributions below are over the displayed random choices and the internal randomness of Inv .

1. **Inversion.** The probability that Inv fails to output a string R such that $\text{Ext}_s(R) = o$ is bounded by ν . Namely,

$$\Pr [\text{Ext}_s(R) \neq o : o \leftarrow \{0, 1\}^m, s \leftarrow \{0, 1\}^d, R \leftarrow \text{Inv}(s, o)] \leq \nu.$$

2. **Uniformity.** The distribution obtained by sampling a uniform seed and a uniform output and then inverting is ν -close to the distribution obtained by sampling a uniform seed and a uniform source string. Namely,

$$\Delta((s, \text{Inv}(s, o), o : o \leftarrow \{0, 1\}^m, s \leftarrow \{0, 1\}^d), (s, R, \text{Ext}_s(R) : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d)) \leq \nu.$$

Our incompressible SKE scheme. Let $\text{Ext} : \{0, 1\}^{\ell_{\text{seed}}} \times \{0, 1\}^{\ell_{\text{ct}}} \rightarrow \{0, 1\}^{\ell_{\text{msg}}}$ be a ν -invertible strong average-case (k, ϵ) -extractor with inverter Inv , where $k \leq \ell_{\text{ct}} - \ell_{\text{st}}$. Let $\text{SKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ denote the following incompressible SKE scheme family.

- **Setup** $(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm samples $\text{sk}_1 \leftarrow \{0, 1\}^{\ell_{\text{seed}}}$ and $\text{sk}_2 \leftarrow \{0, 1\}^{\ell_{\text{msg}}}$, and outputs $\text{sk} = (\text{sk}_1, \text{sk}_2)$.
- **Enc** (sk, m) : Let $\text{sk} = (\text{sk}_1, \text{sk}_2)$ be the secret key and let $m \in \{0, 1\}^{\ell_{\text{msg}}}$ be a plaintext message. The encryption algorithm computes $\text{ct} \leftarrow \text{Inv}(\text{sk}_1, m \oplus \text{sk}_2)$ and outputs ct .
- **Dec** (sk, ct) : Let $\text{sk} = (\text{sk}_1, \text{sk}_2)$ be the secret key. The decryption algorithm outputs $\text{Ext}_{\text{sk}_1}(\text{ct}) \oplus \text{sk}_2$.

Correctness. The scheme has correctness error at most ν . Fix any message $m \in \{0, 1\}^{\ell_{\text{msg}}}$. Let $\text{ct} \leftarrow \text{Inv}(\text{sk}_1, m \oplus \text{sk}_2)$ be a ciphertext encrypting m . By the definition of Dec , decryption is correct if and only if $\text{Ext}_{\text{sk}_1}(\text{ct}) = m \oplus \text{sk}_2$.

Since sk_2 is uniformly sampled from $\{0, 1\}^{\ell_{\text{msg}}}$, the string $m \oplus \text{sk}_2$ is also uniformly distributed over $\{0, 1\}^{\ell_{\text{msg}}}$. Moreover, sk_1 is uniformly sampled from $\{0, 1\}^{\ell_{\text{seed}}}$. Therefore, by the inversion property of Inv ,

$$\Pr [\text{Dec}(\text{sk}, \text{Enc}(\text{sk}, m)) \neq m] = \Pr [\text{Ext}_{\text{sk}_1}(\text{Inv}(\text{sk}_1, m \oplus \text{sk}_2)) \neq m \oplus \text{sk}_2] \leq \nu.$$

Hence, for every $m \in \{0, 1\}^{\ell_{\text{msg}}}$, $\Pr[\text{Dec}(\text{sk}, \text{Enc}(\text{sk}, m)) \neq m] \leq \nu$.

Security.

Theorem 46. Assume that $\text{Ext} : \{0, 1\}^{\ell_{\text{seed}}} \times \{0, 1\}^{\ell_{\text{ct}}} \rightarrow \{0, 1\}^{\ell_{\text{msg}}}$ is a ν -invertible strong average-case (k, ϵ) -extractor with inverter Inv , where $k \leq \ell_{\text{ct}} - \ell_{\text{st}}$. Then SKE has correctness error at most ν . Moreover, for every unbounded two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the experiment $\text{Expt}_{\mathcal{A}, \text{SKE}}^{\text{IncompSKE}}(\lambda)$,

$$\left| \Pr[\text{Expt}_{\mathcal{A}, \text{SKE}}^{\text{IncompSKE}}(\lambda) = 1] - \frac{1}{2} \right| \leq \nu + \epsilon.$$

In particular, if $\epsilon = \epsilon(\lambda)$ and $\nu = \nu(\lambda)$ are negligible functions, then SKE is an incompressible SKE scheme with negligible correctness error and security against unbounded adversaries.

Proof. Correctness was shown above. We prove security using a hybrid argument for the experiment $\text{Expt}_{\mathcal{A}, \text{SKE}}^{\text{IncompSKE}}(\lambda)$.

Hybrid G_0 . This is the real experiment.

1. $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger receives (m_0, m_1, aux) from $\mathcal{A}_1$ and samples $b \leftarrow \{0, 1\}$.
3. The challenger samples $\text{sk}_1 \leftarrow \{0, 1\}^{\ell_{\text{seed}}}$ and $\text{sk}_2 \leftarrow \{0, 1\}^{\ell_{\text{msg}}}$.
4. The challenger sends ct to $\mathcal{A}_1$, where $\text{ct} \leftarrow \text{Inv}(\text{sk}_1, m_b \oplus \text{sk}_2)$.
5. $\mathcal{A}_1$ computes a state st from ct such that $|\text{st}| \leq \ell_{\text{st}}$, and sends it to the challenger.
6. The challenger sends $(\text{aux}, \text{st}, (\text{sk}_1, \text{sk}_2))$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs b' .

Hybrid G_1 . This hybrid is obtained from G_0 by rewriting the one-time-pad component.

1. $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger receives (m_0, m_1, aux) from $\mathcal{A}_1$ and samples $b \leftarrow \{0, 1\}$.
3. The challenger samples $\text{sk}_1 \leftarrow \{0, 1\}^{\ell_{\text{seed}}}$ and $\text{sk}_2 \leftarrow \{0, 1\}^{\ell_{\text{msg}}}$.
4. The challenger sends ct to $\mathcal{A}_1$, where $\text{ct} \leftarrow \text{Inv}(\text{sk}_1, \text{sk}_2)$.
5. $\mathcal{A}_1$ computes a state st from ct such that $|\text{st}| \leq \ell_{\text{st}}$, and sends it to the challenger.
6. The challenger sends $(\text{aux}, \text{st}, (\text{sk}_1, \text{sk}_2 \oplus m_b))$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs b' .

Hybrid G_2 . In this hybrid, we use the uniformity of the inverter to switch from inversion sampling to direct extractor sampling.

1. $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger receives (m_0, m_1, aux) from $\mathcal{A}_1$ and samples $b \leftarrow \{0, 1\}$.
3. The challenger samples $\text{sk}_1 \leftarrow \{0, 1\}^{\ell_{\text{seed}}}$ and $\text{ct} \leftarrow \{0, 1\}^{\ell_{\text{ct}}}$, and computes $\text{sk}_2 := \text{Ext}_{\text{sk}_1}(\text{ct})$.
4. The challenger sends ct to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ computes a state st from ct such that $|\text{st}| \leq \ell_{\text{st}}$, and sends it to the challenger.
6. The challenger sends $(\text{aux}, \text{st}, (\text{sk}_1, \text{sk}_2 \oplus m_b))$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs b' .

Hybrid G_3 . This hybrid is obtained from G_2 by replacing the extractor output $\text{sk}_2 = \text{Ext}_{\text{sk}_1}(\text{ct})$ with a uniform string $u \leftarrow \{0, 1\}^{\ell_{\text{msg}}}$.

1. $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger receives (m_0, m_1, aux) from $\mathcal{A}_1$ and samples $b \leftarrow \{0, 1\}$.
3. The challenger samples $\text{sk}_1 \leftarrow \{0, 1\}^{\ell_{\text{seed}}}$, $\text{ct} \leftarrow \{0, 1\}^{\ell_{\text{ct}}}$ and $u \leftarrow \{0, 1\}^{\ell_{\text{msg}}}$.

4. The challenger sends ct to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ computes a state st from ct such that $|\text{st}| \leq \ell_{\text{st}}$, and sends it to the challenger.
6. The challenger sends $(\text{aux}, \text{st}, (\text{sk}_1, u \oplus m_b))$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs b' .

Let $p_{\mathcal{A}, G_i}$ denote the probability that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputs $b' = b$ in Hybrid G_i .

First, $p_{\mathcal{A}, G_0} = p_{\mathcal{A}, G_1}$. Indeed, fix the challenge bit b and define $t := \text{sk}_2 \oplus m_b$ in Hybrid G_0 . Since xor with the fixed string m_b is a bijection on $\{0, 1\}^{\ell_{\text{msg}}}$, the variable t is uniform over $\{0, 1\}^{\ell_{\text{msg}}}$. Rewriting Hybrid G_0 in terms of t , the sampled tuple becomes

$$(\text{sk}_1, t, \text{ct}, \text{aux}, \text{st}, t \oplus m_b), \quad \text{where} \quad \text{sk}_1 \leftarrow \{0, 1\}^{\ell_{\text{seed}}}, t \leftarrow \{0, 1\}^{\ell_{\text{msg}}}, \text{ct} \leftarrow \text{Inv}(\text{sk}_1, t),$$

with st computed from ct . This is exactly the joint distribution of Hybrid G_1 after renaming t as sk_2 .

Second, $|p_{\mathcal{A}, G_1} - p_{\mathcal{A}, G_2}| \leq \nu$. Hybrid G_1 uses the correlation $(\text{sk}_1, \text{ct}, \text{sk}_2)$ where sk_1 and sk_2 are uniformly sampled from their respective domains, and $\text{ct} \leftarrow \text{Inv}(\text{sk}_1, \text{sk}_2)$. Hybrid G_2 uses the correlation $(\text{sk}_1, \text{ct}, \text{sk}_2)$ where sk_1 and ct are uniformly sampled from their respective domains, and $\text{sk}_2 = \text{Ext}_{\text{sk}_1}(\text{ct})$. This is the only difference between the two hybrids. By the uniformity property of the invertible seeded extractor, the statistical distance between these distributions is at most ν .

Third, $|p_{\mathcal{A}, G_2} - p_{\mathcal{A}, G_3}| \leq \epsilon$. Fix the pre-challenge output (m_0, m_1, aux) , the challenge bit b , and the coins of the adversary. In Hybrid G_2 , the ciphertext ct is uniform over $\{0, 1\}^{\ell_{\text{ct}}}$, and st is a randomized function of ct with output length at most ℓ_{st} . Hence, by Lemma 9,

$$\tilde{H}_{\infty}(\text{ct} \mid \text{st}) \geq \ell_{\text{ct}} - \ell_{\text{st}} \geq k.$$

Since Ext is a strong average-case (k, ϵ) -extractor,

$$(\text{sk}_1, \text{st}, \text{Ext}_{\text{sk}_1}(\text{ct})) \approx_{\epsilon} (\text{sk}_1, \text{st}, U_{\ell_{\text{msg}}}).$$

Xoring the last component with the fixed string m_b and applying the remaining computation of $\mathcal{A}_2$ are post-processing operations. Therefore, $|p_{\mathcal{A}, G_2} - p_{\mathcal{A}, G_3}| \leq \epsilon$.

Finally, $p_{\mathcal{A}, G_3} = 1/2$. In Hybrid G_3 , the string $u \oplus m_b$ is uniformly distributed over $\{0, 1\}^{\ell_{\text{msg}}}$ and is independent of b . Moreover, ct and st are independent of b except through values masked by the uniform string u . Therefore, the entire view of $\mathcal{A}_2$ is independent of b .

Using the triangle inequality,

$$\left| p_{\mathcal{A}, G_0} - \frac{1}{2} \right| \leq |p_{\mathcal{A}, G_0} - p_{\mathcal{A}, G_1}| + |p_{\mathcal{A}, G_1} - p_{\mathcal{A}, G_2}| + |p_{\mathcal{A}, G_2} - p_{\mathcal{A}, G_3}| + \left| p_{\mathcal{A}, G_3} - \frac{1}{2} \right| \leq \nu + \epsilon.$$

This proves the theorem. $\square$

We instantiate seeded invertible extractors using pairwise independent hash functions.

Theorem 47 (Average-Case Leftover Hash Lemma). *Let $\mathcal{H} = h : \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a pairwise independent family of hash functions. If $m \leq k - 2 \log(1/\epsilon)$, then $\text{Ext}_h(x) := h(x)$ is a strong average-case (k, ϵ) -extractor.*

Proposition 48. *Let $\lambda, \ell_{\text{msg}}, \ell_{\text{st}} \in \mathbb{N}$, let $N := \max\{\ell_{\text{st}}, \ell_{\text{msg}} + 2\lambda\}$, let $\mathbb{F}_{2^N}$ be the finite field of order 2^N , fix an $\mathbb{F}_2$ -basis of $\mathbb{F}_{2^N}$, identify each field element with its N -bit coefficient vector in this basis, and let $\pi : \mathbb{F}_{2^N} \rightarrow \{0, 1\}^{\ell_{\text{msg}}}$ be the projection onto the first ℓ_{msg} coordinates. For inputs $(x_1, x_2) \in \mathbb{F}_{2^N}^2$ and a seed (a_1, a_2, b) sampled uniformly from $\mathbb{F}_{2^N}^3$, define*

$$\text{Ext}_{(a_1, a_2, b)}(x_1, x_2) := \pi(a_1 x_1 + a_2 x_2 + b).$$

Then Ext is a ν -invertible strong average-case $(\ell_{\text{msg}} + 2\lambda, 2^{-\lambda})$ seeded extractor with $\nu \leq 2^{-2N}$.

Proof. For every $a_1, a_2, b \in \mathbb{F}_{2^N}$, define $h_{a_1, a_2, b}(x_1, x_2) := \pi(a_1 x_1 + a_2 x_2 + b)$. Then

$$\{h_{a_1, a_2, b} : \mathbb{F}_{2^N}^2 \rightarrow \{0, 1\}^{\ell_{\text{msg}}}\}_{a_1, a_2, b \in \mathbb{F}_{2^N}}$$

is a pairwise independent family of hash functions. Therefore, by Theorem 47, Ext is a strong average-case $(\ell_{\text{msg}} + 2\lambda, 2^{-\lambda})$ extractor.

It remains to prove invertibility. Given $o \in \{0, 1\}^{\ell_{\text{msg}}}$ and (a_1, a_2, b) , first sample a uniform $y \in \mathbb{F}_{2^N}$ conditioned on $\pi(y) = o$, which can be done by fixing the first ℓ_{msg} coordinates to equal o and choosing the remaining $N - \ell_{\text{msg}}$ coordinates uniformly. If $(a_1, a_2) \neq (0, 0)$, the set

$$\{(x_1, x_2) \in \mathbb{F}_{2^N}^2 : a_1 x_1 + a_2 x_2 + b = y\}$$

is nonempty and has size exactly 2^N . In this case, one can efficiently sample uniformly from this affine subspace.

If $(a_1, a_2) = (0, 0)$, inversion may fail. This bad seed event has probability exactly 2^{-2N} . Hence the inversion failure probability is at most $\nu \leq 2^{-2N}$.

Moreover, conditioned on $(a_1, a_2) \neq (0, 0)$, every set $\pi^{-1}(o)$ has size exactly $2^{N - \ell_{\text{msg}}}$ and every fiber

$$\{(x_1, x_2) \in \mathbb{F}_{2^N}^2 : a_1 x_1 + a_2 x_2 + b = y\}$$

has size exactly 2^N . Hence the above procedure samples uniformly from the full preimage of o under $h_{a_1, a_2, b}$. In addition, conditioned on $(a_1, a_2) \neq (0, 0)$, the map $(x_1, x_2) \mapsto a_1 x_1 + a_2 x_2 + b$ is surjective with all fibers of size 2^N , so sampling a uniform source (x_1, x_2) and then applying $h_{a_1, a_2, b}$ induces exactly the same joint distribution as sampling a uniform output o and then inverting. Thus the uniformity property holds perfectly conditioned on the good seed event. The statistical loss from the bad seed event is at most 2^{-2N} . Therefore, Ext is ν -invertible with $\nu \leq 2^{-2N}$. $\square$

Parameters. Appealing to Proposition 48, with $N := \max\{\ell_{\text{st}}, \ell_{\text{msg}} + 2\lambda\}$, we get a ν -invertible strong average-case $(\ell_{\text{msg}} + 2\lambda, 2^{-\lambda})$ extractor with $\nu \leq 2^{-2N}$, seed length $3N$, source length $2N$, and output length ℓ_{msg} . Using this extractor in our construction, the ciphertext size is $\ell_{\text{ct}} = 2N$ and the secret-key size is $3N + \ell_{\text{msg}}$.

Since $N \geq \ell_{\text{msg}} + 2\lambda$, the bound $\nu \leq 2^{-2N}$ is negligible as a function of λ . The extractor security requires $\ell_{\text{ct}} - \ell_{\text{st}} \geq \ell_{\text{msg}} + 2\lambda$. Here,

$$\ell_{\text{ct}} - \ell_{\text{st}} = 2N - \ell_{\text{st}} \geq N \geq \ell_{\text{msg}} + 2\lambda,$$

so this condition is always satisfied. Also, since $N = \max\{\ell_{\text{st}}, \ell_{\text{msg}} + 2\lambda\}$, we have

$$3N + \ell_{\text{msg}} \leq 4 \max\{\ell_{\text{st}}, \ell_{\text{msg}}\} + O(\lambda).$$

Rates. In the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, we have $N = \ell_{\text{msg}} + 2\lambda$, and hence $\ell_{\text{ct}} = 2\ell_{\text{msg}} + 4\lambda$. Therefore, for fixed λ , the asymptotic ciphertext-rate is $\frac{1}{2}$.

For fixed λ and ℓ_{msg} and sufficiently large ℓ_{st} , we have $N = \ell_{\text{st}}$ and hence $\ell_{\text{ct}} = 2\ell_{\text{st}}$. Therefore, the asymptotic compression-rate is $\frac{1}{2}$.

Corollary 49. *There exists an incompressible SKE scheme with uniform key generation, correctness error $\text{negl}(\lambda)$, ciphertext-rate $\frac{1}{2}$, compression-rate $\frac{1}{2}$, secret-key size $3N + \ell_{\text{msg}} \leq 4 \max\{\ell_{\text{st}}, \ell_{\text{msg}}\} + O(\lambda)$, and security against unbounded adversaries, where $N := \max\{\ell_{\text{st}}, \ell_{\text{msg}} + 2\lambda\}$.*

5.2 Generic Transformation

In this subsection, we give a generic transformation that takes any information-theoretic incompressible SKE scheme with uniform key generation and produces a leakage-resilient incompressible SKE scheme. The transformation uses strong average-case extractors to derive keys for the base incompressible SKE scheme from the secret key of the transformed scheme.

Here we restrict attention to leakage classes consisting of all output-length-bounded functions. This is exactly the setting needed in the proof below, because any polynomially many-query leakage phase with total leakage at most the admissible bound can then be aggregated into a single leakage function that outputs the full leakage transcript.

Our construction. We require the following primitives.

- Let $\text{IncSKE} = (\text{IncSKE.Setup}, \text{IncSKE.Enc}, \text{IncSKE.Dec})$ be an incompressible secret-key encryption scheme for message length ℓ_{msg} , state size ℓ_{st} , ciphertext length $\ell_{\text{ct}}^{\text{IncSKE}}$, and secret-key length $\ell_{\text{sk}}^{\text{IncSKE}}$. Assume that IncSKE.Setup samples its secret key uniformly from $\{0, 1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$. Let η_{corr} denote the correctness error of IncSKE over its key-generation and encryption randomness.
- Fix a constant $c > 0$, and define $K_{\text{Ext}} := \ell_{\text{sk}}^{\text{IncSKE}} + c\lambda$.
- Let $\ell_{\text{sk}} \geq K_{\text{Ext}}$ be the secret-key length of the transformed scheme.
- Let $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$ be a strong average-case $(K_{\text{Ext}}, \epsilon_{\text{Ext}})$ -extractor. By Corollary 14, we may instantiate such an extractor with error $\epsilon_{\text{Ext}} = \text{negl}(\lambda)$ and seed length

$$d = O(\log(K_{\text{Ext}} + \lambda) \cdot (\log \ell_{\text{sk}} + \lambda)).$$

Define the admissible leakage bound by

$$\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) := \ell_{\text{sk}} - K_{\text{Ext}} = \ell_{\text{sk}} - (\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda).$$

For each parameter triple $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, let

$$\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}^{\text{obl}} := \bigcup_{0 \leq y \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})} \{h \mid h : \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^y\}.$$

Let $\mathcal{F}_{\text{obl}} = \left\{ \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}^{\text{obl}} \right\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ denote the resulting output-length-bounded leakage-class family.

We define a leakage-resilient incompressible SKE scheme as follows.

- **Setup** $(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm samples $\text{sk} \leftarrow \{0, 1\}^{\ell_{\text{sk}}}$ and outputs sk .
- **Enc** (sk, m) : Let sk be the secret key and let $m \in \{0, 1\}^{\ell_{\text{msg}}}$ be a message. The encryption algorithm samples $s \leftarrow \{0, 1\}^d$, computes $k := \text{Ext}_s(\text{sk})$, and outputs $\text{ct} := (s, \text{IncSKE.Enc}(k, m))$.
- **Dec** (sk, ct) : Let sk be the secret key and let $\text{ct} = (s', \text{ct}')$ be a ciphertext. The decryption algorithm computes $k := \text{Ext}_{s'}(\text{sk})$, decrypts ct' using k by computing $m := \text{IncSKE.Dec}(k, \text{ct}')$, and outputs m .

Correctness. An encryption of a message m under a secret key sk is $\text{ct} = (s, \text{IncSKE.Enc}(k, m))$, where $s \leftarrow \{0, 1\}^d$ and $k = \text{Ext}_s(\text{sk})$. To decrypt this ciphertext using sk , the decryption algorithm recomputes $k' = \text{Ext}_s(\text{sk}) = k$ and returns $\text{IncSKE.Dec}(k, \text{IncSKE.Enc}(k, m))$.

Since sk is uniform over $\{0, 1\}^{\ell_{\text{sk}}}$, we have $H_\infty(\text{sk}) = \ell_{\text{sk}} \geq K_{\text{Ext}}$. Therefore, the extractor guarantee gives

$$(s, \text{Ext}_s(\text{sk})) \approx_{\epsilon_{\text{Ext}}} (s, U_{\ell_{\text{sk}}^{\text{IncSKE}}}).$$

Hence, the correctness experiment of the transformed scheme is ϵ_{Ext} -close to the correctness experiment of IncSKE under a uniform base key. Therefore, the transformed scheme has correctness error at most $\eta_{\text{corr}} + \epsilon_{\text{Ext}}$.

Parameters. By Corollary 14, the extractor seed length is $d = O(\log(K_{\text{Ext}} + \lambda) \cdot (\log \ell_{\text{sk}} + \lambda))$. Hence, the ciphertext length of the transformed scheme is $\ell_{\text{ct}}^{\text{IncSKE}} + d$. The state size in the transformed scheme is the same as the state size for IncSKE, since the security reduction relies on the incompressibility of IncSKE. For any leakage function $h \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}^{\text{obl}}$, the output length of h is at most $\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$. Therefore,

$$\tilde{H}_{\infty}(\text{sk} \mid h(\text{sk})) \geq \ell_{\text{sk}} - \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = K_{\text{Ext}}.$$

Hence the strong average-case extractor guarantee implies that $(s, \text{Ext}_s(\text{sk}), h(\text{sk}))$ is statistically close to $(s, U_{\ell_{\text{sk}}^{\text{IncSKE}}}, h(\text{sk}))$. Thus the key derived for IncSKE is statistically close to uniform even given the leakage value $h(\text{sk})$ produced by the adversary.

Rates. The ciphertext-rate and compression-rate are inherited unchanged from the underlying incompressible SKE scheme IncSKE, exactly as in the general definition from Definition 41. For fixed λ and in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, the transformed ciphertext length is

$$\ell_{\text{ct}}^{\text{IncSKE}} + O(\log(K_{\text{Ext}} + \lambda) \cdot (\log \ell_{\text{sk}} + \lambda)) = \ell_{\text{ct}}^{\text{IncSKE}} + o(\ell_{\text{msg}}),$$

so the transformed scheme has the same asymptotic ciphertext-rate as IncSKE whenever the displayed polylogarithmic seed length is $o(\ell_{\text{msg}})$, as it is for the instantiations used below. For fixed λ and ℓ_{msg} , the same seed bound is $o(\ell_{\text{st}})$ whenever ℓ_{sk} and K_{Ext} are polynomially bounded in ℓ_{st} , so the transformed scheme also has the same asymptotic compression-rate as IncSKE. The leakage rate is

$$\text{LeakRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}{\ell_{\text{sk}}} = \lim_{\ell_{\text{msg}} \rightarrow \infty} \left(1 - \frac{\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda}{\ell_{\text{sk}}} \right).$$

In particular, if $\ell_{\text{sk}} = \omega(\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda)$ along this regime, then the leakage rate equals 1, and in particular is $1 - o(1)$.

Lemma 50. Let $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^n \rightarrow \{0, 1\}^t$ be a strong average-case (δ, ϵ) -extractor. Let $h : \{0, 1\}^n \rightarrow \{0, 1\}^y$ be any function such that $\tilde{H}_{\infty}(R \mid h(R)) \geq \delta$, when R is uniform over $\{0, 1\}^n$. For $z \in \{0, 1\}^y$, $s \in \{0, 1\}^d$, and $u \in \{0, 1\}^t$, define

$$\text{Fib}(z, s, u) := \{r \in \{0, 1\}^n : \text{Ext}_s(r) = u, h(r) = z\}.$$

Let $\text{Samp}(z, s, u)$ denote the randomized procedure that outputs $\perp$ if $\text{Fib}(z, s, u) = \emptyset$, and otherwise outputs a uniformly random element of $\text{Fib}(z, s, u)$. Then, we have

$$(R, h(R), s, \text{Ext}_s(R) : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d) \approx_{\epsilon} \left(\hat{R}_{\perp}, h(R), s, U_t : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d, U_t \leftarrow \{0, 1\}^t, \hat{R}_{\perp} \leftarrow \text{Samp}(h(R), s, U_t) \right).$$

Moreover, in the right-hand distribution, $\Pr[\hat{R}_{\perp} = \perp] \leq \epsilon$.

Proof. We first observe that for any function $g : \{0, 1\}^n \rightarrow \{0, 1\}^m$,

$$(R, g(R) : R \leftarrow \{0, 1\}^n) \equiv \left(\hat{R}, g(R) : R \leftarrow \{0, 1\}^n, \hat{R} \leftarrow \{r \in \{0, 1\}^n : g(r) = g(R)\} \right).$$

The above states that computing $g(R)$ after randomly selecting a string R is the same as computing $g(R)$ for a randomly selected R and then sampling a uniformly random preimage of $g(R)$. Applying the same logic to the function $r \mapsto (h(r), \text{Ext}_s(r))$, with s fixed and then averaged over, we have

$$(R, h(R), s, \text{Ext}_s(R) : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d) \equiv \left(\hat{R}, h(R), s, \text{Ext}_s(R) : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d, \hat{R} \leftarrow \{r \in \{0, 1\}^n : \text{Ext}_s(r) = \text{Ext}_s(R), h(r) = h(R)\} \right).$$

From the fact that Ext is a strong average-case (δ, ϵ) -extractor, and $\tilde{H}_\infty(R \mid h(R)) \geq \delta$ when $R \leftarrow \{0, 1\}^n$, it follows that

$$(h(R), s, \text{Ext}_s(R) : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d) \approx_\epsilon (h(R), s, U_t : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d, U_t \leftarrow \{0, 1\}^t).$$

Let F be the randomized map that, on input (z, s, u) , outputs $(\text{Samp}(z, s, u), z, s, u)$. This map is well-defined on every input because it returns $\perp$ when the corresponding fiber is empty. By the post-processing property of statistical distance, the previous display implies

$$\left(\hat{R}, h(R), s, \text{Ext}_s(R) : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d, \hat{R} \leftarrow \{r \in \{0, 1\}^n : \text{Ext}_s(r) = \text{Ext}_s(R), h(r) = h(R)\} \right) \approx_\epsilon \left(\hat{R}_\perp, h(R), s, U_t : R \leftarrow \{0, 1\}^n, s \leftarrow \{0, 1\}^d, U_t \leftarrow \{0, 1\}^t, \hat{R}_\perp \leftarrow \text{Samp}(h(R), s, U_t) \right).$$

In the left-hand distribution above, the fiber is always nonempty because it contains the sampled string R . Hence $\Pr[\hat{R}_\perp = \perp] = 0$ on the left. Since the two distributions are ϵ -close, it follows that $\Pr[\hat{R}_\perp = \perp] \leq \epsilon$ on the right. This proves the lemma. $\square$

Theorem 51. *Let $\text{IncSKE} = (\text{IncSKE.Setup}, \text{IncSKE.Enc}, \text{IncSKE.Dec})$ be an incompressible secret-key encryption scheme secure against unbounded adversaries, and suppose that IncSKE.Setup samples its secret key uniformly from $\{0, 1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$. Let η_{corr} denote the correctness error of IncSKE over its key-generation and encryption randomness.*

Fix a constant $c > 0$, set $K_{\text{Ext}} := \ell_{\text{sk}}^{\text{IncSKE}} + c\lambda$, and assume that $\ell_{\text{sk}} \geq K_{\text{Ext}}$. Let $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$ be a strong average-case $(K_{\text{Ext}}, \epsilon_{\text{Ext}})$ -extractor with

$$d = O(\log(K_{\text{Ext}} + \lambda) \cdot (\log \ell_{\text{sk}} + \lambda))$$

and $\epsilon_{\text{Ext}} = \text{negl}(\lambda)$.

Then the above construction is an ℓ_{leak} -leakage-resilient incompressible secret-key encryption scheme against unbounded adversaries with respect to the leakage-class family $\mathcal{F}_{\text{obl}}$, where

$$\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = \ell_{\text{sk}} - K_{\text{Ext}} = \ell_{\text{sk}} - (\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda).$$

Its correctness error is at most $\eta_{\text{corr}} + \epsilon_{\text{Ext}}$. If $d = o(\ell_{\text{msg}})$ in the fixed- λ diagonal regime and $d = o(\ell_{\text{st}})$ in the fixed- $(\lambda, \ell_{\text{msg}})$ large-state regime, then its ciphertext-rate and compression-rate are the same as those of IncSKE . Furthermore, if $\ell_{\text{sk}} = \omega(\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda)$ in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, then the leakage rate equals 1, and in particular is $1 - o(1)$.

Proof. Our proof proceeds via a sequence of hybrids. We first describe the hybrids and then bound the statistical distance between successive games.

As in the leakage-resilient incompressible SKE definition, the second-stage adversary receives $(\text{sk}, \text{aux}, \text{st})$.

By definition of $\mathcal{F}_{\text{obl}}$, we may, without loss of generality, assume that $\mathcal{A}_1$ makes a single leakage query $h : \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^y$, where $y \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$. This single function outputs the full transcript of any polynomially many-query leakage phase whose total leakage length is at most $\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.

For $z \in \{0, 1\}^y$, $s \in \{0, 1\}^d$, and $k \in \{0, 1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$, define

$$\text{Fib}(z, s, k) := \{r \in \{0, 1\}^{\ell_{\text{sk}}} : h(r) = z \text{ and } \text{Ext}_s(r) = k\}.$$

Hybrid G_0 . This is the real leakage-resilient incompressible SKE security game for the above construction.

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger samples $\text{sk} \leftarrow \{0, 1\}^{\ell_{\text{sk}}}$.

- **Pre-challenge Query Phase:**

1. The first adversary $\mathcal{A}_1$ submits a leakage query $h : \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^y$, where $y \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, and the challenger returns $h(\text{sk})$.

- **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) .
2. The challenger samples $s \leftarrow \{0, 1\}^d$ and $b \leftarrow \{0, 1\}$.
3. It computes $k := \text{Ext}_s(\text{sk})$ and $\text{ct}^* \leftarrow \text{IncSKE.Enc}(k, m_b)$.
4. It sends (s, ct^*) to $\mathcal{A}_1$.

- **First Response Phase:**

1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.

- **Second Response Phase:**

1. The second adversary $\mathcal{A}_2$ receives $(\text{sk}, \text{aux}, \text{st})$.
2. Finally, $\mathcal{A}_2$ outputs b' .

Hybrid G_1 . This game is obtained from G_0 by sampling the leakage value and the base key as in the right-hand distribution of Lemma 50, and by reconstructing a consistent master key only in the second response phase.

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger samples $\text{sk}' \leftarrow \{0, 1\}^{\ell_{\text{sk}}}$.

- **Pre-challenge Query Phase:**

1. The first adversary $\mathcal{A}_1$ submits a leakage query $h : \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^y$, where $y \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, and the challenger returns $z := h(\text{sk}')$.

- **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) .
2. The challenger samples $s \leftarrow \{0, 1\}^d$, $u \leftarrow \{0, 1\}^{\ell_{\text{IncSKE}}}$, and $b \leftarrow \{0, 1\}$.
3. It computes $\text{ct}^* \leftarrow \text{IncSKE.Enc}(u, m_b)$.
4. It sends (s, ct^*) to $\mathcal{A}_1$.

- **First Response Phase:**

1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.

- **Second Response Phase:**

1. The challenger considers the fiber $\text{Fib}(z, s, u)$.
2. If this fiber is empty, the challenger samples $b' \leftarrow \{0, 1\}$.
3. Otherwise, it samples sk uniformly from $\text{Fib}(z, s, u)$, gives $(\text{sk}, \text{aux}, \text{st})$ to $\mathcal{A}_2$, and receives b' from $\mathcal{A}_2$.

Hybrid G_2 . This game is identical to G_1 , except that the base key is generated by running the setup algorithm of IncSKE .

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger samples $\text{sk}' \leftarrow \{0, 1\}^{\ell_{\text{sk}}}$.
3. The challenger runs $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.

• **Pre-challenge Query Phase:**

1. The first adversary $\mathcal{A}_1$ submits a leakage query $h : \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^y$, where $y \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, and the challenger returns $z := h(\text{sk}')$.

• **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) .
2. The challenger samples $s \leftarrow \{0, 1\}^d$ and $b \leftarrow \{0, 1\}$.
3. It computes $\text{ct}^* \leftarrow \text{IncSKE.Enc}(\text{inc.sk}, m_b)$.
4. It sends (s, ct^*) to $\mathcal{A}_1$.

• **First Response Phase:**

1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.

• **Second Response Phase:**

1. The challenger considers the fiber $\text{Fib}(z, s, \text{inc.sk})$.
2. If this fiber is empty, the challenger samples $b' \leftarrow \{0, 1\}$.
3. Otherwise, it samples sk uniformly from $\text{Fib}(z, s, \text{inc.sk})$, gives $(\text{sk}, \text{aux}, \text{st})$ to $\mathcal{A}_2$, and receives b' from $\mathcal{A}_2$.

Let $p_{\mathcal{A}, G_i}$ denote the probability that game G_i outputs $b' = b$.

Lemma 52. *For every unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$|p_{\mathcal{A}, G_0} - p_{\mathcal{A}, G_1}| \leq \epsilon_{\text{Ext}}.$$

Proof. Since $\text{sk} \leftarrow \{0, 1\}^{\ell_{\text{sk}}}$ is uniform and h has output length y , Lemma 9 gives

$$\tilde{H}_\infty(\text{sk} \mid h(\text{sk})) \geq \ell_{\text{sk}} - y \geq \ell_{\text{sk}} - \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) \geq K_{\text{Ext}}.$$

Therefore, Lemma 50 applies to h . Hence,

$$\begin{aligned} & (\text{sk}, h(\text{sk}), s, \text{Ext}_s(\text{sk}) : \text{sk} \leftarrow \{0, 1\}^{\ell_{\text{sk}}}, s \leftarrow \{0, 1\}^d) \approx_{\epsilon_{\text{Ext}}} \\ & \left(\begin{array}{c} \text{sk}' \leftarrow \{0, 1\}^{\ell_{\text{sk}}}, s \leftarrow \{0, 1\}^d, \\ \widehat{\text{sk}}_\perp, h(\text{sk}'), s, U_{\ell_{\text{IncSKE}}} : U_{\ell_{\text{IncSKE}}} \leftarrow \{0, 1\}^{\ell_{\text{sk}}}, \\ \widehat{\text{sk}}_\perp \leftarrow \text{Samp}(h(\text{sk}'), s, U_{\ell_{\text{IncSKE}}}) \end{array} \right), \end{aligned}$$

where $\text{Samp}(z, s, k)$ denotes the sampler that samples uniformly from

$$\{r \in \{0, 1\}^{\ell_{\text{sk}}} : h(r) = z \text{ and } \text{Ext}_s(r) = k\}$$

if this set is nonempty, and outputs $\perp$ otherwise. The first coordinate is interpreted over the extended space $\{0, 1\}^{\ell_{\text{sk}}} \cup \{\perp\}$.

Now apply the following randomized post-processing to both distributions. Given a tuple (R, z, s, k) , the post-processing continues the execution of $\mathcal{A}_1$ after giving it leakage answer z , obtains (m_0, m_1, aux) , samples the challenge bit $b \leftarrow \{0, 1\}$, computes $\text{ct}^* \leftarrow \text{IncSKE.Enc}(k, m_b)$, gives (s, ct^*) to $\mathcal{A}_1$, and obtains a state st . If $R = \perp$, it outputs the win indicator of an

independent uniform guess. If $R \neq \perp$, it runs $\mathcal{A}_2$ on $(R, \text{aux}, \text{st})$, obtains b' , and outputs the win indicator $\mathbf{1}[b' = b]$.

Applied to the left-hand distribution, this post-processing gives exactly the win indicator in G_0 . Applied to the right-hand distribution, it gives exactly the win indicator in G_1 .

Since statistical distance cannot increase under randomized post-processing, the claim follows. $\square$

Lemma 53. *For every unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$,*

$$p_{\mathcal{A}, G_1} = p_{\mathcal{A}, G_2}.$$

Proof. The only difference between G_1 and G_2 is that in G_1 the challenger samples $u \leftarrow \{0, 1\}_{\ell_{\text{sk}}^{\text{IncSKE}}}$, whereas in G_2 it samples $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$. By assumption, IncSKE.Setup samples its secret key uniformly from $\{0, 1\}_{\ell_{\text{sk}}^{\text{IncSKE}}}$. Therefore, the two games are identically distributed. $\square$

Lemma 54. *Assume IncSKE is incompressible against unbounded adversaries. Then, for every unbounded adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that*

$$p_{\mathcal{A}, G_2} \leq \frac{1}{2} + \text{negl}(\lambda).$$

Proof. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an unbounded adversary against the transformed scheme. We construct an unbounded adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against IncSKE.

- **Initialization Phase:** $\mathcal{B}_1$ receives the message size ℓ_{msg} and the state size ℓ_{st} from $\mathcal{A}_1$ and relays them to the incompressibility challenger for IncSKE. It samples a dummy master key $\text{sk}' \leftarrow \{0, 1\}_{\ell_{\text{sk}}}$.
- **Pre-challenge Query Phase:** When $\mathcal{A}_1$ submits its leakage query $h : \{0, 1\}_{\ell_{\text{sk}}} \rightarrow \{0, 1\}^y$, where $y \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, $\mathcal{B}_1$ computes $z := h(\text{sk}')$ and returns z to $\mathcal{A}_1$.
- **Challenge Phase:** When $\mathcal{A}_1$ outputs (m_0, m_1, aux) , $\mathcal{B}_1$ samples $s \leftarrow \{0, 1\}^d$, sends (m_0, m_1, aux_B) to the incompressibility challenger for IncSKE, where $\text{aux}_B := (\text{aux}, s, h, z)$. When $\mathcal{B}_1$ receives the base challenge ciphertext ct^* , it sends (s, ct^*) to $\mathcal{A}_1$.
- **First Response Phase:** When $\mathcal{A}_1$ outputs a state st with $|\text{st}| \leq \ell_{\text{st}}$, $\mathcal{B}_1$ sends st to the challenger.
- **Second Response Phase:** $\mathcal{B}_2$ receives $(\text{inc.sk}, \text{aux}_B, \text{st})$ from the challenger, where inc.sk is the secret key used to generate ct^* . It parses $\text{aux}_B = (\text{aux}, s, h, z)$. If $\text{Fib}(z, s, \text{inc.sk})$ is empty, $\mathcal{B}_2$ outputs an independent uniform bit. Otherwise, it samples sk uniformly from $\text{Fib}(z, s, \text{inc.sk})$, runs $\mathcal{A}_2$ on $(\text{sk}, \text{aux}, \text{st})$, and outputs whatever $\mathcal{A}_2$ outputs.

The view of $\mathcal{A}$ in this reduction is distributed identically to its view in G_2 . The value $z = h(\text{sk}')$ is the simulated leakage response, the key inc.sk is the uniform base secret key used by the incompressibility challenger, the challenge ciphertext is generated exactly as in G_2 , and the second-stage adversary receives the secret key sampled from the corresponding fiber, with the empty-fiber case handled by an independent uniform guess. Therefore $\mathcal{B}$ wins the incompressibility game for IncSKE with probability $p_{\mathcal{A}, G_2}$. By the security of IncSKE, this probability is at most $1/2 + \text{negl}(\lambda)$. $\square$

By Lemma 52, Lemma 53, Lemma 54, and the triangle inequality, for every unbounded adversary $\mathcal{A}$, there exists a negligible function negl such that

$$\left| p_{\mathcal{A}, G_0} - \frac{1}{2} \right| \leq \epsilon_{\text{Ext}} + \text{negl}(\lambda).$$

Since $\epsilon_{\text{Ext}} = \text{negl}(\lambda)$, the transformed scheme is leakage-resilient incompressible against unbounded adversaries.

Correctness follows from the extractor guarantee and the correctness of **IncSKE**. Indeed, for uniform $\text{sk} \leftarrow \{0, 1\}^{\ell_{\text{sk}}}$ and $s \leftarrow \{0, 1\}^d$, since $\ell_{\text{sk}} \geq K_{\text{Ext}}$, the extractor guarantee gives

$$(s, \text{Ext}_s(\text{sk})) \approx_{\epsilon_{\text{Ext}}} (s, U_{\ell_{\text{sk}}^{\text{IncSKE}}}).$$

Thus, the correctness error of the transformed scheme differs from the correctness error of **IncSKE** by at most ϵ_{Ext} , and is therefore at most $\eta_{\text{corr}} + \epsilon_{\text{Ext}}$.

Finally, the leakage-rate claim follows from

$$\text{LeakRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}{\ell_{\text{sk}}} = \lim_{\ell_{\text{msg}} \rightarrow \infty} \left(1 - \frac{K_{\text{Ext}}}{\ell_{\text{sk}}}\right) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \left(1 - \frac{\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda}{\ell_{\text{sk}}}\right),$$

which equals 1 whenever $\ell_{\text{sk}} = \omega(\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda)$ in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$. The ciphertext of the transformed scheme consists of the seed s together with a ciphertext of **IncSKE**, so its ciphertext overhead is

$$d = O(\log(K_{\text{Ext}} + \lambda) \cdot (\log \ell_{\text{sk}} + \lambda)).$$

Under the two lower-order conditions in the theorem statement, this additive overhead does not change the fixed-parameter ciphertext-rate or compression-rate. $\square$

Combining Corollary 49 with Theorem 51, we obtain the following leakage-resilient incompressible SKE scheme. The base scheme from Corollary 49 has uniform key generation and negligible correctness error, so it satisfies the corresponding hypotheses of Theorem 51.

Corollary 55. *There exists a leakage-resilient incompressible SKE scheme with negligible correctness error, ciphertext-rate $\frac{1}{2}$, compression-rate $\frac{1}{2}$, leakage-rate $1 - o(1)$, and security against unbounded adversaries with respect to the output-length-bounded leakage-class family $\mathcal{F}_{\text{obl}}$ defined above.*

Proof. Let $K_{\text{Ext}} = \ell_{\text{sk}}^{\text{IncSKE}} + c\lambda$ and choose the transformed secret-key length as

$$\ell_{\text{sk}} = K_{\text{Ext}} \log K_{\text{Ext}}.$$

For the base scheme of Corollary 49, $K_{\text{Ext}} = O(\ell_{\text{msg}} + \lambda)$ in the fixed- λ diagonal regime and $K_{\text{Ext}} = O(\ell_{\text{st}})$ in the fixed- $(\lambda, \ell_{\text{msg}})$ large-state regime. Consequently,

$$d = O(\log(K_{\text{Ext}} + \lambda) \cdot (\log \ell_{\text{sk}} + \lambda))$$

is respectively $o(\ell_{\text{msg}})$ and $o(\ell_{\text{st}})$. The rate-preservation conditions of Theorem 51 therefore hold, while $\ell_{\text{sk}} = \omega(K_{\text{Ext}})$ gives leakage rate 1 (and hence $1 - o(1)$). Correctness and security follow directly from Theorem 51. $\square$

6 Leakage-Resilient NC-KEM from Hash Proof Systems

In this section, we show that the Naor-Segev leakage-resilient construction [31] yields a leakage-resilient NC-KEM scheme (see Section 4.2).

Our construction. We require the following primitives.

- Let $\text{HPS} = (\text{HpsGen}, \text{HpsSampl}, \text{HpsSamplY}, \text{HpsKeyGen}, \text{HpsEncap}, \text{HpsDecap})$ be a hash proof system for a language $\mathcal{L} \subseteq X$, and let $\ell = \ell(\lambda, \ell_k)$ be the HPS entropy parameter used by the construction. For the fixed parameters (λ, ℓ) , all encodings have a common length $n_{\text{src}} = n_{\text{src}}(\lambda, \ell)$.
- Fix a constant $c > 0$, define $K_{\text{Ext}} := \ell_k + c\lambda$, and assume $\ell(\lambda, \ell_k) \geq K_{\text{Ext}}$.
- For every public-parameter string pp output by $\text{HpsGen}(1^\lambda, 1^\ell)$, fix an injective encoding $\text{Encode}_{\text{pp}} : \mathcal{K}_{\text{pp}} \rightarrow \{0, 1\}^{n_{\text{src}}}$.

- Let $\text{Ext} : \{0, 1\}^d \times \{0, 1\}^{n_{\text{src}}} \rightarrow \{0, 1\}^{\ell_k}$ be a strong average-case $(K_{\text{Ext}}, \epsilon_{\text{Ext}})$ -extractor, where

$$d = O(\log(K_{\text{Ext}} + \lambda) \cdot (\log n_{\text{src}} + \lambda))$$

and $\epsilon_{\text{Ext}} = \text{negl}(\lambda)$.⁴

Define the admissible leakage bound by

$$\ell_{\text{leak}}(\lambda, \ell_k) := \ell(\lambda, \ell_k) - K_{\text{Ext}} = \ell(\lambda, \ell_k) - (\ell_k + c\lambda).$$

For each parameter pair (λ, ℓ_k) , let

$$\mathcal{F}_{\lambda, \ell_k}^{\text{obl}} := \bigcup_{0 \leq y \leq \ell_{\text{leak}}(\lambda, \ell_k)} \{f : \mathcal{SK}_{\lambda, \ell_k} \rightarrow \{0, 1\}^y \mid f \text{ is PPT}\}.$$

Let $\mathcal{F}_{\text{obl}} = \left\{ \mathcal{F}_{\lambda, \ell_k}^{\text{obl}} \right\}_{\lambda, \ell_k}$ denote the resulting output-length-bounded leakage-class family. Since $\mathcal{F}_{\text{obl}}$ contains all PPT leakage functions whose total output length is bounded by $\ell_{\text{leak}}(\lambda, \ell_k)$, any polynomially many-query pre-challenge leakage phase can be aggregated into a single PPT leakage query that outputs the full leakage transcript. We use this single-query formulation in the hybrid proof, while defining the simulators below on the original online, multiple-query interface.⁵ The leakage-resilient NC-KEM scheme is as follows.

- **Setup**($1^\lambda, 1^{\ell_k}$): The setup algorithm runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$, $(\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$ and $\text{sk} = \text{hps.sk}$, and outputs (pk, sk) .
- **Encaps**(pk): The encapsulation algorithm parses $\text{pk} = (\text{pp}, \text{hps.pk})$, samples a random seed $s \leftarrow \{0, 1\}^d$ and a string-witness pair $(x, w) \leftarrow \text{HpsSampL}(\text{pp})$, computes $R := \text{HpsEncap}(\text{hps.pk}, x, w)$ and $k := \text{Ext}_s(\text{Encode}_{\text{pp}}(R))$, sets $\text{ct} := (x, s)$, and outputs (k, ct) .
- **Decaps**($\text{pk}, \text{sk}, \text{ct}$): Let $\text{pk} = (\text{pp}, \text{hps.pk})$, $\text{sk} = \text{hps.sk}$, and $\text{ct} = (x, s)$. The decapsulation algorithm computes $R := \text{HpsDecap}(\text{hps.sk}, x)$ and $k := \text{Ext}_s(\text{Encode}_{\text{pp}}(R))$, and outputs k .
- **Sim**₁($1^\lambda, 1^{\ell_k}$): The simulator runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}') \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$, samples $s \leftarrow \{0, 1\}^d$, $x \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$, computes $R := \text{HpsDecap}(\text{hps.sk}', x)$, and sets $\text{ct} = (x, s)$. It outputs $(\text{pk}, \text{ct}, \text{state})$, where

$$\text{state} = (\text{pp}, \text{pk}, \text{ct}, R, \perp, (), ()).$$

The last three components are, respectively, a not-yet-sampled working secret key and initially empty lists of leakage functions and answers. The sampled key $\text{hps.sk}'$ is discarded. In particular, **Sim**₁ uses only the efficient HPS algorithms and is PPT; it does not require an efficient uniform sampler for $\mathcal{K}_{\text{pp}}$.

We elaborate on resampling of sk . Define $\mathcal{D}$ as the distribution

$$\mathcal{D} = \left(\text{pp}, \text{pk}, \text{sk}, x, R \left| \begin{array}{l} (\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell) \\ (\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp}), \\ \text{pk} = (\text{pp}, \text{hps.pk}), \text{sk} = \text{hps.sk} \\ x \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}}) \\ R := \text{HpsDecap}(\text{hps.sk}, x) \end{array} \right. \right).$$

Fix an arbitrary fallback element $\text{sk}_\perp(\text{pp})$ of the HPS secret-key space. Let P be the following total, potentially inefficient program. If $(\text{pp}, \text{pk}, x, R)$ has positive probability

⁴The parameters are chosen from Corollary 14.

⁵To obtain the stated leakage budget, we use the lower bound $\log |\mathcal{K}_{\text{pp}}| \geq \ell$ guaranteed by the HPS instance. For the DDH-based HPS from Theorem 29, the estimate $|\text{sk}| = \ell + o(\ell)$ is in the fixed- λ rate regime; the precise bound and the condition for a joint regime are recorded below.

under the corresponding marginal of $\mathcal{D}$, it samples from the conditional distribution of $\mathbf{sk}$ given $(\mathbf{pp}, \mathbf{pk}, x, R)$; otherwise, it outputs $\mathbf{sk}_\perp(\mathbf{pp})$. Then $\tilde{\mathcal{D}} \equiv \mathcal{D}$ when

$$\tilde{\mathcal{D}} = \left(\mathbf{pp}, \mathbf{pk}, \mathbf{sk}, x, R \left| \begin{array}{l} (\mathbf{pp}, \mathbf{pk}, \mathbf{sk}', x, R) \leftarrow \mathcal{D} \\ \mathbf{sk} \leftarrow P(\mathbf{pp}, \mathbf{pk}, x, R) \end{array} \right. \right). \quad (3)$$

By resampling of $\mathbf{sk}$ conditioned on $(\mathbf{pp}, \mathbf{pk}, x, R)$, we mean $\mathbf{sk} \leftarrow P(\mathbf{pp}, \mathbf{pk}, x, R)$. Although the fallback makes P total, it is never used in Eq. (3), because that display samples the conditioning variables from $\mathcal{D}$ itself.

- $\text{Sim}_2(\text{state}, f)$: Parse $\text{state} = (\mathbf{pp}, \mathbf{pk}, \text{ct}, R, \hat{\mathbf{sk}}, \mathbf{f}, \mathbf{r})$ and $\text{ct} = (x, s)$. If $\hat{\mathbf{sk}} = \perp$, sample $\hat{\mathbf{sk}} \leftarrow P(\mathbf{pp}, \mathbf{pk}, x, R)$. Compute $r = f(\hat{\mathbf{sk}})$, append f and r to the respective lists, output r , and set

$$\text{state}' = (\mathbf{pp}, \mathbf{pk}, \text{ct}, R, \hat{\mathbf{sk}}, (\mathbf{f}, f), (\mathbf{r}, r)).$$

Thus every query is answered using the same working secret key.

- $\text{Sim}_3(\text{state}, k)$: Parse $\text{state} = (\mathbf{pp}, \mathbf{pk}, \text{ct}, R, \hat{\mathbf{sk}}, \mathbf{f}, \mathbf{r})$. The simulator samples and outputs $\mathbf{sk}''$ from the conditional distribution determined below by $(\mathbf{pp}, \mathbf{pk}, \text{ct}, \mathbf{f}, \mathbf{r}, k)$. This definition also applies when the two lists are empty.

We elaborate on resampling of $\mathbf{sk}''$. For any finite list of leakage functions $\mathbf{f} = (f_1, \dots, f_q)$, define $\mathcal{D}'_{\mathbf{f}}$ as

$$\mathcal{D}'_{\mathbf{f}} = \left(\mathbf{pp}, \mathbf{pk}, \mathbf{sk}, \mathbf{r}, \text{ct}, k \left| \begin{array}{l} (\mathbf{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell) \\ (\text{hps.pk}, \text{hps.sk}') \leftarrow \text{HpsKeyGen}(\mathbf{pp}), \text{pk} = (\mathbf{pp}, \text{hps.pk}) \\ s \leftarrow \{0, 1\}^d, x \leftarrow \text{HpsSampY}(\mathbf{pp}, \text{td}_{\mathcal{L}}), R \leftarrow \mathcal{K}_{\mathbf{pp}} \\ \mathbf{sk} \leftarrow P(\mathbf{pp}, \mathbf{pk}, x, R), \\ \mathbf{r} := (f_1(\mathbf{sk}), \dots, f_q(\mathbf{sk})), \\ k := \text{Ext}_s(\text{Encode}_{\mathbf{pp}}(R)), \text{ct} = (x, s) \end{array} \right. \right).$$

The uniform raw value in this auxiliary distribution is used only for analysis and conditional resampling, not by Sim_1 . Let $P'_{\mathbf{f}}$ be the following total, potentially inefficient program. If $(\mathbf{pp}, \mathbf{pk}, \text{ct}, \mathbf{r}, k)$ has positive probability under the corresponding marginal of $\mathcal{D}'_{\mathbf{f}}$, it samples from the conditional distribution of $\mathbf{sk}$ given those values; otherwise, it outputs $\mathbf{sk}_\perp(\mathbf{pp})$. Then $\tilde{\mathcal{D}}'_{\mathbf{f}} \equiv \mathcal{D}'_{\mathbf{f}}$ when

$$\tilde{\mathcal{D}}'_{\mathbf{f}} = \left(\mathbf{pp}, \mathbf{pk}, \mathbf{sk}'', \mathbf{r}, \text{ct}, k \left| \begin{array}{l} (\mathbf{pp}, \mathbf{pk}, \mathbf{sk}, \mathbf{r}, \text{ct}, k) \leftarrow \mathcal{D}'_{\mathbf{f}} \\ \mathbf{sk}'' \leftarrow P'_{\mathbf{f}}(\mathbf{pp}, \mathbf{pk}, \text{ct}, \mathbf{r}, k) \end{array} \right. \right). \quad (4)$$

By resampling of $\mathbf{sk}''$ conditioned on $(\mathbf{pp}, \mathbf{pk}, \text{ct}, \mathbf{f}, \mathbf{r}, k)$, we mean $\mathbf{sk}'' \leftarrow P'_{\mathbf{f}}(\mathbf{pp}, \mathbf{pk}, \text{ct}, \mathbf{r}, k)$. Such resamplers exist by exact conditional sampling over finite spaces. The fallback makes both resamplers defined on every input; below we bound the probability that either fallback is reached.

Correctness. For correctness of the generic construction, we use the following pointwise form of HPS correctness: for every $\mathbf{pp}$ output by HpsGen and every $(\text{hps.pk}, \text{hps.sk})$ in the support of $\text{HpsKeyGen}(\mathbf{pp})$, the correctness probability over $(x, w) \leftarrow \text{HpsSampL}(\mathbf{pp})$ and the coins of encapsulation is at least $1 - \text{negl}(\lambda)$, for one negligible function uniform over these choices. A ciphertext in the above scheme is $\text{ct} = (x, s)$ where s is a uniformly random seed and $(x, w) \leftarrow \text{HpsSampL}(\mathbf{pp})$. The encapsulated key is

$$k = \text{Ext}_s(\text{Encode}_{\mathbf{pp}}(\text{HpsEncap}(\text{hps.pk}, x, w))),$$

whereas in the decapsulation algorithm it is

$$k = \text{Ext}_s(\text{Encode}_{\mathbf{pp}}(\text{HpsDecap}(\text{hps.sk}, x))),$$

where $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$, $(\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp})$, $\text{pk} = (\text{pp}, \text{hps.pk})$, and $\text{sk} = \text{hps.sk}$. These two are the same except with negligible probability, by correctness of the HPS.

Rates. The extractor seed length is $d = O(\log(K_{\text{Ext}} + \lambda) \cdot (\log n_{\text{src}} + \lambda))$, so the ciphertext size is $|x| + d$. The relevant asymptotic efficiency notion from Section 4.2 is the fixed- λ leakage rate, with leakage budget $\ell_{\text{leak}}(\lambda, \ell_k) = \ell(\lambda, \ell_k) - K_{\text{Ext}}$. Using the HPS from Theorem 29 instantiated with target HPS entropy parameter ℓ and $\alpha = 3/8$, we obtain

$$|x| = O(\ell^{3/4} b_{\mathcal{G}}) \quad \text{and} \quad |\text{sk}| = \ell + O(\ell^{5/8} + \ell^{3/8} \log p).$$

Thus $|\text{sk}| = \ell + o(\ell)$ in the fixed- λ rate regime. The same conclusion holds in a joint regime provided $\log p = o(\ell^{5/8})$; the stronger condition $b_{\mathcal{G}} = o(\ell^{1/4})$ also makes $|x| = o(\ell)$, where $b_{\mathcal{G}}$ is the group-element encoding length from Theorem 29. Therefore, when $\ell(\lambda, \ell_k) = \omega(K_{\text{Ext}}) = \omega(\ell_k + c\lambda)$ in the fixed- λ rate regime, the leakage rate is

$$\frac{\ell_{\text{leak}}(\lambda, \ell_k)}{|\text{sk}|} = \frac{\ell(\lambda, \ell_k) - (\ell_k + c\lambda)}{|\text{sk}|},$$

which equals $1 - o(1)$. Hence, by the definition in Section 4.2, the resulting NC-KEM family has leakage rate $1 - o(1)$.

Security.

Theorem 56. *Under the setup above, assume that $\text{HPS} = (\text{HpsGen}, \text{HpsSampL}, \text{HpsSampY}, \text{HpsKeyGen}, \text{HpsEncap}, \text{HpsDecap})$ satisfies the pointwise HPS correctness condition stated above, computational language indistinguishability, and statistical smoothness over the raw key space $\mathcal{K}_{\text{pp}}$ (in particular, perfect smoothness suffices). Assume further that for every pp there is an efficient injective encoding $\text{Encode}_{\text{pp}} : \mathcal{K}_{\text{pp}} \rightarrow \{0, 1\}^{n_{\text{src}}}$. Then the above construction is an ℓ_{leak} -LR-NC-KEM with respect to the leakage-class family $\mathcal{F}_{\text{obl}}$. Here Sim_1 is PPT, while Sim_2 and Sim_3 may be inefficient because they use conditional resampling.*

Proof. Our proof proceeds via a sequence of hybrids. Some steps are computational and some are statistical.

Hybrid G_0 : This is the real LR-NC-KEM security game.

- **Initialization Phase:**

1. The adversary $\mathcal{A}$ outputs 1^{ℓ_k} .
2. The challenger runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$ and $\text{sk} = \text{hps.sk}$, and sends pk to $\mathcal{A}$.

- **Pre-challenge Query Phase:** The adversary $\mathcal{A}$ submits a single leakage query $f \in \mathcal{F}_{\lambda, \ell_k}^{\text{obl}}$, and the challenger returns $f(\text{sk})$ to $\mathcal{A}$.

- **Challenge Phase:**

1. The challenger samples a random seed $s^* \leftarrow \{0, 1\}^d$.
2. It samples a string-witness pair $(x^*, w^*) \leftarrow \text{HpsSampL}(\text{pp})$.
3. It computes $R \leftarrow \text{HpsEncap}(\text{hps.pk}, x^*, w^*)$.
4. It computes $k^* := \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))$.
5. It sends $(\text{sk}, k^*, \text{ct} := (x^*, s^*))$ to $\mathcal{A}$.

- **Response Phase:**

1. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

Hybrid G_1 : The challenger uses the decapsulation algorithm instead of the encapsulation algorithm.

- **Initialization Phase:**

1. The adversary $\mathcal{A}$ outputs 1^{ℓ_x} .
2. The challenger runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$ and $\text{sk} = \text{hps.sk}$, and sends pk to $\mathcal{A}$.

- **Pre-challenge Query Phase:** The adversary $\mathcal{A}$ submits a single leakage query $f \in \mathcal{F}_{\lambda, \ell_k}^{\text{obl}}$, and the challenger returns $f(\text{sk})$ to $\mathcal{A}$.

- **Challenge Phase:**

1. The challenger samples a random seed $s^* \leftarrow \{0, 1\}^d$.
2. It samples a string-witness pair $(x^*, w^*) \leftarrow \text{HpsSampL}(\text{pp})$.
3. It computes $R \leftarrow \text{HpsDecap}(\text{hps.sk}, x^*)$.
4. It computes $k^* := \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))$.
5. It sends $(\text{sk}, k^*, \text{ct} := (x^*, s^*))$ to $\mathcal{A}$.

- **Response Phase:**

1. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

Hybrid G_2 : The challenger samples a non-language instance.

- **Initialization Phase:**

1. The adversary $\mathcal{A}$ outputs 1^{ℓ_x} .
2. The challenger runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$ and $\text{sk} = \text{hps.sk}$, and sends pk to $\mathcal{A}$.

- **Pre-challenge Query Phase:** The adversary $\mathcal{A}$ submits a single leakage query $f \in \mathcal{F}_{\lambda, \ell_k}^{\text{obl}}$, and the challenger returns $f(\text{sk})$ to $\mathcal{A}$.

- **Challenge Phase:**

1. The challenger samples a random seed $s^* \leftarrow \{0, 1\}^d$.
2. It samples a string $x^* \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$.
3. It computes $R \leftarrow \text{HpsDecap}(\text{hps.sk}, x^*)$.
4. It computes $k^* := \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))$.
5. It sends $(\text{sk}, k^*, \text{ct} := (x^*, s^*))$ to $\mathcal{A}$.

- **Response Phase:**

1. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

Hybrid G_3 : The challenger generates the ciphertext in the initialization phase, samples a uniformly random raw HPS key R , and resamples the secret key sk .

- **Initialization Phase:**

1. The adversary $\mathcal{A}$ outputs 1^{ℓ_x} .
2. The challenger runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}') \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$, and sends pk to $\mathcal{A}$.

3. The challenger samples a random seed $s^* \leftarrow \{0, 1\}^d$.
 4. It samples a string $x^* \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$.
 5. It samples $R \leftarrow \mathcal{K}_{\text{pp}}$ uniformly.
 6. It computes $k^* := \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))$.
 7. It sets $\text{ct} := (x^*, s^*)$.
 8. It resamples sk conditioned on $(\text{pp}, \text{pk}, x^*, R)$; that is, $\text{sk} \leftarrow P(\text{pp}, \text{pk}, x^*, R)$, where P is as defined in Eq. (3).
- **Pre-challenge Query Phase:** The adversary $\mathcal{A}$ submits a single leakage query $f \in \mathcal{F}_{\lambda, \ell_k}^{\text{obl}}$, and the challenger returns $f(\text{sk})$ to $\mathcal{A}$.
 - **Challenge Phase:**
 1. It sends $(\text{sk}, k^*, \text{ct} := (x^*, s^*))$ to $\mathcal{A}$.
 - **Response Phase:**
 1. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

Hybrid G_4 : The challenger samples k^* uniformly and resamples a new secret key sk'' .

- **Initialization Phase:**
 1. The adversary $\mathcal{A}$ outputs 1^{ℓ_k} .
 2. The challenger runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}') \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$, and sends pk to $\mathcal{A}$.
 3. The challenger samples a random seed $s^* \leftarrow \{0, 1\}^d$.
 4. It samples a string $x^* \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$.
 5. It samples $R \leftarrow \mathcal{K}_{\text{pp}}$ uniformly.
 6. It sets $\text{ct} := (x^*, s^*)$.
 7. It resamples $\text{sk} \leftarrow P(\text{pp}, \text{pk}, x^*, R)$.
- **Pre-challenge Query Phase:** The adversary $\mathcal{A}$ submits a single leakage query $f \in \mathcal{F}_{\lambda, \ell_k}^{\text{obl}}$, and the challenger returns $f(\text{sk})$ to $\mathcal{A}$.
- **Challenge Phase:**
 1. The challenger samples $k^* \leftarrow \{0, 1\}^{\ell_k}$ uniformly.
 2. It resamples sk'' conditioned on $(\text{pp}, \text{pk}, \text{ct}, f(\text{sk}), k^*)$; that is, $\text{sk}'' \leftarrow P'_{(f)}(\text{pp}, \text{pk}, \text{ct}, (f(\text{sk})), k^*)$, where $P'_{(f)}$ is as defined in the description of Sim_3 (see Eq. (4)).
 3. It sends $(\text{sk}'', k^*, \text{ct} := (x^*, s^*))$ to $\mathcal{A}$.
- **Response Phase:**
 1. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

Let $p_{\mathcal{A}, G_i}$ denote the probability that $\mathcal{A}$ outputs 0 in Game G_i . We will show that this probability is almost the same in every game.

Lemma 57. *Assuming pointwise HPS correctness, for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that*

$$|p_{\mathcal{A}, G_1} - p_{\mathcal{A}, G_0}| \leq \text{negl}(\lambda).$$

Proof. The pointwise correctness hypothesis states that for all $\lambda \in \mathbb{N}$ and $\ell \in \mathbb{N}$, every pp output by $\text{HpsGen}(1^\lambda, 1^\ell)$, and every $(\text{hps.pk}, \text{hps.sk})$ in the support of $\text{HpsKeyGen}(\text{pp})$, if $(x, w) \leftarrow \text{HpsSampL}(\text{pp})$, then

$$\Pr[\text{HpsEncap}(\text{hps.pk}, x, w) = \text{HpsDecap}(\text{hps.sk}, x)] \geq 1 - \text{negl}(\lambda).$$

The only difference between G_0 and G_1 is whether the raw HPS key is computed as $\text{HpsEncap}(\text{hps.pk}, x^*, w^*)$ or as $\text{HpsDecap}(\text{hps.sk}, x^*)$. On the event that these two values agree, the extracted key k^* is also identical because both games apply the same seed s^* and the same deterministic map $\text{Ext}_{s^*} \circ \text{Encode}_{\text{pp}}$. Therefore the challenge distributions in G_0 and G_1 differ only with negligible probability. $\square$

Lemma 58. *Assuming computational language indistinguishability of the HPS, for all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that*

$$|p_{\mathcal{A}, G_2} - p_{\mathcal{A}, G_1}| \leq \text{negl}(\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that distinguishes G_1 and G_2 with non-negligible advantage. We build a PPT distinguisher $\mathcal{B}$ for the language-indistinguishability experiment of the HPS.

The distinguisher $\mathcal{B}$ receives (pp, x) , where either $(x, w) \leftarrow \text{HpsSampL}(\text{pp})$ or $x \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$. It samples $(\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$ and $\text{sk} = \text{hps.sk}$, sends pk to $\mathcal{A}$, and answers the displayed leakage query by computing and returning $f(\text{sk})$. This step is efficient because $f \in \mathcal{F}_{\lambda, \ell_x}^{\text{obl}}$ is PPT computable; the fact that pp is public is harmless because it can always be hardwired into the leakage circuit. Next, $\mathcal{B}$ samples $s^* \leftarrow \{0, 1\}^d$, computes

$$R := \text{HpsDecap}(\text{hps.sk}, x) \quad \text{and} \quad k^* := \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R)),$$

sets $\text{ct} = (x, s^*)$, gives $(\text{sk}, k^*, \text{ct})$ to $\mathcal{A}$, and outputs whatever bit $\mathcal{A}$ outputs.

If $(x, w) \leftarrow \text{HpsSampL}(\text{pp})$, then the view of $\mathcal{A}$ is exactly G_1 . If $x \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$, then the view of $\mathcal{A}$ is exactly G_2 . Hence any non-negligible gap between $p_{\mathcal{A}, G_1}$ and $p_{\mathcal{A}, G_2}$ yields a non-negligible distinguisher for the language distributions, contradicting computational language indistinguishability. $\square$

Lemma 59. *Assuming statistical smoothness of the HPS over $\mathcal{K}_{\text{pp}}$, for all adversaries $\mathcal{A}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,*

$$|p_{\mathcal{A}, G_3} - p_{\mathcal{A}, G_2}| \leq \text{negl}(\lambda).$$

Proof. We define an intermediate hybrid $G_{2'}$ to prove the indistinguishability between G_2 and G_3 .

Hybrid $G_{2'}$: The challenger resamples sk conditioned on $(\text{pp}, \text{pk}, x^*, R)$.

• **Initialization Phase:**

1. The adversary $\mathcal{A}$ outputs 1^{ℓ_x} .
2. The challenger runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}') \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$, and sends pk to $\mathcal{A}$.
3. The challenger samples a random seed $s^* \leftarrow \{0, 1\}^d$.
4. It samples a string $x^* \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$.

5. It computes $R \leftarrow \text{HpsDecap}(\text{hps.sk}', x^*)$.
 6. It computes $k^* := \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))$.
 7. It sets $\text{ct} := (x^*, s^*)$.
 8. It resamples $\text{sk} \leftarrow P(\text{pp}, \text{pk}, x^*, R)$.
- **Pre-challenge Query Phase:** The adversary $\mathcal{A}$ submits a single leakage query $f \in \mathcal{F}_{\lambda, \ell_k}^{\text{obl}}$, and the challenger returns $f(\text{sk})$ to $\mathcal{A}$.
 - **Challenge Phase:**
 1. It sends $(\text{sk}, k^*, \text{ct} := (x^*, s^*))$ to $\mathcal{A}$.
 - **Response Phase:**
 1. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

The hybrids G_2 and $G_{2'}$ differ only in the order in which sk , x^* , and R are sampled. In G_2 , the challenger samples $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$, $(\text{hps.pk}, \text{hps.sk}) \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$ and $\text{sk} = \text{hps.sk}$, samples $x^* \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$, and then sets $R = \text{HpsDecap}(\text{hps.sk}, x^*)$. In $G_{2'}$, it first samples the marginal $(\text{pp}, \text{pk}, x^*, R)$ according to the same joint law and then resamples sk from the correct conditional distribution using P . This is exactly the defining property of P in Eq. (3). Hence the joint distribution of $(\text{pp}, \text{pk}, \text{sk}, x^*, R)$ is identical in G_2 and $G_{2'}$, and therefore G_2 and $G_{2'}$ are perfectly indistinguishable.

We now compare $G_{2'}$ and G_3 . Statistical smoothness states that when $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$, $(\text{hps.pk}, \text{hps.sk}') \leftarrow \text{HpsKeyGen}(\text{pp})$, $\text{pk} = (\text{pp}, \text{hps.pk})$, $x^* \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$, the distributions

$$(\text{pp}, \text{pk}, x^*, \text{HpsDecap}(\text{hps.sk}', x^*)) \quad \text{and} \quad (\text{pp}, \text{pk}, x^*, R)$$

with $R \leftarrow \mathcal{K}_{\text{pp}}$ are statistically close. Both $G_{2'}$ and G_3 then apply the same post-processing to $(\text{pp}, \text{pk}, x^*, R)$: they sample the same independent seed s^* , compute $k^* = \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))$, set $\text{ct} = (x^*, s^*)$, resample sk from $P(\text{pp}, \text{pk}, x^*, R)$, answer the leakage query using $f(\text{sk})$, and finally send $(\text{sk}, k^*, \text{ct})$ to the adversary. Therefore, by post-processing, $G_{2'}$ and G_3 are statistically close, which proves the lemma.

This comparison also bounds the unsupported-input branch of P in G_3 . Indeed, the marginal support of $(\text{pp}, \text{pk}, x^*, R)$ under the first distribution has probability one there, whereas the probability assigned to its complement by the second distribution is at most their statistical distance. Thus the fallback branch is reached only with negligible probability (and never for the perfectly smooth DDH instantiation). $\square$

Lemma 60. *Assuming that Ext is a strong average-case extractor, for all adversaries $\mathcal{A}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,*

$$|p_{\mathcal{A}, G_4} - p_{\mathcal{A}, G_3}| \leq \text{negl}(\lambda).$$

Proof. We define an intermediate hybrid $G_{3'}$ to prove the indistinguishability between G_3 and G_4 .

Hybrid $G_{3'}$: The challenger retroactively samples a new secret key to present at the challenge phase.

- **Initialization Phase:**
 1. The adversary $\mathcal{A}$ outputs 1^{ℓ_k} .
 2. The challenger runs $(\text{pp}, \text{td}_{\mathcal{L}}) \leftarrow \text{HpsGen}(1^\lambda, 1^\ell)$ and $(\text{hps.pk}, \text{hps.sk}') \leftarrow \text{HpsKeyGen}(\text{pp})$, sets $\text{pk} = (\text{pp}, \text{hps.pk})$, and sends pk to $\mathcal{A}$.

3. The challenger samples a random seed $s^* \leftarrow \{0, 1\}^d$.
 4. It samples a string $x^* \leftarrow \text{HpsSampY}(\text{pp}, \text{td}_{\mathcal{L}})$.
 5. It samples $R \leftarrow \mathcal{K}_{\text{pp}}$ uniformly.
 6. It sets $\text{ct} := (x^*, s^*)$.
 7. It resamples $\text{sk} \leftarrow P(\text{pp}, \text{pk}, x^*, R)$.
- **Pre-challenge Query Phase:** The adversary $\mathcal{A}$ submits a single leakage query $f \in \mathcal{F}_{\lambda, \ell_k}^{\text{obl}}$, and the challenger returns $f(\text{sk})$ to $\mathcal{A}$.
 - **Challenge Phase:**
 1. The challenger computes $k^* := \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))$.
 2. It resamples $\text{sk}'' \leftarrow P'_{(f)}(\text{pp}, \text{pk}, \text{ct}, (f(\text{sk})), k^*)$.
 3. It sends $(\text{sk}'', k^*, \text{ct} := (x^*, s^*))$ to $\mathcal{A}$.
 - **Response Phase:**
 1. The adversary $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

By definition of $P'_{(f)}$ and Eq. (4), replacing sk by sk'' while keeping $(\text{pp}, \text{pk}, \text{ct}, f(\text{sk}), k^*)$ fixed preserves the joint distribution exactly. Hence G_3 and $G_{3'}$ are perfectly indistinguishable. We now compare $G_{3'}$ and G_4 . Let

$$Z_0 := (\text{pp}, \text{pk}, x^*, \rho, f(\text{sk})),$$

where ρ is the adversary's random tape, sampled independently at the start of the experiment. In $G_{3'}$, conditioned on any fixed value of $(\text{pp}, \text{pk}, x^*, \rho)$, the raw HPS key R is uniformly distributed over $\mathcal{K}_{\text{pp}}$, so

$$H_{\infty}(R \mid \text{pp}, \text{pk}, x^*, \rho) \geq \log |\mathcal{K}_{\text{pp}}| \geq \ell.$$

Moreover, $f(\text{sk})$ has output length at most $\ell_{\text{leak}}(\lambda, \ell_k)$, and $\text{sk} = P(\text{pp}, \text{pk}, x^*, R)$ is a randomized function of $(\text{pp}, \text{pk}, x^*, R)$. We use the standard conditional form of Lemma 9: if L has output length at most t , then $\tilde{H}_{\infty}(X \mid Z, L) \geq \tilde{H}_{\infty}(X \mid Z) - t$. Therefore, by Lemma 9,

$$\tilde{H}_{\infty}(R \mid Z_0) \geq \ell - \ell_{\text{leak}}(\lambda, \ell_k) = K_{\text{Ext}}.$$

The seed s^* is sampled independently of (R, Z_0) ; in particular, it is not included in the extractor's side information. Since $\text{Encode}_{\text{pp}}$ is injective, the encoded source $\text{Encode}_{\text{pp}}(R)$ has the same average conditional min-entropy given Z_0 . Hence, by the strong average-case extractor guarantee,

$$(Z_0, s^*, \text{Ext}_{s^*}(\text{Encode}_{\text{pp}}(R))) \approx_{\text{negl}(\lambda)} (Z_0, s^*, U_{\ell_k}),$$

where $s^* \leftarrow \{0, 1\}^d$ and U_{ℓ_k} is uniform over $\{0, 1\}^{\ell_k}$.

The ciphertext $\text{ct} = (x^*, s^*)$ is a deterministic function of (Z_0, s^*) . Finally, sk'' in both hybrids is sampled from the same total conditional resampling map $P'_{(f)}(\text{pp}, \text{pk}, \text{ct}, (f(\text{sk})), k^*)$. Applying this map and the remaining adversarial computation as post-processing shows that $G_{3'}$ and G_4 are statistically close. Moreover, the event that the fallback branch of $P'_{(f)}$ is used has probability at most the same extractor error in G_4 : the event has probability zero under the left-hand distribution in the preceding display, because its conditioning tuple is sampled from $\mathcal{D}'_{(f)}$, and statistical distance bounds its probability under the right-hand distribution. This proves the lemma. $\square$

In the single-query formulation, G_4 is the simulated world with an idealized initializer that samples R uniformly from $\mathcal{K}_{\text{pp}}$. Although G_4 samples $\text{sk} \leftarrow P(\text{pp}, \text{pk}, x^*, R)$ during initialization, Sim_1 defers precisely this step until the first call to Sim_2 ; no value depending on sk is released between those times. The simulator Sim_2 outputs $f(\text{sk})$, and Sim_3 outputs sk'' using $P'_{(f)}$. If there is no leakage query, Sim_3 instead uses the same definition with the empty function and answer lists, so the interface remains defined.

For polynomially many adaptive queries, fix the coins of the adversary. The complete answer transcript is then a PPT function of the single working secret key, of total output length at most $\ell_{\text{leak}}(\lambda, \ell_k)$. Equivalently, sample the random tape ρ up front and include it in the extractor side information as above. Conditioned on ρ and the public transcript, the ordered query choices are determined by the preceding answers, so the adversary's complete leakage-phase view is reconstructible from ρ and the canonically padded answer transcript. Only the latter, whose length is at most $\ell_{\text{leak}}(\lambda, \ell_k)$, reduces the conditional min-entropy. Consequently the preceding proof applies to this aggregate function. With this idealized initializer, the online Sim_2 implementation has exactly the same distribution: it uses one working key for every answer and records the full ordered lists $(\mathbf{f}, \mathbf{r})$, which Sim_3 uses in P'_f . Thus the proof covers the multiple-query experiment of Definition 42, including the zero-query case.

It remains to compare the idealized initializer with the actual Sim_1 . The latter samples $(\text{pp}, \text{pk}, x, R)$ from the marginal of $\mathcal{D}$. By statistical smoothness, this marginal is within the smoothness error $\delta_{\text{sm}}(\lambda)$ of the same distribution with an independent uniform $R \leftarrow \mathcal{K}_{\text{pp}}$. Adjoin the independent seed and adversary coins and apply the same total randomized continuation: deferred sampling with P , all adaptive leakage answers, the independent uniform challenge key, and the final sampler P'_f . Statistical distance cannot increase under this continuation, even though the resamplers may be inefficient. Thus the actual simulated view is within $\delta_{\text{sm}}(\lambda)$ of G_4 . In the actual simulation, the fallback of P is never used; the fallback of P'_f has probability at most the extractor error plus $\delta_{\text{sm}}(\lambda)$.

Therefore, using the above lemmas and the triangle inequality, for all PPT adversaries $\mathcal{A}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$|p_{\mathcal{A}, G_0} - p_{\mathcal{A}, G_4}| \leq \text{negl}(\lambda).$$

Adding the final $\delta_{\text{sm}}(\lambda)$ comparison gives the required LR-NC-KEM security guarantee for the actual simulators. In total, the smoothness error is incurred twice; for the perfectly smooth DDH instantiation both terms are zero. $\square$

Using the above theorem, we get the following corollary.

Corollary 61. *Assuming the hardness of the DDH problem, for every polynomially bounded ℓ_k and every choice of $\ell = \omega(\ell_k + c\lambda)$, there exists a leakage-resilient NC-KEM with ciphertext size*

$$|\text{ct}| = O(\ell^{3/4} b_G) + O(\log(\ell_k + \lambda) \cdot (\log n_{\text{src}} + \lambda))$$

and leakage rate $1 - o(1)$ in the fixed- λ sense of Section 4.2.

7 Leakage-Resilient Incompressible PKE

In this section, we construct a leakage-resilient incompressible PKE scheme from a leakage-resilient NC-KEM and an incompressible SKE scheme. The security requirement on the base incompressible SKE depends only on the efficiency of the NC-KEM simulators. If the simulators are all PPT, computational incompressibility suffices. Otherwise, even if one simulator may be inefficient, then we need incompressibility against unbounded adversaries.

Our construction. We require the following primitives.

- Let $\text{IncSKE} = (\text{IncSKE.Setup}, \text{IncSKE.Enc}, \text{IncSKE.Dec})$ be an incompressible SKE scheme.

- Let $\text{NCE} = (\text{NCE.Setup}, \text{NCE.Encaps}, \text{NCE.Decaps}, \text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$ be an LR-NC-KEM that is secure with respect to a leakage-class family $\mathcal{F}_{\text{NCE}}$ and leakage-budget function L_{NCE} .

For each parameter triple $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, let $\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ denote the admissible leakage set for the LR-NC-KEM instance with key-length parameter $\ell_{\text{sk}}^{\text{IncSKE}}$, and define

$$\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) := L_{\text{NCE}}(\lambda, \ell_{\text{sk}}^{\text{IncSKE}}).$$

Let $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ denote the induced leakage-class family for the transformed PKE scheme.

We define a PKE scheme $\text{PKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ as follows.

- **Setup** $(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm runs $(\text{pk}, \text{sk}) \leftarrow \text{NCE.Setup}(1^\lambda, 1^{\ell_{\text{sk}}^{\text{IncSKE}}})$ and outputs (pk, sk) .
- **Enc** (pk, m) : For public key pk and message $m \in \mathcal{M}_{\ell_{\text{msg}}}$, the encryption algorithm samples $(k, \text{ct}_0) \leftarrow \text{NCE.Encaps}(\text{pk})$, computes $\text{ct}_1 \leftarrow \text{IncSKE.Enc}(k, m)$, and outputs $\text{ct} := (\text{ct}_0, \text{ct}_1)$.
- **Dec** $(\text{pk}, \text{sk}, \text{ct})$: Given $\text{ct} = (\text{ct}_0, \text{ct}_1)$, the decryption algorithm computes $k \leftarrow \text{NCE.Decaps}(\text{pk}, \text{sk}, \text{ct}_0)$ and outputs $\text{IncSKE.Dec}(k, \text{ct}_1)$.

Correctness. For all $\lambda \in \mathbb{N}$, $\ell_{\text{msg}} \in \mathbb{N}$, $\ell_{\text{st}} \in \mathbb{N}$, and $m \in \mathcal{M}_{\ell_{\text{msg}}}$,

$$\Pr [\text{Dec}(\text{pk}, \text{sk}, \text{Enc}(\text{pk}, m)) = m : (\text{pk}, \text{sk}) \leftarrow \text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})] \geq 1 - \text{negl}(\lambda),$$

where the probability is over the randomness of **Setup** and **Enc**.

Rates. Let $\ell_{\text{ct}}^{\text{NCE}}$ denote the ciphertext length of the LR-NC-KEM instance with key-length parameter $\ell_{\text{sk}}^{\text{IncSKE}}$, and let $\ell_{\text{ct}}^{\text{IncSKE}}$ denote the ciphertext length of **IncSKE**. Then the transformed PKE ciphertext length is

$$\ell_{\text{ct}}^{\text{PKE}} := \ell_{\text{ct}}^{\text{NCE}} + \ell_{\text{ct}}^{\text{IncSKE}}.$$

For fixed λ and in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, the ciphertext rate of the transformed PKE is

$$\text{CiphRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{msg}}}{\ell_{\text{ct}}^{\text{PKE}}},$$

and for fixed λ and ℓ_{msg} , the compression rate is

$$\text{CompRate}(\lambda, \ell_{\text{msg}}) = \lim_{\ell_{\text{st}} \rightarrow \infty} \frac{\ell_{\text{st}}}{\ell_{\text{ct}}^{\text{PKE}}}.$$

In particular, if $\ell_{\text{ct}}^{\text{NCE}} = o(\ell_{\text{msg}})$ in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, then the transformed PKE has the same ciphertext-rate as **IncSKE**. Likewise, if $\ell_{\text{ct}}^{\text{NCE}} = o(\ell_{\text{st}})$ for fixed λ and ℓ_{msg} as $\ell_{\text{st}} \rightarrow \infty$, then the transformed PKE has the same compression-rate as **IncSKE**. Moreover, the admissible leakage functions are exactly those of the underlying LR-NC-KEM instance with key-length parameter $\ell_{\text{sk}}^{\text{IncSKE}}$, and the absolute leakage budget is $\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$. Since the transformed secret key is exactly the LR-NC-KEM secret key, the transformed leakage rate is exactly the leakage rate of the underlying LR-NC-KEM after substituting its key-length parameter by $\ell_{\text{sk}}^{\text{IncSKE}}$.

Theorem 62. Let $\text{NCE} = (\text{NCE.Setup}, \text{NCE.Encaps}, \text{NCE.Decaps}, \text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$ be an LR-NC-KEM that is secure with respect to a leakage-class family $\mathcal{F}_{\text{NCE}}$ and leakage-budget function L_{NCE} . Let $\text{IncSKE} = (\text{IncSKE.Setup}, \text{IncSKE.Enc}, \text{IncSKE.Dec})$ be an incompressible SKE scheme, and let $\mathcal{F}$ and ℓ_{leak} be the induced three-parameter leakage-class family and leakage budget defined above. Assume that $\text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$ outputs a uniform key in $\{0, 1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$.

Then the above construction is an ℓ_{leak} -leakage-resilient incompressible PKE scheme with respect to the induced leakage-class family $\mathcal{F}$ in each of the following two cases.

1. If $\text{Sim}_1, \text{Sim}_2, \text{Sim}_3$ are all PPT, then it suffices that IncSKE be computationally secure as an incompressible SKE scheme.
2. If even one of $\text{Sim}_1, \text{Sim}_2, \text{Sim}_3$ may be inefficient, then it suffices that IncSKE be incompressible secure against unbounded adversaries.

Proof. We use two games. First we show that G_0 and G_1 are indistinguishable by the LR-NC-KEM security of NCE. Then we reduce security of G_1 to incompressibility of IncSKE . We prove the second item. The first item follows from exactly the same reduction, together with the observation that when $\text{Sim}_1, \text{Sim}_2, \text{Sim}_3$ are all PPT, the second reduction remains PPT.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary against the transformed LR-incompressible PKE scheme. We consider the following two games.

Hybrid G_0 . This is the LR-incompressible PKE security game where the challenger randomly chooses $b \in \{0, 1\}$ and encrypts one of m_0, m_1 given by $\mathcal{A}_1$.

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs $(1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
2. The challenger generates $(\text{nce.pk}, \text{nce.sk}) \leftarrow \text{NCE.Setup}(1^\lambda, 1^{\ell_{\text{sk}}^{\text{IncSKE}}})$.
3. It sets public key $\text{pk} = \text{nce.pk}$ and secret key $\text{sk} = \text{nce.sk}$.
4. It sends pk to $\mathcal{A}_1$.

- **Pre-challenge Query Phase:**

1. In this phase, $\mathcal{A}_1$ may make polynomially many leakage queries.
2. For the j^{th} query, it submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, where $f_j : \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_j}$.
3. The challenger returns $f_j(\text{sk})$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.

- **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
2. The challenger samples $b \leftarrow \{0, 1\}$.
3. It generates $(\text{inc.sk}, \text{nce.ct}) \leftarrow \text{NCE.Encaps}(\text{pk})$.
4. It computes $\text{inc.ct} \leftarrow \text{IncSKE.Enc}(\text{inc.sk}, m_b)$.
5. It sends $(\text{nce.ct}, \text{inc.ct})$ to $\mathcal{A}_1$.

- **First Response Phase:**

1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.

- **Second Response Phase:**

1. The second adversary $\mathcal{A}_2$ is given the public key pk , the secret key sk , auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
2. Finally, $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.

Hybrid G_1 . The challenger uses the simulator of the NCE scheme rather than the real algorithms.

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs $(1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.

2. The challenger generates $(\text{pk}, \text{ncc.ct}, \text{state}) \leftarrow \text{Sim}_1(1^\lambda, 1^{\ell_{\text{sk}}^{\text{IncSKE}}})$.
 3. It sends pk to $\mathcal{A}_1$.
- **Pre-challenge Query Phase:**
 1. In this phase, $\mathcal{A}_1$ may make polynomially many leakage queries.
 2. For the j^{th} query, it submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, where $f_j : \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_j}$.
 3. The challenger computes $(r_j, \text{state}') \leftarrow \text{Sim}_2(\text{state}, f_j)$, sets $\text{state} \leftarrow \text{state}'$, and returns r_j , provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.
 - **Challenge Phase:**
 1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
 2. The challenger samples $b \leftarrow \{0, 1\}$.
 3. It generates $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
 4. It computes $\text{inc.ct} \leftarrow \text{IncSKE.Enc}(\text{inc.sk}, m_b)$.
 5. It sends $(\text{ncc.ct}, \text{inc.ct})$ to $\mathcal{A}_1$.
 - **First Response Phase:**
 1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.
 - **Second Response Phase:**
 1. The challenger computes $\text{sk} \leftarrow \text{Sim}_3(\text{state}, \text{inc.sk})$.
 2. The second adversary $\mathcal{A}_2$ is given the public key pk , the secret key sk , auxiliary information aux and state st from the first adversary $\mathcal{A}_1$.
 3. Finally, $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.

Let $p_{\mathcal{A}, G_i}$ denote the probability that $\mathcal{A}$ outputs $b' = b$ in Game G_i .

Lemma 63. *For every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $|p_{\mathcal{A}, G_0} - p_{\mathcal{A}, G_1}| \leq \text{negl}(\lambda)$.*

Proof. We construct a PPT adversary $\mathcal{B}$ against the LR-NC-KEM security experiment. The adversary $\mathcal{B}$ runs $\mathcal{A}$ internally.

On input 1^λ , the adversary $\mathcal{B}$ first runs $\mathcal{A}_1$ until $\mathcal{A}_1$ outputs $(1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$. It then submits $1^{\ell_{\text{sk}}^{\text{IncSKE}}}$ as the LR-NC-KEM key-length parameter to its challenger, receives pk , and forwards pk to $\mathcal{A}_1$. Because $\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ is the admissible leakage set for the LR-NC-KEM instance with key-length parameter $\ell_{\text{sk}}^{\text{IncSKE}}$, every legal leakage query f_j from $\mathcal{A}_1$ is also a legal leakage query for $\mathcal{B}$'s challenger, with the same total leakage bound $\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = L_{\text{NCE}}(\lambda, \ell_{\text{sk}}^{\text{IncSKE}})$. Whenever $\mathcal{A}_1$ makes a leakage query f_j , the adversary $\mathcal{B}$ forwards f_j to its own challenger and returns the answer to $\mathcal{A}_1$.

When $\mathcal{A}_1$ outputs (m_0, m_1, aux) , the adversary $\mathcal{B}$ samples $b \leftarrow \{0, 1\}$. It then receives $(\text{sk}, k, \text{ct}_0)$ from its LR-NC-KEM challenger, computes

$$\text{ct}_1 \leftarrow \text{IncSKE.Enc}(k, m_b),$$

and sends $(\text{ct}_0, \text{ct}_1)$ to $\mathcal{A}_1$. When $\mathcal{A}_1$ returns a state st , the adversary $\mathcal{B}$ gives $(\text{pk}, \text{sk}, \text{aux}, \text{st})$ to $\mathcal{A}_2$, obtains a bit b' , and outputs 0 iff $b' = b$.

If $\mathcal{B}$ is interacting with the real LR-NC-KEM world, then the view of $\mathcal{A}$ is exactly Game G_0 . If $\mathcal{B}$ is interacting with the simulated LR-NC-KEM world, then the view of $\mathcal{A}$ is exactly Game G_1 . Therefore,

$$\Pr[\mathcal{B} \text{ outputs } 0 \text{ in the real world}] = p_{\mathcal{A}, G_0}$$

and

$$\Pr[\mathcal{B} \text{ outputs } 0 \text{ in the simulated world}] = p_{\mathcal{A}, G_1}.$$

Hence the LR-NC-KEM distinguishing advantage of $\mathcal{B}$ is exactly $|p_{\mathcal{A}, G_0} - p_{\mathcal{A}, G_1}|$, which is negligible by security of NCE. $\square$

Lemma 64. *If IncSKE is incompressible secure against unbounded adversaries, then for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, $p_{\mathcal{A}, G_1} \leq \frac{1}{2} + \text{negl}(\lambda)$.*

Proof. Suppose, for contradiction, that some PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ wins Game G_1 with non-negligible advantage. We build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the incompressibility of IncSKE.

- **Initialization Phase:** $\mathcal{B}_1$ runs $\mathcal{A}_1$ on input 1^λ until $\mathcal{A}_1$ outputs $(1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$. The adversary $\mathcal{B}_1$ sends these parameters to its incompressible SKE challenger. It then samples

$$(\text{pk}, \text{ct}_0, \text{state}) \leftarrow \text{Sim}_1(1^\lambda, 1^{\ell_{\text{sk}}^{\text{IncSKE}}})$$

and sends pk to $\mathcal{A}_1$.

- **Pre-challenge Leakage Phase:** For each leakage query f_j made by $\mathcal{A}_1$, the adversary $\mathcal{B}_1$ computes

$$(r_j, \text{state}') \leftarrow \text{Sim}_2(\text{state}, f_j)$$

starting from the state output by Sim_1 , then updates $\text{state} \leftarrow \text{state}'$ and returns r_j to $\mathcal{A}_1$.

- **Challenge Phase:** When $\mathcal{A}_1$ outputs (m_0, m_1, aux) , the adversary $\mathcal{B}_1$ sends

$$(m_0, m_1, \text{aux}') := (m_0, m_1, (\text{pk}, \text{ct}_0, \text{state}, \text{aux}))$$

to its incompressible SKE challenger. The challenger returns a challenge ciphertext ct_1 , and $\mathcal{B}_1$ forwards $(\text{ct}_0, \text{ct}_1)$ to $\mathcal{A}_1$.

- **First Response Phase:** When $\mathcal{A}_1$ outputs a state st with $|\text{st}| \leq \ell_{\text{st}}$, the adversary $\mathcal{B}_1$ forwards st to its challenger.

- **Second Response Phase:** The challenger sends $(k, \text{aux}', \text{st})$ to $\mathcal{B}_2$, where k is the SKE secret key. Parsing $\text{aux}' = (\text{pk}, \text{ct}_0, \text{state}, \text{aux})$, the adversary $\mathcal{B}_2$ computes

$$\text{sk} \leftarrow \text{Sim}_3(\text{state}, k),$$

runs $\mathcal{A}_2$ on $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, and outputs whatever bit $\mathcal{A}_2$ outputs.

This reduction simulates Game G_1 exactly, so

$$\Pr[\text{Expt}_{\mathcal{B}, \text{IncSKE}}^{\text{IncompSKE}}(\lambda) = 1] = p_{\mathcal{A}, G_1}.$$

The only point to note is that the reduction runs the NC-KEM simulators. Hence the reduction is PPT when the simulators are PPT, but it may be inefficient otherwise. Therefore PPT simulators only require computational incompressible SKE security, while inefficient simulators require information-theoretic security against unbounded adversaries.

Since we are proving the second item of the theorem, we may appeal to incompressibility of IncSKE against unbounded adversaries. Thus $\mathcal{B}$ can have at most negligible advantage, contradicting the assumption that $\mathcal{A}$ wins Game G_1 with non-negligible advantage. $\square$

Combining Lemma 63 and Lemma 64, we obtain that every PPT adversary wins the transformed LR-incompressible PKE game with probability at most $1/2 + \text{negl}(\lambda)$. This proves the second item of the theorem. As noted above, when $\text{Sim}_1, \text{Sim}_2, \text{Sim}_3$ are all PPT, the same proof establishes the first item as well. $\square$

Corollary 65. *Assuming the hardness of the DDH problem, there exists a leakage-resilient incompressible PKE scheme with ciphertext-rate $\frac{1}{2}$, compression-rate $\frac{1}{2}$, and leakage-rate $1 - o(1)$.*

Proof. Instantiate Theorem 62 with the incompressible SKE from Corollary 49 and the LR-NC-KEM from Corollary 61. The SKE corollary provides uniform key generation and security against unbounded adversaries, so the hypotheses of the second item of Theorem 62 are satisfied. Choose

$$\ell := (\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda) \log(\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda),$$

which is polynomially bounded and satisfies $\ell = \omega(\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda)$. Then Corollary 61 yields an LR-NC-KEM with ciphertext length

$$O(\ell^{3/4} b_{\mathcal{G}}) + O(\log(\ell_{\text{sk}}^{\text{IncSKE}} + c\lambda) \cdot (\log n_{\text{src}} + \lambda))$$

and leakage-rate $1 - o(1)$. By Corollary 49, we have $\ell_{\text{sk}}^{\text{IncSKE}} = O(\ell_{\text{msg}})$ in the diagonal regime $\ell_{\text{st}} = \ell_{\text{msg}}$, and $\ell_{\text{sk}}^{\text{IncSKE}} = O(\ell_{\text{st}})$ for fixed λ and ℓ_{msg} as $\ell_{\text{st}} \rightarrow \infty$. In the first limit λ is fixed, and in the second both λ and ℓ_{msg} are fixed. Hence $b_{\mathcal{G}}$ and $\log p$ are constant, while n_{src} is polynomially bounded in the displayed parameters. The $3/4$ -power term and the extractor seed are therefore respectively $o(\ell_{\text{msg}})$ and $o(\ell_{\text{st}})$ in the two rate regimes. Hence the LR-NC-KEM ciphertext contributes $o(\ell_{\text{msg}})$ bits on the diagonal and $o(\ell_{\text{st}})$ bits in the large-state regime. By the rates paragraph above, the transformed PKE therefore has ciphertext-rate $\frac{1}{2}$ and compression-rate $\frac{1}{2}$. Its leakage rate is inherited from the LR-NC-KEM and equals $1 - o(1)$. $\square$

8 Leakage-Resilient Incompressible PKE from Leakage-Resilient Incompressible SKE

In this section, we present a transformation from any leakage-resilient incompressible SKE scheme to a leakage-resilient incompressible PKE scheme using the deferred encryption technique. The transformation preserves the absolute leakage budget of the underlying SKE scheme, but the leakage rate decreases because the secret key of the resulting PKE scheme is larger.

Our construction. We require the following primitives.

- Let $\text{IncSKE} = (\text{IncSKE.Setup}, \text{IncSKE.Enc}, \text{IncSKE.Dec})$ be a leakage-resilient incompressible SKE scheme with secret-key length $\ell_{\text{sk}}^{\text{IncSKE}}$ and encryption randomness space $\{0, 1\}^R$.
- Let $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ be a CPA-secure PKE scheme.
- Let $\text{GC} = (\text{GC.Garble}, \text{GC.Eval})$ be a secure garbling scheme.

Let $\mathcal{G} = \{\mathcal{G}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ denote the leakage-class family with respect to which IncSKE is ℓ_{leak} -leakage-resilient incompressible secure. For each parameter triple $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, define $\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ to be the set of all leakage functions f on transformed secret keys such that, for every fixed collection of PKE secret keys

$$T = \{\text{pke.sk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}},$$

the derived function $g_{f,T}$ obtained by hardwiring f and T , and defined by

$$g_{f,T}(x) = f(x, \text{pke.sk}_{i,x_i \in [\ell_{\text{sk}}^{\text{IncSKE}}]}),$$

belongs to $\mathcal{G}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$. Let $\mathcal{F} = \{\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ denote the induced leakage-class family for the transformed PKE scheme.

For each parameter triple $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, let

$$s_{\text{gar}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$$

denote the size of the circuit $C_{m,r}$ that has $m \in \{0,1\}^{\ell_{\text{msg}}}$ and $r \in \{0,1\}^R$ hardwired, takes as input a key $\text{inc.sk} \in \{0,1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$, and outputs $\text{IncSKE.Enc}(\text{inc.sk}, m; r)$. This size depends only on $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, not on the particular values of m and r .

We define a PKE scheme as follows.

- **Setup**($1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$): The setup algorithm generates an incompressible SKE secret key $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$. It also generates $2^{\ell_{\text{sk}}^{\text{IncSKE}}}$ pairs of public and secret keys for PKE: for every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0,1\}$, it samples $(\text{pke.pk}_{i,b}, \text{pke.sk}_{i,b}) \leftarrow \text{PKE.Setup}(1^\lambda)$. The public key is $\text{pk} = \{\text{pke.pk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}$, and the secret key is $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_{i,\text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$.

- **Enc**(pk, m): Let $\text{pk} = \{\text{pke.pk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}$ and $m \in \{0,1\}^{\ell_{\text{msg}}}$. For any m' and r' , let $C_{m',r'}$ be the circuit that has m' and r' hardwired, takes as input a secret key $\text{inc.sk} \in \{0,1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$, and outputs $\text{IncSKE.Enc}(\text{inc.sk}, m'; r')$.

The encryption algorithm samples $r \leftarrow \{0,1\}^R$ and constructs the circuit $C_{m,r}$. It then computes $(\hat{C}, \{\text{lab}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}) \leftarrow \text{GC.Garble}(1^\lambda, C_{m,r})$. For every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0,1\}$, it computes $\text{pke.ct}_{i,b} \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,b}, \text{lab}_{i,b})$. It outputs $\text{ct} = (\hat{C}, \{\text{pke.ct}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}})$.

- **Dec**($\text{pk}, \text{sk}, \text{ct}$): Let $\text{pk} = \{\text{pke.pk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}$, $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_i\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$, where the secret key components are $\text{pke.sk}_i = \text{pke.sk}_{i,\text{inc.sk}[i]}$, and $\text{ct} = (\hat{C}, \{\text{pke.ct}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}})$. For every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$, the decryption algorithm computes

$$\text{lab}_{i,\text{inc.sk}[i]} \leftarrow \text{PKE.Dec}(\text{pke.pk}_{i,\text{inc.sk}[i]}, \text{pke.sk}_i, \text{pke.ct}_{i,\text{inc.sk}[i]}).$$

It then evaluates the garbled circuit and obtains $\text{inc.ct} \leftarrow \text{GC.Eval}(\hat{C}, \{\text{lab}_{i,\text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$. Finally, it outputs $m \leftarrow \text{IncSKE.Dec}(\text{inc.sk}, \text{inc.ct})$.

Correctness. Let $m \in \{0,1\}^{\ell_{\text{msg}}}$ and let $\text{ct} = (\hat{C}, \{\text{pke.ct}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}})$ be an encryption of m . By construction, $(\hat{C}, \{\text{lab}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}) \leftarrow \text{GC.Garble}(1^\lambda, C_{m,r})$ for some $r \in \{0,1\}^R$, and $\text{pke.ct}_{i,b} \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,b}, \text{lab}_{i,b})$ for every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0,1\}$.

The secret key is $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_{i,\text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$. By correctness of PKE, for every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$, decryption of $\text{pke.ct}_{i,\text{inc.sk}[i]}$ using the pair $(\text{pke.pk}_{i,\text{inc.sk}[i]}, \text{pke.sk}_{i,\text{inc.sk}[i]})$ recovers the label $\text{lab}_{i,\text{inc.sk}[i]}$. Thus, the decryption algorithm obtains the labels corresponding to the input inc.sk of the garbled circuit. By correctness of the garbling scheme, $\text{GC.Eval}(\hat{C}, \{\text{lab}_{i,\text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]}) = C_{m,r}(\text{inc.sk}) = \text{IncSKE.Enc}(\text{inc.sk}, m; r)$. Finally, correctness of IncSKE implies that $\text{IncSKE.Dec}(\text{inc.sk}, \text{IncSKE.Enc}(\text{inc.sk}, m; r)) = m$.

Rates. The ciphertext of the above scheme has the form $\text{ct} = (\hat{C}, \{\text{pke.ct}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}})$. Hence its size is

$$\ell_{\text{ct}}^{\text{PKE}} = |\hat{C}| + 2^{\ell_{\text{sk}}^{\text{IncSKE}}} \cdot |\text{pke.ct}|.$$

Here $\hat{C}$ is a garbling of the circuit $C_{m,r}$, whose size may be $\text{poly}(\lambda, \ell_{\text{st}}, \ell_{\text{msg}})$ even when the underlying incompressible SKE scheme has ciphertext-rate $1 - o(1)$ and compression-rate $1 - o(1)$. Accordingly, this transformation is not intended to preserve high ciphertext-rate or high compression-rate.

The construction preserves the same absolute leakage budget function $\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$ as the underlying incompressible SKE scheme. The transformed secret key is $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_{i,\text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$, so its length is $\ell_{\text{sk}}^{\text{IncSKE}}(1 + |\text{pke.sk}|)$. Hence, by the leakage-rate definition in Section 4.1, the transformed PKE leakage rate is the underlying SKE leakage rate divided by $1 + |\text{pke.sk}|$. Thus, the absolute leakage budget is preserved, while the leakage rate decreases by the constant factor $1/(1 + |\text{pke.sk}|)$.

Theorem 66. Assume $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ is a CPA-secure PKE scheme, $\text{IncSKE} = (\text{IncSKE.Setup}, \text{IncSKE.Enc}, \text{IncSKE.Dec})$ is an incompressible SKE scheme that is ℓ_{leak} -leakage-resilient with respect to $\mathcal{G}$ in the sense of Section 4.1, and $\text{GC} = (\text{GC.Garble}, \text{GC.Eval})$ is a secure garbling scheme. Then the above construction is an ℓ_{leak} -leakage-resilient incompressible PKE scheme with respect to the induced leakage class $\mathcal{F}$. In particular, by the leakage-rate definition in Section 4.1, the transformed PKE leakage rate is the underlying SKE leakage rate divided by $1 + |\text{pke.sk}|$.

Proof. Our proof proceeds via a sequence of hybrids. First, we will describe all hybrids, and then show that the hybrids are computationally indistinguishable.

As in the incompressible-PKE definition, the second-stage adversary receives $(\text{pk}, \text{sk}, \text{aux}, \text{st})$.

Hybrid G_0 : This is the real LR-incompressible PKE security game for the above construction.

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
2. The challenger runs the key generation $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
3. It computes $(\text{pke.pk}_{i,b}, \text{pke.sk}_{i,b}) \leftarrow \text{PKE.Setup}(1^\lambda)$ for all $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0, 1\}$.
4. It sets $\text{pk} = \{\text{pke.pk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}}$ and $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_{i, \text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$, and sends pk to $\mathcal{A}_1$.

- **Pre-challenge Query Phase:**

1. In this phase, $\mathcal{A}_1$ may make polynomially many leakage queries.
2. For the j^{th} query, it submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ with output length ℓ_j .
3. The challenger returns $f_j(\text{sk})$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.

- **Challenge Phase:**

1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) , where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
2. The challenger randomly samples $r^* \in \{0, 1\}^R$ and $d \in \{0, 1\}$, and creates the circuit C_{m_d, r^*} .
3. It computes $(\hat{C}^*, \{\text{lab}_{i,b}^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}}) \leftarrow \text{GC.Garble}(1^\lambda, C_{m_d, r^*})$.
4. For all $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0, 1\}$, it computes $\text{pke.ct}_{i,b}^* \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,b}, \text{lab}_{i,b}^*)$.
5. It sends $\text{ct}^* = (\hat{C}^*, \{\text{pke.ct}_{i,b}^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}})$ to $\mathcal{A}_1$.

- **First Response Phase:**

1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.

- **Second Response Phase:**

1. The second adversary $\mathcal{A}_2$ is given the public key pk , the secret key sk , auxiliary information aux , and state st from the first adversary $\mathcal{A}_1$.
2. Finally, $\mathcal{A}_2$ outputs $d' \in \{0, 1\}$.

Hybrid G_1 : This game is identical to G_0 , except that in the challenge phase, for every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$, both ciphertexts $\text{pke.ct}_{i,0}^*$ and $\text{pke.ct}_{i,1}^*$ encrypt the active label $\text{lab}_{i, \text{inc.sk}[i]}^*$.

- **Initialization Phase:**

1. The first adversary $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.

2. The challenger runs the key generation $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
 3. It computes $(\text{pke.pk}_{i,b}, \text{pke.sk}_{i,b}) \leftarrow \text{PKE.Setup}(1^\lambda)$ for all $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0, 1\}$.
 4. It sets $\text{pk} = \{\text{pke.pk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}}$ and $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_{i, \text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$, and sends pk to $\mathcal{A}_1$.
- **Pre-challenge Query Phase:**
 1. In this phase, $\mathcal{A}_1$ may make polynomially many leakage queries.
 2. For the j^{th} query, it submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ with output length ℓ_j .
 3. The challenger returns $f_j(\text{sk})$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.
 - **Challenge Phase:**
 1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) , where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.
 2. The challenger randomly samples $r^* \in \{0, 1\}^R$ and $d \in \{0, 1\}$, and creates the circuit C_{m_d, r^*} .
 3. It computes $(\hat{C}^*, \{\text{lab}_{i,b}^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}}) \leftarrow \text{GC.Garble}(1^\lambda, C_{m_d, r^*})$.
 4. For all $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$, it computes $\text{pke.ct}_{i, \text{inc.sk}[i]}^* \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i, \text{inc.sk}[i]}, \text{lab}_{i, \text{inc.sk}[i]}^*)$.
 5. For all $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$, it computes $\text{pke.ct}_{i, 1 - \text{inc.sk}[i]}^* \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i, 1 - \text{inc.sk}[i]}, \text{lab}_{i, \text{inc.sk}[i]}^*)$.
 6. It sends $\text{ct}^* = (\hat{C}^*, \{\text{pke.ct}_{i,b}^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}})$ to $\mathcal{A}_1$.
 - **First Response Phase:**
 1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.
 - **Second Response Phase:**
 1. The second adversary $\mathcal{A}_2$ is given the public key pk , the secret key sk , auxiliary information aux , and state st from the first adversary $\mathcal{A}_1$.
 2. Finally, $\mathcal{A}_2$ outputs $d' \in \{0, 1\}$.

Hybrid G_2 : This game is identical to G_1 , except that in the challenge phase, the garbled circuit and the active input labels are simulated.

- **Initialization Phase:**
 1. The first adversary $\mathcal{A}_1$ outputs $1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}}$.
 2. The challenger runs the key generation $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$.
 3. It computes $(\text{pke.pk}_{i,b}, \text{pke.sk}_{i,b}) \leftarrow \text{PKE.Setup}(1^\lambda)$ for all $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0, 1\}$.
 4. It sets $\text{pk} = \{\text{pke.pk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}}$ and $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_{i, \text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$, and sends pk to $\mathcal{A}_1$.
- **Pre-challenge Query Phase:**
 1. In this phase, $\mathcal{A}_1$ may make polynomially many leakage queries.
 2. For the j^{th} query, it submits a function $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ with output length ℓ_j .
 3. The challenger returns $f_j(\text{sk})$, provided the total leakage satisfies $\sum_j \ell_j \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$.
- **Challenge Phase:**
 1. The first adversary $\mathcal{A}_1$ sends (m_0, m_1, aux) , where aux is auxiliary information which will be relayed to the second adversary $\mathcal{A}_2$.

2. The challenger randomly samples $r^* \in \{0, 1\}^R$ and $d \in \{0, 1\}$, and creates the circuit C_{m_d, r^*} .
3. It computes $\text{inc.ct}^* \leftarrow \text{IncSKE.Enc}(\text{inc.sk}, m_d; r^*)$.
4. It computes $(\hat{C}^*, \{\text{lab}_i^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]}) \leftarrow \text{Sim}(1^\lambda, 1^{\ell_{\text{sk}}^{\text{IncSKE}}}, 1^{s_{\text{gar}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}, \text{inc.ct}^*)$.
5. For all $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0, 1\}$, it computes $\text{pke.ct}_{i,b}^* \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,b}, \text{lab}_i^*)$.
6. It sends $\text{ct}^* = (\hat{C}^*, \{\text{pke.ct}_{i,b}^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0, 1\}})$ to $\mathcal{A}_1$.

- **First Response Phase:**

1. The first adversary $\mathcal{A}_1$ outputs a state st such that $|\text{st}| \leq \ell_{\text{st}}$.

- **Second Response Phase:**

1. The second adversary $\mathcal{A}_2$ is given the public key pk , the secret key sk , auxiliary information aux , and state st from the first adversary $\mathcal{A}_1$.
2. Finally, $\mathcal{A}_2$ outputs $d' \in \{0, 1\}$.

Let $p_{\mathcal{A}, G_i}$ denote the probability that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ outputs $d' = d$ in game G_i .

Lemma 67. *Assuming PKE is CPA-secure, for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that $|p_{\mathcal{A}, G_1} - p_{\mathcal{A}, G_0}| \leq \text{negl}(\lambda)$.*

Proof. The only difference between G_0 and G_1 is that, for each $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$, the ciphertext $\text{pke.ct}_{i,1-\text{inc.sk}[i]}^*$ encrypts $\text{lab}_{i,\text{inc.sk}[i]}^*$ in G_1 instead of $\text{lab}_{i,1-\text{inc.sk}[i]}^*$ in G_0 . The secret key $\text{pke.sk}_{i,1-\text{inc.sk}[i]}$ is not included in the transformed secret key and is not used to answer leakage queries.

Consider intermediate hybrids $H_0, \dots, H_{\ell_{\text{sk}}^{\text{IncSKE}}}$, where in H_j , for every $i \leq j$, the ciphertext $\text{pke.ct}_{i,1-\text{inc.sk}[i]}^*$ encrypts $\text{lab}_{i,\text{inc.sk}[i]}^*$, and for every $i > j$, it encrypts $\text{lab}_{i,1-\text{inc.sk}[i]}^*$. Thus $H_0 = G_0$ and $H_{\ell_{\text{sk}}^{\text{IncSKE}}} = G_1$.

Suppose there exists $j \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ such that $\mathcal{A}$ distinguishes H_{j-1} and H_j with non-negligible advantage. We construct a CPA adversary $\mathcal{B}$ against PKE. The adversary $\mathcal{B}$ samples $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$ and generates all PKE key pairs honestly, except that it uses the CPA challenge public key as $\text{pke.pk}_{j,1-\text{inc.sk}[j]}$. It can still form the transformed secret key because this key contains only $\text{pke.sk}_{i,\text{inc.sk}[i]}$ for each $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$. Thus $\mathcal{B}$ can answer all pre-challenge leakage queries exactly as in the hybrid game.

In the challenge phase, $\mathcal{B}$ generates the garbled circuit and all labels honestly. It sends the pair $(\text{lab}_{j,1-\text{inc.sk}[j]}^*, \text{lab}_{j,\text{inc.sk}[j]}^*)$ to its CPA challenger and uses the returned challenge ciphertext as $\text{pke.ct}_{j,1-\text{inc.sk}[j]}^*$. All other ciphertexts are generated according to the hybrid being simulated. If the CPA challenge ciphertext encrypts $\text{lab}_{j,1-\text{inc.sk}[j]}^*$, then $\mathcal{B}$ simulates H_{j-1} ; if it encrypts $\text{lab}_{j,\text{inc.sk}[j]}^*$, then $\mathcal{B}$ simulates H_j . Hence any non-negligible distinguishing advantage between adjacent hybrids yields a non-negligible CPA advantage. Since $\ell_{\text{sk}}^{\text{IncSKE}}$ is polynomial, the claim follows by a hybrid argument. $\square$

Lemma 68. *Assuming GC is secure, for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that $|p_{\mathcal{A}, G_2} - p_{\mathcal{A}, G_1}| \leq \text{negl}(\lambda)$.*

Proof. This follows from the simulation security of the garbling scheme. In G_1 , the challenger gives the adversary a real garbling of C_{m_d, r^*} together with encryptions of the active input labels corresponding to inc.sk . In G_2 , the challenger first computes the output $\text{inc.ct}^* = \text{IncSKE.Enc}(\text{inc.sk}, m_d; r^*)$ and then uses the garbling simulator to produce a simulated garbled circuit and simulated active labels.

If an adversary distinguishes G_1 and G_2 with non-negligible advantage, then we build a garbling adversary that uses this adversary as a subroutine. The reduction simulates the public key, secret key, leakage oracle, and all parts of the experiment up to the challenge phase. At

the challenge phase, it submits the circuit C_{m_d, r^*} and input inc.sk to the garbling security experiment. If the garbling challenger returns a real garbled circuit and real active labels, the reduction simulates G_1 ; if it returns a simulated garbled circuit and simulated active labels, the reduction simulates G_2 . The PKE ciphertexts are then formed by encrypting the returned active label for coordinate i under both $\text{pke.pk}_{i,0}$ and $\text{pke.pk}_{i,1}$. Therefore, any non-negligible difference between G_1 and G_2 contradicts the security of GC. $\square$

Lemma 69. *Assuming IncSKE is an ℓ_{leak} -leakage-resilient incompressible SKE scheme, for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function negl such that $p_{\mathcal{A}, G_2} \leq 1/2 + \text{negl}(\lambda)$.*

Proof. Suppose there exists an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that has non-negligible advantage in G_2 . We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for the leakage-resilient incompressible SKE security game.

The adversary $\mathcal{B}_1$ receives the parameters ℓ_{msg} and ℓ_{st} from $\mathcal{A}_1$ and forwards them to its SKE challenger. The challenger samples $\text{inc.sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$. The adversary $\mathcal{B}_1$ independently samples $(\text{pke.pk}_{i,b}, \text{pke.sk}_{i,b}) \leftarrow \text{PKE.Setup}(1^\lambda)$ for every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0, 1\}$, sets $\text{pk} = \{\text{pke.pk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}$, and sends pk to $\mathcal{A}_1$.

For each pre-challenge leakage query $f_j \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ made by $\mathcal{A}_1$, the adversary $\mathcal{B}_1$ defines a derived leakage function $g_{f_j, T}$, where $T = \{\text{pke.sk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}$ is the collection of PKE secret keys sampled by $\mathcal{B}_1$. The function $g_{f_j, T}$ has f_j and T hardwired, takes as input only an incompressible SKE secret key $x \in \{0, 1\}^{\ell_{\text{sk}}^{\text{IncSKE}}}$, and is defined by

$$g_{f_j, T}(x) = f_j(x, \text{pke.sk}_{i, x_{i \in [\ell_{\text{sk}}^{\text{IncSKE}]}}}).$$

By the definition of the induced leakage class, the hardwired function $g_{f_j, T}$ belongs to the set $\mathcal{G}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ of admissible leakage functions for the underlying incompressible SKE scheme. The adversary $\mathcal{B}_1$ sends $g_{f_j, T}$ to its SKE leakage oracle and receives

$$g_{f_j, T}(\text{inc.sk}) = f_j(\text{inc.sk}, \{\text{pke.sk}_{i, \text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]}),$$

which it returns to $\mathcal{A}_1$. Since each $g_{f_j, T}$ has the same output length as f_j , and since the transformation preserves the same leakage budget function $\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, any legal sequence of leakage queries for the transformed PKE scheme induces a legal sequence of leakage queries for the underlying SKE scheme.

When $\mathcal{A}_1$ outputs (m_0, m_1, aux) , the adversary $\mathcal{B}_1$ sends (m_0, m_1, aux') to its SKE challenger, where $\text{aux}' = (\text{pk}, \text{aux}, \{\text{pke.sk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}})$. The SKE challenger returns a challenge ciphertext inc.ct^* , which is an encryption of m_d under inc.sk for a random bit d . The adversary $\mathcal{B}_1$ computes

$$(\hat{C}^*, \{\text{lab}_i^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]}) \leftarrow \text{Sim}(1^\lambda, 1^{\ell_{\text{sk}}^{\text{IncSKE}}}, 1^{s_{\text{gar}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})}, \text{inc.ct}^*),$$

using the simulator from the garbling scheme. It then computes $\text{pke.ct}_{i,b}^* \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,b}, \text{lab}_i^*)$ for every $i \in [\ell_{\text{sk}}^{\text{IncSKE}}]$ and $b \in \{0, 1\}$, and sends $\text{ct}^* = (\hat{C}^*, \{\text{pke.ct}_{i,b}^*\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}})$ to $\mathcal{A}_1$.

The adversary $\mathcal{A}_1$ outputs a state st with $|\text{st}| \leq \ell_{\text{st}}$, and $\mathcal{B}_1$ forwards st to its SKE challenger. In the second phase, the SKE challenger gives inc.sk , aux' , and st to $\mathcal{B}_2$. The adversary $\mathcal{B}_2$ parses aux' to recover pk , aux , and $\{\text{pke.sk}_{i,b}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}], b \in \{0,1\}}$, sets $\text{sk} = (\text{inc.sk}, \{\text{pke.sk}_{i, \text{inc.sk}[i]}\}_{i \in [\ell_{\text{sk}}^{\text{IncSKE}}]})$, and gives $(\text{pk}, \text{sk}, \text{aux}, \text{st})$ to $\mathcal{A}_2$. Finally, $\mathcal{A}_2$ outputs d' , and $\mathcal{B}_2$ outputs the same bit.

The view of $\mathcal{A}$ in this reduction is distributed identically to its view in G_2 . The public key is generated honestly, leakage queries are answered exactly as in G_2 , the challenge ciphertext is formed using the simulated garbling of the SKE challenge ciphertext, and the second-stage adversary receives the correctly distributed transformed secret key. The only point to note is that the simulator input length $s_{\text{gar}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$ is valid because it equals $|C_{m,r}|$ for every choice of $m \in \{0, 1\}^{\ell_{\text{msg}}}$ and $r \in \{0, 1\}^R$. Therefore $\mathcal{B}$ wins the leakage-resilient incompressible SKE game with probability $p_{\mathcal{A}, G_2}$. By the security of IncSKE, this probability is at most $1/2 + \text{negl}(\lambda)$. $\square$

By Lemma 67, Lemma 68, Lemma 69, and the triangle inequality, for every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $p_{\mathcal{A}, G_0} \leq 1/2 + \text{negl}(\lambda)$. This proves the theorem. $\square$

9 Bit-Leakage-Resilient Incompressible PKE from any Incompressible PKE

In this section, we give a transformation from any incompressible PKE scheme to a leakage-resilient incompressible PKE scheme for the class of leakage functions that reveal subsets of the secret-key bits. The same transformation also applies in the secret-key setting: starting from an incompressible SKE scheme, one obtains a bit-leakage-resilient incompressible SKE scheme.

Throughout this section, bit leakage is measured with respect to the encoded secret-key length. Thus, an adversary may adaptively request subsets of encoded secret-key coordinates, but the total number of distinct leaked coordinates must be at most an ℓ fraction of the encoded secret-key length.

Our construction. We require the following primitives.

- $\text{IncPKE} = (\text{IncPKE.Setup}, \text{IncPKE.Enc}, \text{IncPKE.Dec})$ be an incompressible public-key encryption scheme for arbitrary message length ℓ_{msg} , with ciphertext length $\ell_{\text{ct}}^{\text{IncPKE}}$ and secret-key length $\ell_{\text{sk}}^{\text{IncPKE}}$. We assume that the secret key inc.sk of IncPKE is represented as a binary string in $\{0, 1\}^{\ell_{\text{sk}}^{\text{IncPKE}}}$.
- $\text{RPE} = (\text{RPE.Enc}, \text{RPE.Dec}, \text{RPE.Recon})$ be a $(\ell_{\text{sk}}^{\text{IncPKE}}, \ell_{\text{sk}}, 0, \ell)$ binary reconstructable probabilistic encoding scheme.

For the transformed scheme on parameters $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, define

$$\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) := \lfloor \ell \cdot \ell_{\text{sk}} \rfloor,$$

where ℓ_{sk} denotes the encoded secret-key length for those parameters. For every subset $Q \subseteq [\ell_{\text{sk}}]$, define the projection function

$$\pi_Q : \{0, 1\}^{\ell_{\text{sk}}} \rightarrow \{0, 1\}^{|Q|} \quad \text{by} \quad \pi_Q(w) := w_Q.$$

For each parameter triple $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, let

$$\mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}^{\text{bit}} := \{\pi_Q : Q \subseteq [\ell_{\text{sk}}]\}.$$

Let $\mathcal{F}_{\text{bit}} = \left\{ \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}^{\text{bit}} \right\}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ denote the resulting leakage-class family. Throughout this section, the pre-challenge leakage phase is specialized as follows: the adversary may adaptively query functions $\pi_{Q^{(j)}} \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}^{\text{bit}}$, and the challenger answers the j^{th} query iff

$$\left| \bigcup_{t \leq j} Q^{(t)} \right| \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}).$$

This is the bit-leakage specialization of the general LR-incompressible PKE experiment from Section 4.1.

We define a new leakage-resilient incompressible PKE scheme $\text{PKE} = (\text{Setup}, \text{Enc}, \text{Dec})$ as follows.

- **Setup** $(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: The setup algorithm runs $(\text{inc.pk}, \text{inc.sk}) \leftarrow \text{IncPKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$. It then computes $\text{sk} \leftarrow \text{RPE.Enc}(\text{inc.sk})$ and outputs public key $\text{pk} = \text{inc.pk}$ and secret key sk .

- $\text{Enc}(\text{pk}, m)$: On input public key $\text{pk} = \text{inc.pk}$ and message $m \in \{0, 1\}^{\ell_{\text{msg}}}$, the encryption algorithm computes $\text{ct} \leftarrow \text{IncPKE}.\text{Enc}(\text{inc.pk}, m)$ and outputs ct .
- $\text{Dec}(\text{pk}, \text{sk}, \text{ct})$: On input public key $\text{pk} = \text{inc.pk}$, secret key sk , and ciphertext ct , the decryption algorithm first computes $\text{inc.sk} \leftarrow \text{RPE}.\text{Dec}(\text{sk})$. It then outputs $m := \text{IncPKE}.\text{Dec}(\text{inc.pk}, \text{inc.sk}, \text{ct})$.

Correctness. Let $m \in \{0, 1\}^{\ell_{\text{msg}}}$ and let $\text{ct} \leftarrow \text{Enc}(\text{pk}, m)$. By construction, $\text{pk} = \text{inc.pk}$ and $\text{ct} \leftarrow \text{IncPKE}.\text{Enc}(\text{inc.pk}, m)$. The secret key of the new scheme is $\text{sk} \leftarrow \text{RPE}.\text{Enc}(\text{inc.sk})$. Since RPE has decoding error parameter 0, we have $\text{RPE}.\text{Dec}(\text{sk}) = \text{inc.sk}$. Therefore, decryption in the new scheme returns $\text{IncPKE}.\text{Dec}(\text{inc.pk}, \text{inc.sk}, \text{ct})$, which equals m by the correctness of IncPKE.

Rates. The ciphertexts of the transformed scheme are exactly the ciphertexts of IncPKE. Hence the ciphertext-rate and compression-rate are inherited unchanged from IncPKE. The transformed secret-key length is ℓ_{sk} , the output length of the RPE, and the absolute leakage budget is $\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = \lfloor \ell \cdot \ell_{\text{sk}} \rfloor$. Therefore, by the leakage-rate definition in Section 4.1, the transformed scheme has leakage-rate $\ell - o(1)$; in particular, if ℓ is constant, then the leakage-rate equals ℓ .

Theorem 70. Let $\text{IncPKE} = (\text{IncPKE}.\text{Setup}, \text{IncPKE}.\text{Enc}, \text{IncPKE}.\text{Dec})$ be an incompressible public-key encryption scheme for arbitrary message length ℓ_{msg} , with ciphertext length $\ell_{\text{ct}}^{\text{IncPKE}}$ and secret-key length $\ell_{\text{sk}}^{\text{IncPKE}}$. Let $\text{RPE} = (\text{RPE}.\text{Enc}, \text{RPE}.\text{Dec}, \text{RPE}.\text{Recon})$ be a $(\ell_{\text{sk}}^{\text{IncPKE}}, \ell_{\text{sk}}, 0, \ell)$ binary reconstructable probabilistic encoding scheme.

Then the above construction is an ℓ_{leak} -leakage-resilient incompressible public-key encryption scheme with respect to the leakage-class family $\mathcal{F}_{\text{bit}}$ and the bit-leakage experiment specified above.

Proof. We show that any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that wins the bit-leakage-resilient incompressibility game for the transformed scheme can be used to build an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that wins the incompressibility game for IncPKE with exactly the same success probability. The adversary $\mathcal{B}$ internally runs $\mathcal{A}$ and simulates its leakage oracle.

Initialization. The adversary $\mathcal{B}_1$ runs $\mathcal{A}_1$ on input 1^λ . Whenever $\mathcal{A}_1$ specifies the parameters ℓ_{msg} and ℓ_{st} , the adversary $\mathcal{B}_1$ forwards these parameters to the incompressibility challenger for IncPKE. The challenger runs $(\text{inc.pk}, \text{inc.sk}) \leftarrow \text{IncPKE}.\text{Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$ and sends inc.pk to $\mathcal{B}_1$. The adversary $\mathcal{B}_1$ gives $\text{pk} := \text{inc.pk}$ to $\mathcal{A}_1$.

Pre-challenge Query phase. The adversary $\mathcal{B}_1$ initializes an empty table $T = \emptyset$. The adversary $\mathcal{A}_1$ may adaptively make leakage queries $\pi_{Q(j)} \in \mathcal{F}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}^{\text{bit}}$. Let $S_j := \bigcup_{t \leq j} Q^{(t)}$ be the set of distinct encoded secret-key coordinates requested up to the j -th leakage query. By the definition of the bit-leakage experiment in this section, we have $|S_j| \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$ for every j .

To answer a query $\pi_{Q(j)}$, the adversary $\mathcal{B}_1$ proceeds as follows. For every coordinate $i \in Q^{(j)}$, if T already contains an entry (i, r_i) , it reuses this value. Otherwise, it samples a fresh uniform bit $r_i \leftarrow \{0, 1\}$ and stores (i, r_i) in T . It returns the indexed tuple $(r_i)_{i \in Q^{(j)}}$ to $\mathcal{A}_1$.

Let $S := \{i : (i, r_i) \in T\}$ be the final set of leaked coordinates. Let $\tilde{w}_S \in \{0, 1\}^S$ denote the corresponding partial assignment, defined by $(\tilde{w}_S)_i = r_i$ for every $i \in S$.

Challenge phase. The adversary $\mathcal{A}_1$ outputs two messages $m_0, m_1 \in \{0, 1\}^{\ell_{\text{msg}}}$ together with auxiliary information aux . The adversary $\mathcal{B}_1$ sends (m_0, m_1, aux') to its challenger, where $\text{aux}' := (\text{aux}, T)$. The challenger samples $b \leftarrow \{0, 1\}$, computes $\text{ct}^* \leftarrow \text{IncPKE}.\text{Enc}(\text{inc.pk}, m_b)$, and sends ct^* to $\mathcal{B}_1$. The adversary $\mathcal{B}_1$ forwards ct^* to $\mathcal{A}_1$.

First response phase. The adversary $\mathcal{A}_1$ outputs a state st satisfying $|\text{st}| \leq \ell_{\text{st}}$. The adversary $\mathcal{B}_1$ forwards st to its challenger.

Second response phase. The incompressibility challenger gives $(\text{inc.pk}, \text{inc.sk}, \text{aux}', \text{st})$ to $\mathcal{B}_2$. Since $\text{aux}' = (\text{aux}, T)$, the adversary $\mathcal{B}_2$ recovers the final leaked-coordinate set S and the partial assignment $\tilde{w}_S$ from T . It computes $\text{sk} \leftarrow \text{RPE.Recon}(\text{inc.sk}, S, \tilde{w}_S)$. Finally, it gives $(\text{inc.pk}, \text{sk}, \text{aux}, \text{st})$ to $\mathcal{A}_2$. The adversary $\mathcal{A}_2$ outputs a bit b' , and $\mathcal{B}_2$ outputs the same bit b' .

We now prove that the simulation is perfect. Fix the coins of $\mathcal{A}_1$ and consider the leakage-query transcript. In the real leakage-resilient game, the secret key of the transformed scheme is sampled as $\text{sk} \leftarrow \text{RPE.Enc}(\text{inc.sk})$, and the leakage oracle answers every query $\pi_{Q(j)}$ with $\text{sk}_{Q(j)}$. Equivalently, if S is the final set of distinct leaked coordinates, the adversary receives the indexed partial assignment sk_S .

In the simulated game, $\mathcal{B}_1$ answers the adaptively chosen leakage queries by assigning fresh uniform bits to newly requested coordinates and reusing previously assigned bits on repeated coordinates. Because $|S| \leq \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = \lfloor \ell \cdot \ell_{\text{sk}} \rfloor$, the uniformity property of the RPE implies that $\text{RPE.Enc}(\text{inc.sk})_S$ is uniformly distributed over $\{0, 1\}^S$. Therefore, the leakage transcript seen by $\mathcal{A}_1$ in the simulation is identically distributed to the leakage transcript in the real leakage-resilient game.

Moreover, conditioned on the final leakage transcript $(S, \tilde{w}_S)$, the real secret key is distributed as $\text{RPE.Enc}(\text{inc.sk})$ conditioned on $\text{RPE.Enc}(\text{inc.sk})_S = \tilde{w}_S$. By the reconstructability property of the RPE, the output $\text{sk} \leftarrow \text{RPE.Recon}(\text{inc.sk}, S, \tilde{w}_S)$ has exactly this conditional distribution. Hence the joint distribution of $(S, \tilde{w}_S, \text{sk})$ in the simulation is identical to the corresponding joint distribution in the real leakage-resilient game.

All other components of $\mathcal{A}$'s view are also distributed identically: the public key is inc.pk , the challenge ciphertext is an encryption of m_b under inc.pk , the state st is generated by $\mathcal{A}_1$, and the auxiliary information aux is preserved. Therefore $\mathcal{B}$ perfectly simulates the leakage-resilient incompressibility game for $\mathcal{A}$. Consequently,

$$\Pr[\mathcal{B} \text{ wins the incompressibility game for IncPKE}] = \Pr[\mathcal{A} \text{ wins the leakage-resilient incompressibility game}].$$

Thus, if $\mathcal{A}$ has non-negligible advantage against the transformed scheme, then $\mathcal{B}$ has the same non-negligible advantage against IncPKE, contradicting the incompressibility of IncPKE. $\square$

The same transformation and proof apply verbatim in the secret-key setting by replacing the public-key encryption scheme IncPKE with an incompressible secret-key encryption scheme.

Using Corollary 28, fix a positive rational constant β , let $K := \ell_{\text{sk}}^{\text{IncPKE}}$, and set $\varepsilon(K) := (\log K)^{-1/(2+\beta)}$. To give the construction a rational input, use the dyadic value

$$\widehat{\varepsilon}(K) := 2^{-\lceil \log_2(\lceil \log_2 K \rceil) / (2+\beta) \rceil}.$$

For $K \geq 2$, this is at most $\varepsilon(K)$ and within a constant factor of it. It is computable by integer comparisons since β is fixed and rational. For all sufficiently large K , we simultaneously have $K \geq k_\beta$ and $\widehat{\varepsilon}(K) \leq \varepsilon_\beta$, because the two thresholds depend only on β . For example, it suffices that $K \geq \max\{k_\beta, 2^{\varepsilon_\beta^{-(2+\beta)}}\}$. The uniform construction therefore gives an RPE of input length K with

$$N = O(K \widehat{\varepsilon}(K)^{-(2+\beta)}) = O(K \log K),$$

$$\ell \geq \frac{1}{2} - \frac{\widehat{\varepsilon}(K)}{2} - \frac{1}{N} = \frac{1}{2} - o(1).$$

Both construction and all RPE algorithms run in time polynomial in K . The finitely many smaller values of K can be handled by restricting a fixed larger code to K dimensions. Combining this with Theorem 17, we obtain the following corollary.

Corollary 71. *Assuming the hardness of LWE together with DCR, or assuming DDH together with DCR, there exists a leakage-resilient incompressible PKE scheme with ciphertext-rate $1 - o(1)$, compression-rate $1 - o(1)$, and leakage-rate $1/2 - o(1)$ against adaptive pre-challenge bit-leakage queries whose total union of leaked encoded secret-key coordinates has size at most a $1/2 - o(1)$ fraction of the encoded secret-key length.*

Similarly, assuming LWE or DCR in the secret-key setting, there exists a leakage-resilient incompressible SKE scheme with ciphertext-rate $1 - o(1)$, compression-rate $1 - o(1)$, and leakage-rate $1/2 - o(1)$ against adaptive pre-challenge bit-leakage queries whose total union of leaked encoded secret-key coordinates has size at most a $1/2 - o(1)$ fraction of the encoded secret-key length.

10 Leakage-Resilient Incompressible Signatures

We construct leakage-resilient incompressible signatures in the classical random oracle model; see Section 4.3. Following the template of Guan *et al.* [24], we combine the adaptive random-oracle incompressible encoding of Moran and Wichs [30] with the full-state leakage-resilient signatures of Boyle, Segev, and Wichs [4]. The assumptions are the conjunction

$$(\text{DCR or LWE}) \quad \text{and} \quad \text{DLIN}.$$

The encoding uses the first assumption; the generic bit-string BSW instantiation in Section 3.8 uses the second. In particular, using LWE for the encoding does not make the combined construction post-quantum.

Parameters and oracle. Let $\text{Code} = (\text{Enc}, \text{Dec})$ be the padded encoding in Section 3.4. Given $(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, compute its codeword length $C = \ell_{\text{code}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$ and set

$$n = \lambda + \lceil \log_2(C + 2) \rceil. \quad (5)$$

Use BSW at internal parameter n , with secret-key length $\mu(n)$, leakage budget $L_{\text{BSW}}(n)$, and authentication length $p(n)$ as specified in Equation (2). Set

$$\ell_{\text{sk}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = \mu(n), \quad \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = L_{\text{BSW}}(n).$$

The leakage class consists of all efficient classical oracle computations on the secret signing key, with the total output length bounded by ℓ_{leak} . As usual, for each fixed adversary the combined running time of its stages and leakage computations is polynomial.

There is one global classical oracle $\text{RO} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$, independent of the BSW key generation. Both encoding algorithms, all adversarial stages, and the leakage computations use this same oracle, including before parameters or messages are selected. Its encoding inputs are domain-separated as $\langle \text{MW}, r, j \rangle$; no other oracle functionality is needed. BSW's public admissible hash is an ordinary deterministic function. The oracle is never reset or reprogrammed, and no query transcript is supplied as free auxiliary state.

Our construction. We define $\text{Sig} = (\text{Gen}, \text{Sign}, \text{Verify})$ as follows. Public inputs to the signer are suppressed, as in Section 3.8.

- $\text{Gen}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$: Compute C and n as above and sample $(v, x) \leftarrow \text{BSW.Gen}(1^n)$. Output $\text{vk} = (\lambda, \ell_{\text{msg}}, \ell_{\text{st}}, v)$ and $\text{sk} = x$.
- $\text{Sign}(\text{sk}, m)$: Compute $c \leftarrow \text{Enc}^{\text{RO}}(1^\lambda, 1^{\ell_{\text{st}}}, m)$ and $\sigma \leftarrow \text{BSW.Sign}(x, c)$, using v as public input. Output the message-recovering signature (c, σ) .
- $\text{Verify}(\text{vk}, (c, \sigma))$: Check the prescribed lengths and header, and check $\text{BSW.Verify}(v, c, \sigma) = 1$. If any check fails, output $\perp$. Otherwise compute $m' = \text{Dec}^{\text{RO}}(c)$ and output m' if it is a ℓ_{msg} -bit message; output $\perp$ otherwise.

Correctness and lengths. BSW verifies the signature on c , and decoding recovers m , except with the sum of the two component correctness errors. The BSW algorithms are oracle-free and its key generation is independent of the oracle; the encoding therefore introduces no additional setup requirement for BSW correctness. Its argument proves a true statement using the encryption coins and x as witness. Completeness also holds when the public data are obviously sampled as specified by BSW: a non-negligible, efficiently detectable failure would distinguish this sampling from honest setup. Thus the composed correctness error is negligible for polynomially bounded parameters, also for efficiently selected messages.

The signature length is exactly

$$\ell_{\text{sign}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = C + p(n). \quad (6)$$

Here C includes the encoding's nonce, public key, block data, and parameter header. The term $p(n)$ includes the BSW ciphertext and argument, with their framing and fixed-length padding. The BSW tag is recomputed from c . Its hash description, encryption public key, and argument reference string are in v , sampled by key generation; they are public verification data, not a separate global CRS and not an additional part of each signature. The parameter triple fixes all parsing lengths. Neither the public data nor a copy of them is included in the secret key x , in accordance with BSW's secret-key convention.

Rates and efficiency. For fixed λ and $C \rightarrow \infty$, Equation (5) gives $n = O_\lambda(\log C)$. Since p is a fixed polynomial,

$$\frac{p(n)}{C} = o(1).$$

Using the block parameters of Equation (1), the fixed- λ diagonal limit $\ell_{\text{st}} = \ell_{\text{msg}} \rightarrow \infty$ is therefore

$$\text{SignRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{msg}}}{C + p(n)} = \frac{\min\{r(\lambda), h(\lambda)\}}{d(\lambda)} = 1 - o(1).$$

For fixed $\lambda, \ell_{\text{msg}}$, the limit $\ell_{\text{st}} \rightarrow \infty$ similarly gives

$$\text{CompRate}(\lambda, \ell_{\text{msg}}) = \frac{h(\lambda)}{d(\lambda)} = 1 - o(1).$$

In these two displayed rates, $o(1)$ tends to zero as $\lambda \rightarrow \infty$ *after* taking the stated length limits. For the leakage rate, Equation (2) instead gives the exact fixed- λ limit

$$\text{LeakRate}(\lambda) = \lim_{\ell_{\text{msg}} \rightarrow \infty} \frac{\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})}{\ell_{\text{sk}}(\lambda, \ell_{\text{msg}}, \ell_{\text{msg}})} = \lim_{\ell_{\text{msg}} \rightarrow \infty} (1 - 1/n) = 1.$$

At every finite parameter choice, the unexposed margin is $\kappa(n) + n$ bits. A leakage-rate limit of 1 does not allow leakage of the entire finite secret key.

This parameterization is uniform. All block functions, length bounds, and the internal parameter are computable from the unary input lengths; the algorithms take time polynomial in $\lambda + \ell_{\text{msg}} + \ell_{\text{st}}$. In the security experiment these lengths are polynomially bounded in λ , so $C = \text{poly}(\lambda)$ and $n = \lambda + O(\log \lambda)$, with the constant in the latter bound allowed to depend on the adversary's polynomial. Consequently, BSW security at parameter n yields negligible error in λ , subject only to the polynomial parameter-guessing loss established below. The fixed- λ rate limits do not assert security against unbounded-time adversaries at that fixed parameter.

Theorem 72. *Assume (DCR or LWE) and DLIN. The above construction is an ℓ_{leak} -leakage-resilient incompressible signature scheme in the classical random oracle model, against the leakage class specified above, where*

$$\ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}}) = \mu(n) - \kappa(n) - n.$$

It has signature-rate and compression-rate $1 - o(1)$ and leakage-rate 1 in the fixed- λ sense of Section 4.3, with $o(1)$ tending to zero as $\lambda \rightarrow \infty$.

Proof. Fix a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for the experiment in Section 4.3. It first chooses the unary lengths, then receives vk , makes its leakage queries, and fixes aux before any signing query. It then makes adaptive signing queries and produces st of length at most ℓ_{st} ; finally $\mathcal{A}_2(\text{vk}, \text{aux}, \text{st})$ outputs a signature. Oracle queries are allowed throughout. Let $Q = Q(\lambda) \geq 1$ be a polynomial upper bound on the number of signing queries. Write $c_1, \dots, c_q$ for the codewords signed in the actual execution, where $q \leq Q$. A successful output parses as (c', σ') with a valid BSW signature and a valid decoded message. Let p_{new} be the probability of success with $c' \notin \{c_1, \dots, c_q\}$, and p_{old} that with $c' \in \{c_1, \dots, c_q\}$.

A codeword that was never signed. We first describe the BSW reduction for matching internal parameters, and then account for the parameter choice. It runs $\mathcal{A}_1$ through parameter selection and gives it its BSW challenger's verification key, augmented by the chosen public lengths. It answers leakage using its leakage oracle. For each signing query m_i , it computes $c_i \leftarrow \text{Enc}^{\text{RO}}(1^\lambda, 1^{\ell_{\text{st}}}, m_i)$, asks the BSW challenger to sign c_i , and returns (c_i, σ_i) . Finally it runs $\mathcal{A}_2$ with exactly $(\text{vk}, \text{aux}, \text{st})$ and outputs the ordinary BSW forgery (c', σ') . Every success in the new-codeword case is an unforgeability violation; changing the signature on an old codeword is not counted in this case.

Here the challenger is the *standard-model* BSW challenger, so we must also implement all oracle calls, including hidden calls inside leakage computations. For the fixed adversary, bound the total number of classical oracle evaluations by a polynomial T , counting calls made by both stages, the encoding and verification algorithms, and all leakage evaluations. Also bound their input lengths by a polynomial. Sample one T -wise independent random function on that finite input domain with output length λ , independently of the BSW challenger, and use it for every oracle evaluation in the simulation. Such a function has a polynomial-size description and polynomial-time evaluation: embed length-delimited inputs in a sufficiently large binary field, evaluate a random polynomial of degree $T - 1$, and project to λ output bits. Every execution making at most T adaptive classical evaluations has exactly the same joint distribution as with a random oracle. This includes evaluations on secret-dependent inputs inside leakage computations.

For a leakage request f_j^{RO} , submit the ordinary efficient function that evaluates f_j with this fixed function description hardwired. BSW's full-state leakage interface can recover x from the key-generation state and apply this function to x . The returned output has exactly the original length, and the total budget remains $L_{\text{BSW}}(n)$. The simulator uses the same function for all other calls; no hidden leakage-query transcript needs to be learned. Its description is private reduction state and is never given to either stage of $\mathcal{A}$. In particular it is not added to aux or st . This gives an exact classical simulation without an additional oracle security premise for BSW. The query bound is fixed for the reduction, not as a restriction on the scheme.

The internal parameter is selected through the adversary's lengths, whereas BSW's game fixes n first. A uniform BSW adversary on input 1^n chooses λ uniformly in $[n]$, starts the preceding simulation with this λ , and aborts if Equation (5) does not yield its input n . It reveals the challenger's key only after this check. Before that point $\mathcal{A}$ sees no key, so sampling the BSW key earlier does not change its view. The oracle simulation uses polynomial bounds in n dominating the running time for every $\lambda \leq n$. For each fixed λ , let I_λ be the polynomial-size range of possible internal parameters, all at least λ and at most $\lambda + O(\log \lambda)$. If $\epsilon_{\text{BSW}}(n)$ denotes the success probability of this single uniform reduction, the matching branches give

$$p_{\text{new}}(\lambda) \leq \sum_{n \in I_\lambda} n \epsilon_{\text{BSW}}(n) = \text{negl}(\lambda).$$

The abort is solely a parameter mismatch; it costs only this polynomial factor. In particular, message length is not identified with n .

A previously signed codeword. Construct an encoding adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ using the *actual global oracle of its encoding challenger*. Every oracle call in this reduction, including calls made by the simulated leakage computations, is forwarded to that same oracle. Choose $i^* \leftarrow [Q]$ independently. Run $\mathcal{A}_1$ through parameter selection, compute C, n , generate an honest BSW

key pair, and supply vk . Answer all leakage queries honestly using x and the challenger oracle. The adversary then fixes aux , before any signature has been returned.

Answer signing queries with honest encodings and BSW signatures, except at query i^* . On that query, submit to the encoding challenger

$$(1^{\ell_{\text{st}}}, m_{i^*}, \text{aux}'), \quad \text{aux}' = (\text{vk}, \text{aux}),$$

receive its challenge codeword c_{i^*} , sign it using x , and return the pair. Continue the simulation, using honest encoding and signing for all subsequent queries. The target message may depend on the verification key, leakage, earlier signatures, and earlier oracle queries. The auxiliary input aux' is fixed before the target encoding is sampled; this timing is permitted by adaptive RO encoding security. The reduction's first stage may retain its own working state while it finishes the simulation, but sends only $\mathcal{A}_1$'s final st as the compressed state. Its length is still at most ℓ_{st} .

In the second phase, $\mathcal{B}_2$ receives $(\text{st}, m_{i^*}, \text{aux}')$, extracts (vk, aux) , and runs $\mathcal{A}_2(\text{vk}, \text{aux}, \text{st})$ with the same challenger oracle. It outputs the codeword component c' of the resulting signature, or an arbitrary string if parsing fails. It sends no signature-key material or oracle transcript between stages. The extra target-message input supplied by the encoding game need not be used by $\mathcal{A}_2$.

If fewer than i^* signing queries occur, the first stage submits a dummy message of the chosen length with the already fixed auxiliary input and uses an arbitrary legal compressed state; this branch need not succeed. For every reached index, substituting the challenger's encoding for the honest encoding gives exactly the original distribution. Thus the view of $\mathcal{A}$ is independent of i^* , and, on every successful old-codeword execution, at least one index matches c' . This remains true when several signed codewords coincide. Consequently,

$$\Pr[\mathcal{B} \text{ reconstructs its challenge exactly}] \geq p_{\text{old}}(\lambda)/Q(\lambda).$$

Adaptive encoding security makes this probability negligible, so p_{old} is negligible. This reduction uses single-target adaptive security; it does not assume that composability by itself permits an online choice of an entire message list.

The success probability of $\mathcal{A}$ is $p_{\text{new}} + p_{\text{old}}$. The first term is bounded by the polynomially weighted BSW errors above, and the second by Q times the encoding error. All query and running-time bounds are polynomial in λ , so their sum is negligible. Correctness and the rates were established above. $\square$

The selective CRS alternative. The CRS encoding of Moran and Wichs [30] provides only selective incompressibility. Substituting it in this composition therefore supports only a restricted selective-message or selective-target incompressible signature formulation, in which the target message and relevant auxiliary information are fixed before the encoding CRS is revealed. In a multiple-message formulation, the complete relevant message list and auxiliary information may need to be fixed at that time. This timing condition alone does not provide unrestricted composition under a reused CRS: Moran and Wichs obtain CRS composability with fresh independent strings for the individual encodings. A formal multiple-message variant must also specify the setup and the composition guarantee it uses. The adaptive target selection in the proof above would not be permitted. This observation is separate from our ROM theorem; it is neither adaptive incompressible security nor ordinary selective unforgeability, and we do not assert a formal CRS theorem here.

11 Impossibility Results for Rate-1 $- o(1)$ Incompressible Encryption Schemes

In this section, we establish black-box barriers for achieving rate $1 - o(1)$ incompressible encryption. We first show that incompressible PKE schemes with ciphertext-rate $1 - o(1)$ and

sufficiently short secret keys cannot have their security based on a standard assumption in a black-box manner. This impossibility is meaningful even when the public key is allowed to be arbitrarily large. The barrier only depends on the secret-key length and the ciphertext-over-state overhead, but not on the public-key length.

We then prove an analogous barrier for leakage-resilient incompressible SKE. In this setting, the relevant obstruction is the combination of large leakage and small ciphertext overhead. More precisely, if the residual key length after leakage, namely $\ell_{\text{sk}} - \ell_{\text{leak}}$, together with the ciphertext-over-state overhead $\ell_{\text{ct}} - \ell_{\text{st}}$, is too small compared to the message length, then leakage-resilient incompressible security cannot be obtained from any standard assumption by a black-box reduction. This impossibility holds even in a strong setting where the adversary is not given access to an encryption oracle.

11.1 Impossibility Result for Incompressible PKE with Short Keys

We show that for any rate $1 - o(1)$ incompressible PKE scheme, if the secret key is too short relative to the message length and to the ciphertext-over-state overhead, then the security of the scheme cannot be based on a standard assumption in a black-box manner. This resolves an open question from [24] in the short-key regime. Throughout this subsection, we focus on parameter regimes where $\ell_{\text{msg}} = \omega(\log \lambda)$.

Theorem 73. *Let $\text{IncPKE} = (\text{IncPKE.Setup}, \text{IncPKE.Enc}, \text{IncPKE.Dec})$ be a PKE scheme with deterministic decryption and almost perfect correctness. Fix any parameter regime $\ell_{\text{msg}} = \ell_{\text{msg}}(\lambda)$ and $\ell_{\text{st}} = \ell_{\text{st}}(\lambda)$ with $\ell_{\text{msg}} = \omega(\log \lambda)$. Let $\ell_{\text{ct}} = \ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$ and $\ell_{\text{sk}} = \ell_{\text{sk}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$ denote the ciphertext length and secret-key length in this regime. Suppose there exists a function $\eta = \eta(\lambda) = \omega(\log \lambda)$ such that, for all sufficiently large λ , we have $\ell_{\text{sk}} + \ell_{\text{ct}} - \ell_{\text{st}} \leq \ell_{\text{msg}} - \eta$. Then there is no secure cryptographic game $\mathcal{G}$ such that there is a black-box reduction deriving the incompressible security of IncPKE in this parameter regime from the security of $\mathcal{G}$.*

Thus, a necessary, but weaker, asymptotic consequence for any scheme that avoids this black-box barrier is $\ell_{\text{sk}} + \ell_{\text{ct}} - \ell_{\text{st}} \geq \ell_{\text{msg}} - o(\ell_{\text{msg}})$ in the relevant parameter regime; the theorem itself excludes precisely those regimes having a superlogarithmic deficit. In particular, if $\ell_{\text{ct}} = \ell_{\text{msg}} + o(\ell_{\text{msg}})$ and $\ell_{\text{st}} = \ell_{\text{msg}} - o(\ell_{\text{msg}})$, then $\ell_{\text{ct}} - \ell_{\text{st}} = o(\ell_{\text{msg}})$, and therefore avoiding the barrier requires $\ell_{\text{sk}} \geq \ell_{\text{msg}} - o(\ell_{\text{msg}})$. Thus, in the rate $1 - o(1)$ ciphertext regime, this necessary condition requires the secret key to be essentially as large as the message.

Before giving the proof, we make a minor notational refinement of the incompressible PKE security experiment. In Definition 15, the first-phase adversary is written as a single algorithm that first chooses the challenge messages and auxiliary information and later compresses the challenge ciphertext into a state. In the proof below, we split this first phase into two separate components. Thus, we write the adversary as $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_0$, on input $(1^\lambda, \text{pk})$, outputs (m_0, m_1, aux) , $\mathcal{A}_1$, on input $(\text{pk}, \text{aux}, \text{ct}^*)$, outputs the compressed state st , and $\mathcal{A}_2$, on input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, outputs the final guess. This is only a syntactic refinement of the security experiment in Definition 15: any such three-component adversary can be viewed as a standard two-phase adversary by combining $\mathcal{A}_0$ and $\mathcal{A}_1$ into the first phase, and conversely any two-phase adversary can be decomposed in this way by separating its message-selection and state-compression steps. Therefore, this change does not alter the security notion; it only makes the simulatable attack easier to describe formally.

Proof. If $\ell_{\text{st}} \geq \ell_{\text{ct}}$, there is already a direct efficient attack: the first-stage adversary stores the entire challenge ciphertext (padding its state if necessary), and after receiving the secret key it decrypts and identifies the challenge message. Almost perfect correctness gives this adversary overwhelming success probability. A black-box reduction from a secure game would turn this efficient attacker into an efficient attacker against that game, contradicting its security. We may therefore restrict the remainder of the proof to $\ell_{\text{st}} < \ell_{\text{ct}}$.

It suffices to construct an average-case simulatable attack, in the sense of Definition 35, against the incompressible security of IncPKE in the parameter regime specified in the theorem.

Fix the parameter regime from the theorem and write $\Delta = \ell_{\text{ct}} - \ell_{\text{st}}$. By assumption, $\Delta \geq 0$. Define $\tau = \ell_{\text{msg}} - \ell_{\text{sk}} - \Delta$. The hypothesis gives $\tau \geq \eta = \omega(\log \lambda)$. Finally, define $\gamma = (\Delta + \tau/2)/\ell_{\text{msg}}$. For each fixed λ , we have $0 < \gamma < 1$. Indeed, $\Delta \geq 0$ and $\tau > 0$ imply $\gamma > 0$, while $\Delta + \tau = \ell_{\text{msg}} - \ell_{\text{sk}} \leq \ell_{\text{msg}}$ implies $\gamma \ell_{\text{msg}} = \Delta + \tau/2 < \ell_{\text{msg}}$. We will repeatedly use the identities $\gamma \ell_{\text{msg}} - \Delta = \tau/2$ and $(1 - \gamma)\ell_{\text{msg}} - \ell_{\text{sk}} = \tau/2$.

We use the standard convention that the message space has size $2^{\ell_{\text{msg}}}$, the secret-key space has size at most $2^{\ell_{\text{sk}}}$, and the ciphertext space has size at most $2^{\ell_{\text{ct}}}$. We also identify the ciphertext space with its canonical encoding whenever invoking the set-image bound from Theorem 39.

Recall that a simulatable attack consists of an inefficient attacker against the primitive and an efficient stateful simulator. We now define both.

Inefficient adversary. Let G be the set of all functions $g : \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \mathcal{M}_{\ell_{\text{msg}}} \times \mathcal{M}_{\ell_{\text{msg}}}$, and let H be the set of all functions $h : \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_{\text{st}}}$. For every $g \in G$ and $h \in H$, define the inefficient adversary $\mathcal{A}^{g,h} = (\mathcal{A}_0^{g,h}, \mathcal{A}_1^{g,h}, \mathcal{A}_2^{g,h})$ as follows.

1. $\mathcal{A}_0^{g,h}$: On input $(1^\lambda, \text{pk})$, compute $(m_0, m_1) = g(\text{pk})$, set $\text{aux} = (m_0, m_1)$, and output (m_0, m_1, aux) .
2. $\mathcal{A}_1^{g,h}$: On input $(\text{pk}, \text{aux}, \text{ct}^*)$, output $\text{st} = h(\text{ct}^*)$.
3. $\mathcal{A}_2^{g,h}$: On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, first compute $(m_0, m_1) = g(\text{pk})$. If $\text{aux} \neq (m_0, m_1)$, output $\perp$. Otherwise, define $C^{h,\text{st}} = \{\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} : h(\text{ct}) = \text{st}\}$ and

$$M^{h,\text{st},\text{pk},\text{sk}} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C^{h,\text{st}} \text{ such that } m = \text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct})\}.$$

If $\{m_0, m_1\} \cap M^{h,\text{st},\text{pk},\text{sk}} = \emptyset$ or $\{m_0, m_1\} \subseteq M^{h,\text{st},\text{pk},\text{sk}}$, output $\perp$. Otherwise, if $m_0 \in M^{h,\text{st},\text{pk},\text{sk}}$, output 0, and otherwise output 1.

We first show that this inefficient adversary wins the incompressible PKE game with probability negligibly close to 1, on average over the choice of g and h .

Lemma 74. *There exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\Pr_{g,h,\text{Game}} [\mathcal{A}^{g,h}(1^\lambda) \text{ wins the incompressible PKE game}] \geq 1 - \text{negl}(\lambda).$$

Proof. Let $(m_0, m_1) = g(\text{pk})$ be the challenge messages chosen by $\mathcal{A}_0^{g,h}$, set $\text{aux} = (m_0, m_1)$, let $b \leftarrow \{0, 1\}$ be the challenge bit, $\text{ct}^* \leftarrow \text{IncPKE.Enc}(\text{pk}, m_b)$ be the challenge ciphertext, and $\text{st} = h(\text{ct}^*)$ be the state output by $\mathcal{A}_1^{g,h}(\text{pk}, \text{aux}, \text{ct}^*)$.

By almost perfect correctness, except with negligible probability over the generation of (pk, sk) and the randomness of encryption, we have $\text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct}^*) = m_b$. On this event, $m_b \in M^{h,\text{st},\text{pk},\text{sk}}$, because ct^* itself belongs to $C^{h,\text{st}} = \{\text{ct} : h(\text{ct}) = \text{st}\}$. Hence the adversary fails to output the correct bit only if $m_{1-b} \in M^{h,\text{st},\text{pk},\text{sk}}$. Indeed, if $m_b \in M^{h,\text{st},\text{pk},\text{sk}}$ but $m_{1-b} \notin M^{h,\text{st},\text{pk},\text{sk}}$, then exactly one of the two messages m_0, m_1 lies in $M^{h,\text{st},\text{pk},\text{sk}}$, and the adversary outputs the corresponding bit, which is the correct bit. We now bound the probability that the other message also lies in this set.

Condition on $\text{pk}, \text{sk}, b, m_b$, the encryption randomness used to generate ct^* , and the resulting ciphertext ct^* . After this conditioning, the ciphertext ct^* is fixed. The remaining randomness in the event $m_{1-b} \in M^{h,h(\text{ct}^*),\text{pk},\text{sk}}$ is over the random function h and over the non-challenge message m_{1-b} . Since $g(\text{pk})$ is sampled as a pair of independent uniform messages and we have conditioned only on the coordinate m_b , the other coordinate m_{1-b} remains uniform over $\mathcal{M}_{\ell_{\text{msg}}}$. Moreover, m_{1-b} is independent of h . Therefore, for every fixed choice of h , the probability over m_{1-b} that m_{1-b} lands in the set $M^{h,h(\text{ct}^*),\text{pk},\text{sk}}$ is at most $|M^{h,h(\text{ct}^*),\text{pk},\text{sk}}|/2^{\ell_{\text{msg}}}$. Averaging this bound over the remaining randomness of h , we get

$$\Pr[m_{1-b} \in M^{h,h(\text{ct}^*),\text{pk},\text{sk}}] \leq \frac{\mathbb{E}_h[|M^{h,h(\text{ct}^*),\text{pk},\text{sk}}|]}{2^{\ell_{\text{msg}}}}.$$

Here, the probability on the left is over the remaining randomness of both m_{1-b} and h , after the conditioning above, while the expectation on the right is only over the uniformly random function h .

Moreover, for every fixed h , every message in $M^{h,h(\text{ct}^*),\text{pk},\text{sk}}$ is obtained by decrypting some ciphertext in $h^{-1}(h(\text{ct}^*))$. Therefore, the number of distinct messages in $M^{h,h(\text{ct}^*),\text{pk},\text{sk}}$ is at most the number of ciphertexts in that preimage, namely $|M^{h,h(\text{ct}^*),\text{pk},\text{sk}}| \leq |h^{-1}(h(\text{ct}^*))|$. Thus, it is enough to bound the expected size of this random preimage. For fixed ct^* and uniformly random $h : \mathcal{C}_{\lambda,\ell_{\text{msg}},\ell_{\text{st}}} \rightarrow \{0,1\}^{\ell_{\text{st}}}$, the ciphertext ct^* always belongs to $h^{-1}(h(\text{ct}^*))$, contributing 1 to its size. For every other ciphertext $\text{ct} \neq \text{ct}^*$, the value $h(\text{ct})$ is uniform and independent of $h(\text{ct}^*)$, and hence ct belongs to the same preimage exactly when $h(\text{ct}) = h(\text{ct}^*)$, which happens with probability $2^{-\ell_{\text{st}}}$. Therefore, by linearity of expectation,

$$\mathbb{E}_h[|h^{-1}(h(\text{ct}^*))|] = 1 + \frac{|\mathcal{C}_{\lambda,\ell_{\text{msg}},\ell_{\text{st}}}| - 1}{2^{\ell_{\text{st}}}} \leq 1 + 2^{\ell_{\text{ct}} - \ell_{\text{st}}} = 1 + 2^\Delta \leq 2^{\Delta+1}.$$

Combining the previous inequalities, we obtain

$$\Pr[m_{1-b} \in M^{h,h(\text{ct}^*),\text{pk},\text{sk}}] \leq \frac{2^{\Delta+1}}{2^{\ell_{\text{msg}}}} = 2^{-(\ell_{\text{msg}} - \Delta) + 1} \leq 2^{-\tau+1},$$

where the last inequality uses $\ell_{\text{msg}} - \Delta = \ell_{\text{sk}} + \tau \geq \tau$. This is negligible because $\tau = \omega(\log \lambda)$. Combining this with the almost perfect correctness error gives the lemma. $\square$

For fixed (g, h) , let $W_{g,h}(\lambda)$ denote the winning probability of $\mathcal{A}^{g,h}$ in the incompressible PKE game, where the probability is over the randomness of the game. By Lemma 74, $\mathbb{E}_{g,h}[1 - W_{g,h}(\lambda)] \leq \text{negl}(\lambda)$. Hence, by Markov's inequality, $\Pr_{g,h}[W_{g,h}(\lambda) < 3/4] \leq 4\text{negl}(\lambda)$. Therefore, except with negligible probability over the choice of (g, h) , the fixed attacker $\mathcal{A}^{g,h}$ wins the incompressible PKE game with probability at least $3/4$. Since the incompressible PKE advantage is the normalized prediction advantage $\text{Adv}_{\text{IncPKE}}^{\mathcal{A}^{g,h}}(\lambda) = |2W_{g,h}(\lambda) - 1|$, we obtain $\text{Adv}_{\text{IncPKE}}^{\mathcal{A}^{g,h}}(\lambda) \geq 1/2$ except with negligible probability over (g, h) .

Simulator. We now define the efficient stateful simulator $\text{Sim} = (\text{Sim}_0, \text{Sim}_1, \text{Sim}_2)$. The simulator maintains two tables Q_0 and Q_1 , used to lazily sample the functions g and h .

1. Sim_0 : On input $(1^\lambda, \text{pk})$, check whether Q_0 already contains an entry of the form $(\text{pk}, (m_0, m_1))$. If so, return (m_0, m_1, aux) , where $\text{aux} = (m_0, m_1)$. Otherwise, sample $(m_0, m_1) \leftarrow \mathcal{M}_{\ell_{\text{msg}}} \times \mathcal{M}_{\ell_{\text{msg}}}$, store $(\text{pk}, (m_0, m_1))$ in Q_0 , set $\text{aux} = (m_0, m_1)$, and return (m_0, m_1, aux) .
2. Sim_1 : On input $(\text{pk}, \text{aux}, \text{ct}^*)$, check whether Q_1 already contains an entry of the form (ct^*, st) . If so, return st . Otherwise, sample $\text{st} \leftarrow \{0, 1\}^{\ell_{\text{st}}}$, store (ct^*, st) in Q_1 , and return st .
3. Sim_2 : On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, first check whether Q_0 contains an entry of the form $(\text{pk}, (m_0, m_1))$ and whether $\text{aux} = (m_0, m_1)$. If either check fails, return $\perp$. Otherwise, define $C_{\text{Sim}} = \{\text{ct} : (\text{ct}, \text{st}) \in Q_1\}$ and $M_{\text{Sim}} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C_{\text{Sim}} \text{ such that } m = \text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct})\}$. If $\{m_0, m_1\} \cap M_{\text{Sim}} = \emptyset$ or $\{m_0, m_1\} \subseteq M_{\text{Sim}}$, return $\perp$. Otherwise, if $m_0 \in M_{\text{Sim}}$, return 0, and otherwise return 1.

The simulator is efficient because it stores and processes only the polynomially many queries made by the oracle machine interacting with it. We next show that no polynomial-query oracle machine can distinguish oracle access to $\mathcal{A}^{g,h}$, for random (g, h) , from oracle access to Sim .

Lemma 75. *For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R}[R^{\mathcal{A}^{g,h}}(1^\lambda) \rightarrow 1] - \Pr_{R,\text{Sim}}[R^{\text{Sim}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. We introduce a sequence of hybrid adversaries between $\mathcal{A}$ and Sim . First, we record the simple combinatorial facts needed in the hybrid argument. Throughout, we use the notions of γ -bad messages, γ -good messages, and γ -bad key pairs from Definition 16, instantiated with $\alpha = \gamma$.

Claim 1. For every $\lambda, \ell_{\text{msg}}, \ell_{\text{st}} \in \mathbb{N}$ and every $\gamma \in (0, 1)$, no γ -bad key pair $(\text{pk}, \text{sk}) \in \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \times \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$ exists.

Proof. Assume, for contradiction, that (pk, sk) is a γ -bad key pair. Let

$$\mathfrak{M}_{(\text{pk}, \text{sk})} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : m \text{ is } \gamma\text{-bad with respect to } (\text{pk}, \text{sk})\}.$$

By assumption, $|\mathfrak{M}_{(\text{pk}, \text{sk})}| > 2^{\gamma \ell_{\text{msg}}}$. Moreover, for every $m \in \mathfrak{M}_{(\text{pk}, \text{sk})}$, the set

$$C_m = \{\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} : \text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct}) = m\}$$

satisfies $|C_m| > |\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \cdot 2^{-\gamma \ell_{\text{msg}}}$ by Definition 16. Since decryption is deterministic, the sets $\{C_m\}_{m \in \mathfrak{M}_{(\text{pk}, \text{sk})}}$ are pairwise disjoint. Hence $|\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \geq \sum_{m \in \mathfrak{M}_{(\text{pk}, \text{sk})}} |C_m| > 2^{\gamma \ell_{\text{msg}}} \cdot |\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \cdot 2^{-\gamma \ell_{\text{msg}}} = |\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}|$, a contradiction. $\square$

Claim 2. For every $(\text{pk}, \text{sk}) \in \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \times \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, we have

$$\Pr_{m \leftarrow \mathcal{M}_{\ell_{\text{msg}}}} [m \text{ is } \gamma\text{-bad with respect to } (\text{pk}, \text{sk})] \leq 2^{-(1-\gamma)\ell_{\text{msg}}}.$$

Moreover, for every $\text{pk} \in \mathcal{PK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, we have

$$\Pr_{m \leftarrow \mathcal{M}_{\ell_{\text{msg}}}} [m \text{ is } \gamma\text{-good with respect to } \text{pk}] \geq 1 - 2^{-\tau/2}.$$

In particular, $2^{-\tau/2}$ is negligible in λ .

Proof. The first inequality follows immediately from Claim 1: otherwise (pk, sk) would be a γ -bad key pair. For the second inequality, take a union bound over all secret keys. For every fixed pk , the probability that a uniformly random $m \leftarrow \mathcal{M}_{\ell_{\text{msg}}}$ is not γ -good with respect to pk is at most $|\mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \cdot 2^{-(1-\gamma)\ell_{\text{msg}}} \leq 2^{\ell_{\text{sk}}} \cdot 2^{-(1-\gamma)\ell_{\text{msg}}} = 2^{-((1-\gamma)\ell_{\text{msg}} - \ell_{\text{sk}})} = 2^{-\tau/2}$, where the last equality uses $(1-\gamma)\ell_{\text{msg}} - \ell_{\text{sk}} = \tau/2$. $\square$

We now present the hybrid adversaries.

Adversary $\mathcal{B}$. Define $\mathcal{B}^{g,h} = (\mathcal{B}_0^{g,h}, \mathcal{B}_1^{g,h}, \mathcal{B}_2^{g,h})$ as follows. The adversary maintains a table Q_0 , initially empty.

1. $\mathcal{B}_0^{g,h}$: On input $(1^\lambda, \text{pk})$, compute $(m_0, m_1, \text{aux}) \leftarrow \mathcal{A}_0^{g,h}(1^\lambda, \text{pk})$, store $(\text{pk}, (m_0, m_1))$ in Q_0 , and return (m_0, m_1, aux) .
2. $\mathcal{B}_1^{g,h}$: On input $(\text{pk}, \text{aux}, \text{ct}^*)$, return $\mathcal{A}_1^{g,h}(\text{pk}, \text{aux}, \text{ct}^*)$.
3. $\mathcal{B}_2^{g,h}$: On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, first check whether Q_0 contains an entry of the form $(\text{pk}, \text{aux} = (m_0, m_1))$. If no such entry exists, return $\perp$. Otherwise, output $\mathcal{A}_2^{g,h}(\text{pk}, \text{sk}, \text{aux}, \text{st})$.

Lemma 76. For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,

$$\left| \Pr_{g,h,R} [R^{\mathcal{A}^{g,h}}(1^\lambda) \rightarrow 1] - \Pr_{g,h,R} [R^{\mathcal{B}^{g,h}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. We couple the two experiments using the same (g, h) and the same randomness for R . The executions can differ only if R makes a query $(\text{pk}, \text{sk}, \text{aux}, \text{st})$ to $\mathcal{A}_2/\mathcal{B}_2$ such that pk was never queried to $\mathcal{A}_0/\mathcal{B}_0$, but $\mathcal{A}_2^{g,h}(\text{pk}, \text{sk}, \text{aux}, \text{st}) \neq \perp$.

Analyze these guesses along the unconditioned $\mathcal{B}$ execution. Its final component never inspects $g(\text{pk})$ at an unrecorded public key. Thus, conditioned on any prefix of this execution, such a value remains uniform over $\mathcal{M}_{\ell_{\text{msg}}} \times \mathcal{M}_{\ell_{\text{msg}}}$, independently of the query formed next. The probability that its guessed aux equals $g(\text{pk})$ is at most $2^{-2\ell_{\text{msg}}}$. A first disagreement of the coupled executions requires one of these guesses to succeed, since the first check of $\mathcal{A}_2$ must pass. A union bound over the polynomially many guesses along $\mathcal{B}$ proves the claim, since $\ell_{\text{msg}} = \omega(\log \lambda)$. $\square$

Adversary $\mathcal{C}$. Define $\mathcal{C}^{g,h} = (\mathcal{C}_0^{g,h}, \mathcal{C}_1^{g,h}, \mathcal{C}_2^{g,h})$ as follows. The adversary maintains two tables Q_0 and Q_1 , initially empty.

1. $\mathcal{C}_0^{g,h}$: On input $(1^\lambda, \text{pk})$, compute $(m_0, m_1, \text{aux}) \leftarrow \mathcal{A}_0^{g,h}(1^\lambda, \text{pk})$, store $(\text{pk}, (m_0, m_1))$ in Q_0 , and return (m_0, m_1, aux) .
2. $\mathcal{C}_1^{g,h}$: On input $(\text{pk}, \text{aux}, \text{ct}^*)$, compute $\text{st} \leftarrow \mathcal{A}_1^{g,h}(\text{pk}, \text{aux}, \text{ct}^*)$, store (ct^*, st) in Q_1 , and return st .
3. $\mathcal{C}_2^{g,h}$: On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, first check whether Q_0 contains an entry of the form $(\text{pk}, \text{aux} = (m_0, m_1))$. If no such entry exists, return $\perp$. If $\text{aux} \neq (m_0, m_1)$, return $\perp$. Otherwise, define $C_C = \{\text{ct} : (\text{ct}, \text{st}) \in Q_1\}$ and $M_C = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C_C \text{ such that } m = \text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct})\}$. If $\{m_0, m_1\} \cap M_C = \emptyset$, return $\perp$. Otherwise, output $\mathcal{A}_2^{g,h}(\text{pk}, \text{sk}, \text{aux}, \text{st})$.

Lemma 77. *For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R}[R^{\mathcal{B}^{g,h}}(1^\lambda) \rightarrow 1] - \Pr_{g,h,R}[R^{\mathcal{C}^{g,h}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. We analyze the coupling from the table-only execution $R^{\mathcal{D}^{g,h}}$, where $\mathcal{D}$ is defined below, and sample g and h lazily in the tables Q_0 and Q_1 . This execution reveals values of h only at ciphertexts queried to $\mathcal{D}_1$; its final component never inspects an unrecorded value of h .

Let q be a polynomial upper bound on the number of oracle queries. Each new public key queried to $\mathcal{D}_0$ receives an independent uniform message pair, conditional on the preceding execution. By Claim 2 and a union bound, the probability that any pair recorded in Q_0 is not γ -good with respect to its public key is at most $2q \cdot 2^{-\tau/2}$. We do not condition on the complement of this event.

For the i -th second-stage query $(\text{pk}_i, \text{sk}_i, \text{aux}_i, \text{st}_i)$ in the table-only execution whose initial checks pass and whose recorded pair is γ -good, write $\text{aux}_i = (m_0^{(i)}, m_1^{(i)})$ and set

$$T_{i,b} = \{\text{ct} : \text{IncPKE.Dec}(\text{pk}_i, \text{sk}_i, \text{ct}) = m_b^{(i)}\} \quad (b \in \{0, 1\}).$$

Then $|T_{i,b}| \leq 2^{\ell_{\text{ct}} - \gamma \ell_{\text{msg}}}$. Let $I_i \subseteq \{0, 1\}$ contain exactly those indices whose messages are obtained by decrypting ciphertexts already recorded in Q_1 with state st_i , and define

$$S_i = \begin{cases} T_{i,0} \cup T_{i,1}, & I_i = \emptyset, \\ T_{i,1-b}, & I_i = \{b\}, \\ \emptyset, & I_i = \{0, 1\}. \end{cases}$$

For a query that fails these checks, or an absent i -th query, set $S_i = \emptyset$ (with an arbitrary target for an absent query). Thus on every execution $|S_i| \leq 2 \cdot 2^{\ell_{\text{ct}} - \gamma \ell_{\text{msg}}}$, and no previously queried point in S_i has h -image st_i .

Couple the $\mathcal{B}$, $\mathcal{C}$, and table-only $\mathcal{D}$ executions until their first disagreement. All prior answers, and hence the next query, agree. Unless a recorded pair is bad, a case analysis on I_i shows that a disagreement implies $\text{st}_i \in h(S_i)$. If $I_i = \emptyset$, an unrecorded ciphertext must supply the unique message found using the full fiber. If $I_i = \{b\}$, an unrecorded ciphertext must supply $m_{1-b}^{(i)}$ for any full-fiber answer to differ from the table-only answer. If $I_i = \{0, 1\}$, every component returns $\perp$.

The sets S_i and targets st_i are generated along the table-only execution, which accesses h only through point queries. For each fixed i , stop that execution immediately after (S_i, st_i) is formed (or when the execution ends). Testing goodness and forming the set require only finite enumeration and no additional h -queries; the set-image theorem allows this computation to be unbounded. No future event is conditioned on, so the unqueried h -values remain uniform conditional on this prefix. In this truncated execution no queried point in S_i has image st_i , so Theorem 39, applied to this prefix with one output pair, bounds the probability of the i -th event. A union bound over the polynomially many possible queries bounds the probability of any set-image event by

$$\text{poly}(\lambda) \cdot \frac{2 \cdot 2^{\ell_{\text{ct}} - \gamma \ell_{\text{msg}}}}{2^{\ell_{\text{st}}}} = \text{poly}(\lambda) \cdot 2^{-\tau/2+1}.$$

Together with the separately bounded bad-message event, this is negligible. In particular, the coupled $\mathcal{B}$ and $\mathcal{C}$ executions differ only with negligible probability. $\square$

Adversary $\mathcal{D}$. Define $\mathcal{D}^{g,h} = (\mathcal{D}_0^{g,h}, \mathcal{D}_1^{g,h}, \mathcal{D}_2^{g,h})$. The adversary $\mathcal{D}$ is identical to $\mathcal{C}$ in its first two components, but its third component answers only using the information recorded in Q_0 and Q_1 .

1. $\mathcal{D}_0^{g,h}$: On input $(1^\lambda, \text{pk})$, compute $(m_0, m_1, \text{aux}) \leftarrow \mathcal{A}_0^{g,h}(1^\lambda, \text{pk})$, store $(\text{pk}, (m_0, m_1))$ in Q_0 , and return (m_0, m_1, aux) .
2. $\mathcal{D}_1^{g,h}$: On input $(\text{pk}, \text{aux}, \text{ct}^*)$, compute $\text{st} \leftarrow \mathcal{A}_1^{g,h}(\text{pk}, \text{aux}, \text{ct}^*)$, store (ct^*, st) in Q_1 , and return st .
3. $\mathcal{D}_2^{g,h}$: On input $(\text{pk}, \text{sk}, \text{aux}, \text{st})$, first check whether Q_0 contains an entry of the form $(\text{pk}, (m_0, m_1))$. If no such entry exists, return $\perp$. If $\text{aux} \neq (m_0, m_1)$, return $\perp$. Otherwise, define $C_{\mathcal{D}} = \{\text{ct} : (\text{ct}, \text{st}) \in Q_1\}$ and $M_{\mathcal{D}} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C_{\mathcal{D}} \text{ such that } m = \text{IncPKE.Dec}(\text{pk}, \text{sk}, \text{ct})\}$. If $\{m_0, m_1\} \cap M_{\mathcal{D}} = \emptyset$ or $\{m_0, m_1\} \subseteq M_{\mathcal{D}}$, return $\perp$. Otherwise, if $m_0 \in M_{\mathcal{D}}$, return 0, and otherwise return 1.

Lemma 78. *For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R} [R^{\mathcal{C}^{g,h}}(1^\lambda) \rightarrow 1] - \Pr_{g,h,R} [R^{\mathcal{D}^{g,h}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. Use the same three-way coupling and the same bad event defined along the table-only $\mathcal{D}$ execution in the proof of Lemma 77. That proof shows that, except when a recorded message pair is not γ -good or when some adaptively generated S_i contains an unqueried preimage of st_i , all three executions give the same answer to every query. The former event has probability at most $\text{poly}(\lambda)2^{-\tau/2}$, and Theorem 39 bounds the latter by $\text{poly}(\lambda)2^{-\tau/2+1}$. Hence the $\mathcal{C}$ and $\mathcal{D}$ executions differ only with negligible probability. $\square$

Lemma 79. *For every oracle machine R making polynomially many queries to its oracle and for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R} [R^{\mathcal{D}^{g,h}}(1^\lambda) \rightarrow 1] - \Pr_{R, \text{Sim}} [R^{\text{Sim}}(1^\lambda) \rightarrow 1] \right| = 0.$$

Proof. The adversary $\mathcal{D}$ uses the random functions g and h only through values that are explicitly queried and then recorded in the tables Q_0 and Q_1 . Specifically, $\mathcal{D}_0$ uses g only to answer first-stage queries on public keys, and $\mathcal{D}_1$ uses h only to answer state-generation queries on auxiliary-information/ciphertext inputs. The component $\mathcal{D}_2$ does not evaluate g or h on any new input; it only reads the tables Q_0 and Q_1 .

Therefore, sampling the full random functions (g, h) at the beginning and answering queries using these functions induces exactly the same distribution as lazy sampling their values on demand and storing the answers in the corresponding tables. But lazy sampling g and h on demand is precisely the behavior of Sim_0 and Sim_1 , and the third component of Sim is identical to $\mathcal{D}_2$ under the same table contents. Hence the oracle transcript generated by $\mathcal{D}^{g,h}$, for uniformly random (g, h) , is identically distributed to the oracle transcript generated by Sim . $\square$

Combining Lemma 76, Lemma 77, Lemma 78, and Lemma 79 by the triangle inequality proves Lemma 75. $\square$

By Lemma 74, the inefficient adversary $\mathcal{A}^{g,h}$ wins the incompressible PKE security game with probability negligibly close to 1, except with negligible probability over the choice of (g, h) . Equivalently, except with negligible probability over (g, h) , the fixed attacker $\mathcal{A}^{g,h}$ has normalized incompressible-PKE advantage at least $1/2$. By Lemma 75, no polynomial-query oracle machine can distinguish oracle access to $\mathcal{A}^{g,h}$, for random (g, h) , from oracle access to the efficient stateful simulator Sim , except with negligible advantage.

Thus, the family $\{\mathcal{A}^{g,h}\}_{g \in G, h \in H}$ together with the simulator Sim constitutes an average-case simulatable attack against the incompressible security of IncPKE . By Theorem 36, this rules out any black-box reduction from a secure cryptographic game to the incompressible security of IncPKE in the parameter regime of the theorem. This completes the proof of Theorem 73. $\square$

11.2 Impossibility Result for Leakage-Resilient Incompressible SKE

We show that leakage-resilient incompressible SKE cannot simultaneously tolerate too much leakage and have too small a ciphertext. The relevant quantity is the residual uncertainty in the secret key after leakage, namely $\ell_{\text{sk}} - \ell_{\text{leak}}$, together with the ciphertext-over-state overhead $\ell_{\text{ct}} - \ell_{\text{st}}$. Our lower bound shows that, if the sum of these two quantities is smaller than the message length by a superlogarithmic margin, then the leakage-resilient incompressible security of the scheme cannot be based on any standard assumption through a black-box reduction.

Theorem 80. *Let $\text{IncSKE} = (\text{IncSKE.Setup}, \text{IncSKE.Enc}, \text{IncSKE.Dec})$ be an SKE scheme with deterministic decryption and almost perfect correctness. Fix any parameter regime $\ell_{\text{msg}} = \ell_{\text{msg}}(\lambda)$ and $\ell_{\text{st}} = \ell_{\text{st}}(\lambda)$ with $\ell_{\text{msg}} = \omega(\log \lambda)$. Let $\ell_{\text{ct}} = \ell_{\text{ct}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, $\ell_{\text{sk}} = \ell_{\text{sk}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$, and $\ell_{\text{leak}} = \ell_{\text{leak}}(\lambda, \ell_{\text{msg}}, \ell_{\text{st}})$ denote the ciphertext length, secret-key length, and leakage length in this regime, where $0 \leq \ell_{\text{leak}} \leq \ell_{\text{sk}}$.*

Suppose the leakage class contains an efficiently computable leakage function $f : \text{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_{\text{leak}}}$ such that every leakage fiber has size at most $2^{\ell_{\text{sk}} - \ell_{\text{leak}}}$.⁶

Suppose there exists a function $\eta = \eta(\lambda) = \omega(\log \lambda)$ such that, for all sufficiently large λ , we have

$$(\ell_{\text{sk}} - \ell_{\text{leak}}) + \ell_{\text{ct}} - \ell_{\text{st}} \leq \ell_{\text{msg}} - \eta.$$

Then there is no secure cryptographic game $\mathcal{G}$ such that there is a black-box reduction deriving the leakage-resilient incompressible security of IncSKE in this parameter regime from the security of $\mathcal{G}$.

Thus, a necessary, but weaker, asymptotic consequence for any scheme that avoids this black-box barrier is

$$(\ell_{\text{sk}} - \ell_{\text{leak}}) + \ell_{\text{ct}} - \ell_{\text{st}} \geq \ell_{\text{msg}} - o(\ell_{\text{msg}})$$

⁶For example, f may be the truncation function that outputs the first ℓ_{leak} bits of the secret-key encoding.

in the relevant parameter regime. Thus, if the ciphertext has length $\ell_{\text{ct}} = \ell_{\text{msg}} + o(\ell_{\text{msg}})$ and the compressed state has length $\ell_{\text{st}} = \ell_{\text{msg}} - o(\ell_{\text{msg}})$, then $\ell_{\text{ct}} - \ell_{\text{st}} = o(\ell_{\text{msg}})$, and avoiding the barrier requires

$$\ell_{\text{sk}} - \ell_{\text{leak}} \geq \ell_{\text{msg}} - o(\ell_{\text{msg}}).$$

The theorem itself excludes precisely a superlogarithmic deficit in the sum of residual key length and ciphertext-over-state overhead. In particular, the displayed necessary condition does not by itself exclude residual key length equal to $\ell_{\text{msg}} - o(\ell_{\text{msg}})$.

Before giving the proof, we make a minor notational refinement of the leakage-resilient incompressible SKE security experiment. We write the adversary as $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, where $\mathcal{A}_0$, on input $(1^\lambda, \text{sk})$, outputs the leakage z , $\mathcal{A}_1$, on input $(1^\lambda, z)$, outputs the challenge messages and auxiliary information, $\mathcal{A}_2$, on input $(\text{ct}^*, \text{aux})$, outputs the compressed state st , and $\mathcal{A}_3$, on input $(\text{sk}, \text{aux}, \text{st})$, outputs the final guess. This is only a syntactic refinement of the usual experiment: the first component describes leakage generation, the second component performs message selection using the leakage, the third component compresses the challenge ciphertext, and the fourth component performs final recovery given the secret key and the compressed state.

Proof. If $\ell_{\text{st}} \geq \ell_{\text{ct}}$, there is already a direct efficient attack: after making the permitted leakage query and choosing two distinct messages, the adversary stores the entire challenge ciphertext (padding its state if necessary), and after receiving the secret key it decrypts and identifies the challenge message. Almost perfect correctness gives this adversary overwhelming success probability. A black-box reduction from a secure game would turn this efficient attacker into an efficient attacker against that game, contradicting its security. We may therefore restrict the remainder of the proof to $\ell_{\text{st}} < \ell_{\text{ct}}$.

It suffices to construct an average-case simulatable attack, in the sense of Definition 35, against the leakage-resilient incompressible security of IncSKE in the parameter regime specified in the theorem.

Fix the parameter regime from the theorem and write

$$L = \ell_{\text{sk}} - \ell_{\text{leak}} \quad \text{and} \quad \Delta = \ell_{\text{ct}} - \ell_{\text{st}}.$$

By assumption, there exists $\eta = \omega(\log \lambda)$ such that

$$L + \Delta \leq \ell_{\text{msg}} - \eta.$$

Define

$$\tau = \ell_{\text{msg}} - L - \Delta.$$

Then $\tau \geq \eta = \omega(\log \lambda)$. Finally, define

$$\gamma = \frac{\Delta + \tau/2}{\ell_{\text{msg}}}.$$

For every sufficiently large λ , we have $0 < \gamma < 1$. Indeed, $\Delta \geq 0$ and $\tau > 0$ imply $\gamma > 0$, while $\Delta + \tau = \ell_{\text{msg}} - L \leq \ell_{\text{msg}}$ implies $\gamma \ell_{\text{msg}} = \Delta + \tau/2 < \ell_{\text{msg}}$. We will repeatedly use the identities

$$\gamma \ell_{\text{msg}} - \Delta = \tau/2$$

and

$$(1 - \gamma)\ell_{\text{msg}} - L = \tau/2.$$

We use the standard convention that the message space has size $2^{\ell_{\text{msg}}}$, the ciphertext space has size at most $2^{\ell_{\text{ct}}}$, and every leakage fiber of f has size at most 2^L . We also identify the ciphertext space with its canonical encoding whenever invoking the set-image bound from Theorem 39.

Recall that a simulatable attack consists of an inefficient attacker against the primitive and an efficient stateful simulator. We now define both.

Inefficient adversary. Let G be the set of all functions $g : \{0, 1\}^{\ell_{\text{leak}}} \rightarrow \mathcal{M}_{\ell_{\text{msg}}} \times \mathcal{M}_{\ell_{\text{msg}}}$, and let H be the set of all functions $h : \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_{\text{st}}}$. The leakage function $f : \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_{\text{leak}}}$ is fixed.

For every $g \in G$ and $h \in H$, define the inefficient adversary $\mathcal{A}^{f, g, h} = (\mathcal{A}_0^{f, g, h}, \mathcal{A}_1^{f, g, h}, \mathcal{A}_2^{f, g, h}, \mathcal{A}_3^{f, g, h})$ as follows.

1. $\mathcal{A}_0^{f, g, h}$: On input $(1^\lambda, \text{sk})$, output $z = f(\text{sk})$.
2. $\mathcal{A}_1^{f, g, h}$: On input $(1^\lambda, z)$, compute $(m_0, m_1) = g(z)$, set $\text{aux} = (z, m_0, m_1)$, and output (m_0, m_1, aux) .
3. $\mathcal{A}_2^{f, g, h}$: On input $(\text{ct}^*, \text{aux})$, output $\text{st} = h(\text{ct}^*)$.
4. $\mathcal{A}_3^{f, g, h}$: On input $(\text{sk}, \text{aux}, \text{st})$, parse aux as (z, m_0, m_1) . If parsing fails, output $\perp$. If $f(\text{sk}) \neq z$, output $\perp$. If $g(z) \neq (m_0, m_1)$, output $\perp$. Otherwise, define

$$C^{h, \text{st}} = \{\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} : h(\text{ct}) = \text{st}\}$$

and

$$M^{h, \text{st}, \text{sk}} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C^{h, \text{st}} \text{ such that } m = \text{IncSKE.Dec}(\text{sk}, \text{ct})\}.$$

If $\{m_0, m_1\} \cap M^{h, \text{st}, \text{sk}} = \emptyset$ or $\{m_0, m_1\} \subseteq M^{h, \text{st}, \text{sk}}$, output $\perp$. Otherwise, if $m_0 \in M^{h, \text{st}, \text{sk}}$, output 0, and otherwise output 1.

We first show that this inefficient adversary wins the leakage-resilient incompressible SKE game with probability negligibly close to 1, on average over the choice of g and h .

Lemma 81. *There exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\Pr_{g, h, \text{Game}} [\mathcal{A}^{f, g, h}(1^\lambda) \text{ wins the leakage-resilient incompressible SKE game}] \geq 1 - \text{negl}(\lambda).$$

Proof. Let $\text{sk} \leftarrow \text{IncSKE.Setup}(1^\lambda, 1^{\ell_{\text{msg}}}, 1^{\ell_{\text{st}}})$ be the secret key generated in the experiment, and let $z = f(\text{sk})$ be the leakage output by $\mathcal{A}_0^{f, g, h}$. Let $(m_0, m_1) = g(z)$ be the challenge messages chosen by $\mathcal{A}_1^{f, g, h}$, set $\text{aux} = (z, m_0, m_1)$, let $b \leftarrow \{0, 1\}$ be the challenge bit, let $\text{ct}^* \leftarrow \text{IncSKE.Enc}(\text{sk}, m_b)$ be the challenge ciphertext, and let $\text{st} = h(\text{ct}^*)$ be the state output by $\mathcal{A}_2^{f, g, h}(\text{ct}^*, \text{aux})$.

By almost perfect correctness, except with negligible probability over the generation of sk and the randomness of encryption, we have $\text{IncSKE.Dec}(\text{sk}, \text{ct}^*) = m_b$. On this event, $m_b \in M^{h, \text{st}, \text{sk}}$, because ct^* itself belongs to $C^{h, \text{st}} = \{\text{ct} : h(\text{ct}) = \text{st}\}$. Hence the adversary fails to output the correct bit only if $m_{1-b} \in M^{h, \text{st}, \text{sk}}$. Indeed, if $m_b \in M^{h, \text{st}, \text{sk}}$ but $m_{1-b} \notin M^{h, \text{st}, \text{sk}}$, then exactly one of the two messages m_0, m_1 lies in $M^{h, \text{st}, \text{sk}}$, and the adversary outputs the corresponding bit, which is the correct bit. We now bound the probability that the other message also lies in this set.

Condition on sk, z, b, m_b , the encryption randomness used to generate ct^* , and the resulting ciphertext ct^* . After this conditioning, the ciphertext ct^* is fixed. The remaining randomness in the event $m_{1-b} \in M^{h, h(\text{ct}^*), \text{sk}}$ is over the random function h and over the non-challenge message m_{1-b} . Since $g(z)$ is sampled as a pair of independent uniform messages and we have conditioned only on the coordinate m_b , the other coordinate m_{1-b} remains uniform over $\mathcal{M}_{\ell_{\text{msg}}}$. Moreover, m_{1-b} is independent of h . Therefore, for every fixed choice of h , the probability over m_{1-b} that m_{1-b} lands in the set $M^{h, h(\text{ct}^*), \text{sk}}$ is at most $|M^{h, h(\text{ct}^*), \text{sk}}|/2^{\ell_{\text{msg}}}$. Averaging this bound over the remaining randomness of h , we get

$$\Pr[m_{1-b} \in M^{h, h(\text{ct}^*), \text{sk}}] \leq \frac{\mathbb{E}_h[|M^{h, h(\text{ct}^*), \text{sk}}|]}{2^{\ell_{\text{msg}}}}.$$

Moreover, for every fixed h , every message in $M^{h, h(\text{ct}^*), \text{sk}}$ is obtained by decrypting some ciphertext in $h^{-1}(h(\text{ct}^*))$. Therefore,

$$|M^{h, h(\text{ct}^*), \text{sk}}| \leq |h^{-1}(h(\text{ct}^*))|.$$

For fixed ct^* and uniformly random $h : \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} \rightarrow \{0, 1\}^{\ell_{\text{st}}}$, the ciphertext ct^* always belongs to $h^{-1}(h(\text{ct}^*))$. Every other ciphertext $\text{ct} \neq \text{ct}^*$ belongs to the same preimage exactly when $h(\text{ct}) = h(\text{ct}^*)$, which happens with probability $2^{-\ell_{\text{st}}}$. Therefore, by linearity of expectation,

$$\mathbb{E}_h[|h^{-1}(h(\text{ct}^*))|] = 1 + \frac{|\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| - 1}{2^{\ell_{\text{st}}}} \leq 1 + 2^{\ell_{\text{ct}} - \ell_{\text{st}}} = 1 + 2^\Delta \leq 2^{\Delta+1}.$$

Combining the previous inequalities, we obtain

$$\Pr[m_{1-b} \in M^{h, h(\text{ct}^*), \text{sk}}] \leq \frac{2^{\Delta+1}}{2^{\ell_{\text{msg}}}} = 2^{-(\ell_{\text{msg}} - \Delta) + 1}.$$

Since $\ell_{\text{msg}} - \Delta = L + \tau \geq \tau$, this probability is at most $2^{-\tau+1}$, which is negligible because $\tau = \omega(\log \lambda)$. Combining this with the almost perfect correctness error gives the lemma. $\square$

For fixed (g, h) , let $W_{g,h}(\lambda)$ denote the winning probability of $\mathcal{A}^{f,g,h}$ in the leakage-resilient incompressible SKE game, where the probability is over the randomness of the game. By Lemma 81, $\mathbb{E}_{g,h}[1 - W_{g,h}(\lambda)] \leq \text{negl}(\lambda)$. Hence, by Markov's inequality, $\Pr_{g,h}[W_{g,h}(\lambda) < 3/4] \leq 4\text{negl}(\lambda)$. Therefore, except with negligible probability over the choice of (g, h) , the fixed attacker $\mathcal{A}^{f,g,h}$ wins the leakage-resilient incompressible SKE game with probability at least $3/4$. Since the leakage-resilient incompressible SKE advantage is the normalized prediction advantage $\text{Adv}_{\text{IncSKE}}^{\mathcal{A}^{f,g,h}}(\lambda) = |2W_{g,h}(\lambda) - 1|$, we obtain $\text{Adv}_{\text{IncSKE}}^{\mathcal{A}^{f,g,h}}(\lambda) \geq 1/2$ except with negligible probability over (g, h) .

Simulator. We now define the efficient stateful simulator $\text{Sim} = (\text{Sim}_0, \text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$. The simulator maintains two tables Q_1 and Q_2 , initially empty. The table Q_1 lazily samples the random function g , and the table Q_2 lazily samples the random function h .

1. Sim_0 : On input $(1^\lambda, \text{sk})$, return $z = f(\text{sk})$.
2. Sim_1 : On input $(1^\lambda, z)$, check whether Q_1 already contains an entry of the form $(z, (m_0, m_1))$. If so, return (m_0, m_1, aux) , where $\text{aux} = (z, m_0, m_1)$. Otherwise, sample $(m_0, m_1) \leftarrow \mathcal{M}_{\ell_{\text{msg}}} \times \mathcal{M}_{\ell_{\text{msg}}}$, store $(z, (m_0, m_1))$ in Q_1 , set $\text{aux} = (z, m_0, m_1)$, and return (m_0, m_1, aux) .
3. Sim_2 : On input $(\text{ct}^*, \text{aux})$, check whether Q_2 already contains an entry of the form (ct^*, st) . If so, return st . Otherwise, sample $\text{st} \leftarrow \{0, 1\}^{\ell_{\text{st}}}$, store (ct^*, st) in Q_2 , and return st .
4. Sim_3 : On input $(\text{sk}, \text{aux}, \text{st})$, parse aux as (z, m_0, m_1) . If parsing fails, return $\perp$. First check whether $f(\text{sk}) = z$ and whether Q_1 contains $(z, (m_0, m_1))$. If either check fails, return $\perp$. Otherwise, define

$$C_{\text{Sim}} = \{\text{ct} : (\text{ct}, \text{st}) \in Q_2\}$$

and

$$M_{\text{Sim}} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C_{\text{Sim}} \text{ such that } m = \text{IncSKE.Dec}(\text{sk}, \text{ct})\}.$$

If $\{m_0, m_1\} \cap M_{\text{Sim}} = \emptyset$ or $\{m_0, m_1\} \subseteq M_{\text{Sim}}$, return $\perp$. Otherwise, if $m_0 \in M_{\text{Sim}}$, return 0, and otherwise return 1.

The simulator is efficient because it stores and processes only the polynomially many queries made by the oracle machine interacting with it, and because the leakage function f is efficiently computable. We next show that no polynomial-query oracle machine can distinguish oracle access to $\mathcal{A}^{f,g,h}$, for random (g, h) , from oracle access to Sim .

Lemma 82. *For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R}[R^{\mathcal{A}^{f,g,h}}(1^\lambda) \rightarrow 1] - \Pr_{R,\text{Sim}}[R^{\text{Sim}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. We introduce a sequence of hybrid adversaries between $\mathcal{A}$ and Sim . First, we record the simple combinatorial facts needed in the hybrid argument.

For a secret key sk , say that a message $m \in \mathcal{M}_{\ell_{\text{msg}}}$ is γ -bad with respect to sk if

$$|\{\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} : \text{IncSKE.Dec}(\text{sk}, \text{ct}) = m\}| > |\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \cdot 2^{-\gamma \ell_{\text{msg}}}.$$

Otherwise, say that m is γ -good with respect to sk .

Claim 3. For every $\text{sk} \in \mathcal{SK}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}$, we have

$$\Pr_{m \leftarrow \mathcal{M}_{\ell_{\text{msg}}}} [m \text{ is } \gamma\text{-bad with respect to } \text{sk}] \leq 2^{-(1-\gamma)\ell_{\text{msg}}}.$$

Proof. Fix any sk . Let

$$\mathfrak{M}_{\text{sk}} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : m \text{ is } \gamma\text{-bad with respect to } \text{sk}\}.$$

For every $m \in \mathfrak{M}_{\text{sk}}$, define

$$C_m = \{\text{ct} \in \mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}} : \text{IncSKE.Dec}(\text{sk}, \text{ct}) = m\}.$$

By definition of γ -badness, $|C_m| > |\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \cdot 2^{-\gamma \ell_{\text{msg}}}$ for every $m \in \mathfrak{M}_{\text{sk}}$. Since decryption is deterministic, the sets $\{C_m\}_{m \in \mathfrak{M}_{\text{sk}}}$ are pairwise disjoint. Hence

$$|\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \geq \sum_{m \in \mathfrak{M}_{\text{sk}}} |C_m| > |\mathfrak{M}_{\text{sk}}| \cdot |\mathcal{C}_{\lambda, \ell_{\text{msg}}, \ell_{\text{st}}}| \cdot 2^{-\gamma \ell_{\text{msg}}}.$$

Therefore $|\mathfrak{M}_{\text{sk}}| < 2^{\gamma \ell_{\text{msg}}}$, and so

$$\Pr_{m \leftarrow \mathcal{M}_{\ell_{\text{msg}}}} [m \text{ is } \gamma\text{-bad with respect to } \text{sk}] \leq \frac{2^{\gamma \ell_{\text{msg}}}}{2^{\ell_{\text{msg}}}} = 2^{-(1-\gamma)\ell_{\text{msg}}}.$$

□

Claim 4. For every $z \in \{0, 1\}^{\ell_{\text{eak}}}$ and every $i \in 0, 1$,

$$\Pr_{g_i} [g_i(z) \text{ is } \gamma\text{-bad with respect to some } \text{sk} \in f^{-1}(z)] \leq 2^{-\tau/2}.$$

Consequently,

$$\Pr_g [\text{at least one of } g_0(z), g_1(z) \text{ is } \gamma\text{-bad with respect to some } \text{sk} \in f^{-1}(z)] \leq 2^{-\tau/2+1}.$$

Proof. Fix $z \in \{0, 1\}^{\ell_{\text{eak}}}$ and $i \in 0, 1$. By the fiber-size assumption, $|f^{-1}(z)| \leq 2^L$. For every $\text{sk} \in f^{-1}(z)$, Claim 3 implies that the number of γ -bad messages with respect to sk is at most $2^{\gamma \ell_{\text{msg}}}$. Hence the total number of messages that are γ -bad with respect to some key in $f^{-1}(z)$ is at most $2^L \cdot 2^{\gamma \ell_{\text{msg}}}$. Since $g_i(z)$ is uniform over $\mathcal{M}_{\ell_{\text{msg}}}$, we get

$$\Pr_{g_i} [g_i(z) \text{ is } \gamma\text{-bad with respect to some } \text{sk} \in f^{-1}(z)] \leq \frac{2^L \cdot 2^{\gamma \ell_{\text{msg}}}}{2^{\ell_{\text{msg}}}} = 2^{-((1-\gamma)\ell_{\text{msg}} - L)}.$$

Using $(1-\gamma)\ell_{\text{msg}} - L = \tau/2$, this is exactly $2^{-\tau/2}$. The second statement follows by a union bound over $i \in 0, 1$. □

We now present the hybrid adversaries.

Adversary $\mathcal{B}$. Define $\mathcal{B}^{f,g,h} = (\mathcal{B}_0^{f,g,h}, \mathcal{B}_1^{f,g,h}, \mathcal{B}_2^{f,g,h}, \mathcal{B}_3^{f,g,h})$ as follows. The adversary maintains a table Q_1 , initially empty.

1. $\mathcal{B}_0^{f,g,h}$: On input $(1^\lambda, \text{sk})$, output $\mathcal{A}_0^{f,g,h}(1^\lambda, \text{sk})$.

2. $\mathcal{B}_1^{f,g,h}$: On input $(1^\lambda, z)$, compute $(m_0, m_1, \text{aux}) \leftarrow \mathcal{A}_1^{f,g,h}(1^\lambda, z)$, store $(z, (m_0, m_1))$ in Q_1 , and return (m_0, m_1, aux) .
3. $\mathcal{B}_2^{f,g,h}$: On input $(\text{ct}^*, \text{aux})$, return $\mathcal{A}_2^{f,g,h}(\text{ct}^*, \text{aux})$.
4. $\mathcal{B}_3^{f,g,h}$: On input $(\text{sk}, \text{aux}, \text{st})$, parse aux as (z, m_0, m_1) . If parsing fails, return $\perp$. If $f(\text{sk}) \neq z$, return $\perp$. If Q_1 does not contain $(z, (m_0, m_1))$, return $\perp$. Otherwise, output $\mathcal{A}_3^{f,g,h}(\text{sk}, \text{aux}, \text{st})$.

Lemma 83. *For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R}[R^{\mathcal{A}^{f,g,h}}(1^\lambda) \rightarrow 1] - \Pr_{g,h,R}[R^{\mathcal{B}^{f,g,h}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. We couple the two experiments using the same (g, h) and the same randomness for R . The executions can differ only if R makes a query $(\text{sk}, \text{aux}, \text{st})$ to $\mathcal{A}_3/\mathcal{B}_3$ such that $\text{aux} = (z, m_0, m_1)$, the value z was never queried to $\mathcal{A}_1/\mathcal{B}_1$, but $\mathcal{A}_3^{f,g,h}(\text{sk}, \text{aux}, \text{st}) \neq \perp$.

Analyze these guesses along the unconditioned $\mathcal{B}$ execution. Its final component never inspects $g(z)$ at an unrecorded leakage value. Thus, conditioned on any prefix of this execution, such a value remains uniform over $\mathcal{M}_{\ell_{\text{msg}}} \times \mathcal{M}_{\ell_{\text{msg}}}$, independently of the query formed next. The probability that its guessed (m_0, m_1) equals $g(z)$ is at most $2^{-2\ell_{\text{msg}}}$. A first disagreement of the coupled executions requires one of these guesses to succeed, since the checks $f(\text{sk}) = z$ and $g(z) = (m_0, m_1)$ of $\mathcal{A}_3$ must pass. A union bound over the polynomially many guesses along $\mathcal{B}$ proves the claim, since $\ell_{\text{msg}} = \omega(\log \lambda)$. $\square$

Adversary $\mathcal{C}$. Define $\mathcal{C}^{f,g,h} = (\mathcal{C}_0^{f,g,h}, \mathcal{C}_1^{f,g,h}, \mathcal{C}_2^{f,g,h}, \mathcal{C}_3^{f,g,h})$ as follows. The adversary maintains two tables Q_1 and Q_2 , initially empty.

1. $\mathcal{C}_0^{f,g,h}$: On input $(1^\lambda, \text{sk})$, output $\mathcal{A}_0^{f,g,h}(1^\lambda, \text{sk})$.
2. $\mathcal{C}_1^{f,g,h}$: On input $(1^\lambda, z)$, compute $(m_0, m_1, \text{aux}) \leftarrow \mathcal{A}_1^{f,g,h}(1^\lambda, z)$, store $(z, (m_0, m_1))$ in Q_1 , and return (m_0, m_1, aux) .
3. $\mathcal{C}_2^{f,g,h}$: On input $(\text{ct}^*, \text{aux})$, compute $\text{st} \leftarrow \mathcal{A}_2^{f,g,h}(\text{ct}^*, \text{aux})$, store (ct^*, st) in Q_2 , and return st .
4. $\mathcal{C}_3^{f,g,h}$: On input $(\text{sk}, \text{aux}, \text{st})$, parse aux as (z, m_0, m_1) . If parsing fails, return $\perp$. If $f(\text{sk}) \neq z$, return $\perp$. If Q_1 does not contain $(z, (m_0, m_1))$, return $\perp$. Otherwise, define

$$C_C = \{\text{ct} : (\text{ct}, \text{st}) \in Q_2\}$$

and

$$M_C = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C_C \text{ such that } m = \text{IncSKE.Dec}(\text{sk}, \text{ct})\}.$$

If $\{m_0, m_1\} \cap M_C = \emptyset$, return $\perp$. Otherwise, output $\mathcal{A}_3^{f,g,h}(\text{sk}, \text{aux}, \text{st})$.

Lemma 84. *For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R}[R^{\mathcal{B}^{f,g,h}}(1^\lambda) \rightarrow 1] - \Pr_{g,h,R}[R^{\mathcal{C}^{f,g,h}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. We analyze the coupling from the table-only execution $R^{\mathcal{D}^{f,g,h}}$, where $\mathcal{D}$ is defined below, and sample g and h lazily in the tables Q_1 and Q_2 . This execution reveals values of h only at ciphertexts queried to $\mathcal{D}_2$; its final component never inspects an unrecorded value of h .

Let q be a polynomial upper bound on the number of oracle queries. Each new leakage value queried to $\mathcal{D}_1$ receives an independent uniform message pair, conditional on the preceding

execution. By Claim 4 and a union bound, the probability that any recorded pair fails to be γ -good for every key in its leakage fiber is at most $q \cdot 2^{-\tau/2+1}$. We do not condition on the complement of this event.

For the i -th final-stage query $(\text{sk}_i, \text{aux}_i, \text{st}_i)$ in the table-only execution whose initial checks pass and whose recorded pair is γ -good for its leakage fiber, parse $\text{aux}_i = (z_i, m_0^{(i)}, m_1^{(i)})$ and set

$$T_{i,b} = \{\text{ct} : \text{IncSKE.Dec}(\text{sk}_i, \text{ct}) = m_b^{(i)}\} \quad (b \in \{0, 1\}).$$

Since $f(\text{sk}_i) = z_i$, goodness gives $|T_{i,b}| \leq 2^{\ell_{\text{ct}} - \gamma \ell_{\text{msg}}}$. Let $I_i \subseteq \{0, 1\}$ contain exactly those indices whose messages are obtained by decrypting ciphertexts already recorded in Q_2 with state st_i , and define

$$S_i = \begin{cases} T_{i,0} \cup T_{i,1}, & I_i = \emptyset, \\ T_{i,1-b}, & I_i = \{b\}, \\ \emptyset, & I_i = \{0, 1\}. \end{cases}$$

For a query that fails these checks, or an absent i -th query, set $S_i = \emptyset$ (with an arbitrary target for an absent query). Thus on every execution $|S_i| \leq 2 \cdot 2^{\ell_{\text{ct}} - \gamma \ell_{\text{msg}}}$, and no previously queried point in S_i has h -image st_i .

Couple the $\mathcal{B}$, $\mathcal{C}$, and table-only $\mathcal{D}$ executions until their first disagreement. All prior answers, and hence the next query, agree. Unless a recorded pair is bad, exactly as in the public-key proof, a case analysis on I_i shows that a disagreement implies $\text{st}_i \in h(S_i)$: when no message is recorded an unrecorded ciphertext must supply the unique full-fiber message, and when exactly one is recorded an unrecorded ciphertext must supply the other message. When both are recorded, every component returns $\perp$.

The sets S_i and targets st_i are generated along the table-only execution, which accesses h only through point queries. For each fixed i , stop that execution immediately after (S_i, st_i) is formed (or when the execution ends). Testing goodness for the leakage fiber and forming the set require only finite enumeration and no additional h -queries; the set-image theorem allows this computation to be unbounded. No future event is conditioned on, so the unqueried h -values remain uniform conditional on this prefix. In this truncated execution no queried point in S_i has image st_i , so Theorem 39, applied to this prefix with one output pair, bounds the probability of the i -th event. A union bound over the polynomially many possible queries bounds the probability of any set-image event by

$$\text{poly}(\lambda) \cdot \frac{2 \cdot 2^{\ell_{\text{ct}} - \gamma \ell_{\text{msg}}}}{2^{\ell_{\text{st}}}} = \text{poly}(\lambda) \cdot 2^{-\tau/2+1}.$$

Together with the separately bounded bad-message event, this is negligible. In particular, the coupled $\mathcal{B}$ and $\mathcal{C}$ executions differ only with negligible probability. $\square$

Adversary $\mathcal{D}$. Define $\mathcal{D}^{f,g,h} = (\mathcal{D}_0^{f,g,h}, \mathcal{D}_1^{f,g,h}, \mathcal{D}_2^{f,g,h}, \mathcal{D}_3^{f,g,h})$. The adversary $\mathcal{D}$ is identical to $\mathcal{C}$ in its first three components, but its fourth component answers only using the information recorded in Q_1 and Q_2 .

1. $\mathcal{D}_0^{f,g,h}$: On input $(1^\lambda, \text{sk})$, output $\mathcal{A}_0^{f,g,h}(1^\lambda, \text{sk})$.
2. $\mathcal{D}_1^{f,g,h}$: On input $(1^\lambda, z)$, compute $(m_0, m_1, \text{aux}) \leftarrow \mathcal{A}_1^{f,g,h}(1^\lambda, z)$, store $(z, (m_0, m_1))$ in Q_1 , and return (m_0, m_1, aux) .
3. $\mathcal{D}_2^{f,g,h}$: On input $(\text{ct}^*, \text{aux})$, compute $\text{st} \leftarrow \mathcal{A}_2^{f,g,h}(\text{ct}^*, \text{aux})$, store (ct^*, st) in Q_2 , and return st .
4. $\mathcal{D}_3^{f,g,h}$: On input $(\text{sk}, \text{aux}, \text{st})$, parse aux as (z, m_0, m_1) . If parsing fails, return $\perp$. If $f(\text{sk}) \neq z$, return $\perp$. If Q_1 does not contain $(z, (m_0, m_1))$, return $\perp$. Otherwise, define

$$C_{\mathcal{D}} = \{\text{ct} : (\text{ct}, \text{st}) \in Q_2\}$$

and

$$M_{\mathcal{D}} = \{m \in \mathcal{M}_{\ell_{\text{msg}}} : \exists \text{ct} \in C_{\mathcal{D}} \text{ such that } m = \text{IncSKE.Dec}(\text{sk}, \text{ct})\}.$$

If $\{m_0, m_1\} \cap M_{\mathcal{D}} = \emptyset$ or $\{m_0, m_1\} \subseteq M_{\mathcal{D}}$, return $\perp$. Otherwise, if $m_0 \in M_{\mathcal{D}}$, return 0, and otherwise return 1.

Lemma 85. *For every oracle machine R making polynomially many queries to its oracle, there exists a negligible function negl such that, for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R} [R^{\mathcal{C}^{f,g,h}}(1^\lambda) \rightarrow 1] - \Pr_{g,h,R} [R^{\mathcal{D}^{f,g,h}}(1^\lambda) \rightarrow 1] \right| \leq \text{negl}(\lambda).$$

Proof. Use the same three-way coupling and the same bad event defined along the table-only $\mathcal{D}$ execution in the proof of Lemma 84. That proof shows that, except when a recorded message pair is not γ -good for its leakage fiber or when some adaptively generated S_i contains an unqueried preimage of st_i , all three executions give the same answer to every query. The former event has probability at most $\text{poly}(\lambda)2^{-\tau/2+1}$, and Theorem 39 bounds the latter by the same quantity. Hence the $\mathcal{C}$ and $\mathcal{D}$ executions differ only with negligible probability. $\square$

Lemma 86. *For every oracle machine R making polynomially many queries to its oracle and for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{g,h,R} [R^{\mathcal{D}^{f,g,h}}(1^\lambda) \rightarrow 1] - \Pr_{R,\text{Sim}} [R^{\text{Sim}}(1^\lambda) \rightarrow 1] \right| = 0.$$

Proof. The adversary $\mathcal{D}$ uses the random functions g and h only through values that are explicitly queried and then recorded in the tables Q_1 and Q_2 . Specifically, $\mathcal{D}_1$ uses g only to answer message-selection queries on leakage values, and $\mathcal{D}_2$ uses h only to answer state-generation queries on ciphertexts. The component $\mathcal{D}_3$ does not evaluate g or h on any new input; it only reads the tables Q_1 and Q_2 . The component $\mathcal{D}_0$ computes only the fixed efficient leakage function f .

Therefore, sampling the full random functions (g, h) at the beginning and answering queries using these functions induces exactly the same distribution as lazy sampling their values on demand and storing the answers in the corresponding tables. But lazy sampling g and h on demand is precisely the behavior of Sim_1 and Sim_2 , while Sim_0 and $\mathcal{D}_0$ both compute f , and Sim_3 is identical to $\mathcal{D}_3$ under the same table contents. Hence the oracle transcript generated by $\mathcal{D}^{f,g,h}$, for uniformly random (g, h) , is identically distributed to the oracle transcript generated by Sim . $\square$

Combining Lemma 83, Lemma 84, Lemma 85, and Lemma 86 by the triangle inequality proves Lemma 82. $\square$

By Lemma 81, the inefficient adversary $\mathcal{A}^{f,g,h}$ wins the leakage-resilient incompressible SKE security game with probability negligibly close to 1, except with negligible probability over the choice of (g, h) . Equivalently, except with negligible probability over (g, h) , the fixed attacker $\mathcal{A}^{f,g,h}$ has normalized leakage-resilient incompressible-SKE advantage at least $1/2$. By Lemma 82, no polynomial-query oracle machine can distinguish oracle access to $\mathcal{A}^{f,g,h}$, for random (g, h) , from oracle access to the efficient stateful simulator Sim , except with negligible advantage.

Thus, the family $\{\mathcal{A}^{f,g,h}\}_{g \in G, h \in H}$ together with the simulator Sim constitutes an average-case simulatable attack against the leakage-resilient incompressible security of IncSKE . By Theorem 36, this rules out any black-box reduction from a secure cryptographic game to the leakage-resilient incompressible security of IncSKE in the parameter regime of the theorem. This completes the proof of Theorem 80. $\square$

12 Declaration on the Use of AI Tools

All research ideas and technical results presented in this work were developed by the authors in 2024, independently of generative AI tools and prior to their use in preparing this full version. Generative AI tools, including ChatGPT and Claude, were used only to assist with proofreading, language editing, improving the clarity and presentation of the manuscript, and identifying potential inconsistencies or points requiring further verification. No research ideas, definitions, constructions, theorems, proofs, or other technical contributions presented in this work originated from generative AI. All suggestions and checks arising from the use of these tools were independently reviewed and verified by the authors, who take full responsibility for the content of this manuscript.

References

- [1] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous Hardcore Bits and Cryptography against Memory Attacks. In: Reingold, O. (ed.) *Theory of Cryptography*. pp. 474–495. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_28 3, 4
- [2] Ball, M., Dachman-Soled, D., Guo, S., Malkin, T., Tan, L.Y.: Non-malleable codes for small-depth circuits. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS). pp. 826–837. IEEE (2018) 19, 20
- [3] Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: CCS '12 (2012) 19
- [4] Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. *Journal of Cryptology* 26(3), 513–558 (2013). <https://doi.org/10.1007/s00145-012-9136-3>, <https://eprint.iacr.org/2010/488> 6, 11, 24, 25, 26, 68
- [5] Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability: (or: Quadratic residuosity strikes back). In: *Advances in Cryptology—CRYPTO 2010: 30th Annual Cryptology Conference*, Santa Barbara, CA, USA, August 15–19, 2010. Proceedings 30. pp. 1–20. Springer (2010) 4
- [6] Brakerski, Z., Lombardi, A., Segev, G., Vaikuntanathan, V.: Anonymous ibe, leakage resilience and circular security from new assumptions. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. pp. 535–564. Springer (2018) 4
- [7] Branco, P., Döttling, N., Dujmović, J.: Rate-1 Incompressible Encryption from Standard Assumptions. In: Kiltz, E., Vaikuntanathan, V. (eds.) *Theory of Cryptography*. pp. 33–69. Lecture Notes in Computer Science, Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-22365-5_2 4, 5, 7, 15, 17, 22, 23
- [8] Canetti, R., Dodis, Y., Halevi, S., Kushilevitz, E., Sahai, A.: Exposure-resilient functions and all-or-nothing transforms. In: *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques* Bruges, Belgium, May 14–18, 2000 Proceedings 19. pp. 453–469. Springer (2000) 4
- [9] Cheraghchi, M., Didier, F., Shokrollahi, A.: Invertible extractors and wiretap protocols. *IEEE Transactions on Information Theory* 58(2), 1254–1274 (2011) 35
- [10] Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: *Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19–21, 2008*. Proceedings 5. pp. 427–444. Springer (2008) 19

- [11] Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: A black-box construction of non-malleable encryption from semantically secure encryption. *Journal of Cryptology* **31**, 172–201 (2018) [19](#)
- [12] Dachman-Soled, D., Gordon, S.D., Liu, F.H., O’Neill, A., Zhou, H.S.: Leakage resilience from program obfuscation. *Journal of Cryptology* **32**, 742–824 (2019) [4](#)
- [13] Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security*, Singapore, December 5–9, 2010. *Proceedings 16*. pp. 613–631. Springer (2010) [4](#)
- [14] Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. pp. 621–630 (2009) [4](#)
- [15] Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, May 2–6, 2004. *Proceedings 23*. pp. 523–540. Springer (2004) [12](#)
- [16] Dodis, Y., Sahai, A., Smith, A.: On Perfect and Adaptive Security in Exposure-Resilient Cryptography. In: Pfitzmann, B. (ed.) *Advances in Cryptology — EUROCRYPT 2001*. pp. 301–324. *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_19 [4](#)
- [17] Dziembowski, S.: On Forward-Secure Storage. In: Dwork, C. (ed.) *Advances in Cryptology - CRYPTO 2006*. pp. 251–270. *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11818175_15 [3](#), [6](#), [7](#), [10](#), [16](#)
- [18] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: *FOCS (2013)* [15](#)
- [19] Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing* **45**(3), 882–929 (2016) [15](#)
- [20] Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. pp. 99–108 (2011) [5](#)
- [21] Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: *Innovations in Computer Science, ICS 2010*. pp. 230–240. Tsinghua University Press (2010) [4](#)
- [22] Goyal, R., Koppula, V., Rajasree, M.S., Verma, A.: Incompressible functional encryption. *Cryptology ePrint Archive* (2024) [4](#), [5](#), [6](#)
- [23] Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: *Proceedings of Eurocrypt 2006*, volume 4004 of LNCS. pp. 339–358. Springer (2006) [26](#)
- [24] Guan, J., Wichs, D., Zhandry, M.: Incompressible Cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) *Advances in Cryptology – EUROCRYPT 2022*. pp. 700–730. *Lecture Notes in Computer Science*, Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-06944-4_24 [3](#), [4](#), [5](#), [6](#), [11](#), [15](#), [17](#), [33](#), [68](#), [72](#)
- [25] Guan, J., Wichs, D., Zhandry, M.: Somewhere Randomness Extraction and Security against Bounded-Storage Mass Surveillance (2023), <https://eprint.iacr.org/2023/409>, report Number: 409 [4](#)

- [26] Hajiabadi, M., Kapron, B.M., Srinivasan, V.: On generic constructions of circularly-secure, leakage-resilient public-key encryption schemes. In: Public-Key Cryptography–PKC 2016, pp. 129–158. Springer (2016) [4](#)
- [27] Jeronimo, F.G., Quintana, D., Srivastava, S., Tulsiani, M.: Unique decoding of explicit epsilon-balanced codes near the gilbert-varshamov bound. In: 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS). pp. 434–445. IEEE (2020) [20](#), [21](#)
- [28] Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. pp. 703–720. Springer Berlin Heidelberg, Berlin, Heidelberg (2009) [24](#)
- [29] Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO’99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 388–397. Springer (1999) [3](#)
- [30] Moran, T., Wichs, D.: Incompressible Encodings. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. pp. 494–523. Lecture Notes in Computer Science, Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-56784-2_17, <https://eprint.iacr.org/2020/814> [6](#), [11](#), [17](#), [18](#), [68](#), [71](#)
- [31] Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 18–35. Springer (2009) [4](#), [45](#)
- [32] Rivest, R.L.: All-or-nothing encryption and the package transform. In: Fast Software Encryption: 4th International Workshop, FSE’97 Haifa, Israel, January 20–22 1997 Proceedings 4. pp. 210–218. Springer (1997) [3](#)
- [33] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC. pp. 475–484 (2014) [15](#)
- [34] Vadhan, S.P.: Pseudorandomness. Foundations and Trends® in Theoretical Computer Science **7**(1–3), 1–336 (2012). <https://doi.org/10.1561/04000000010>, <http://dx.doi.org/10.1561/04000000010> [13](#)
- [35] Wichs, D.: Barriers in cryptography with weak, correlated and leaky sources. In: Kleinberg, R.D. (ed.) Innovations in Theoretical Computer Science, ITCS ’13, Berkeley, CA, USA, January 9–12, 2013. pp. 111–126. ACM (2013). <https://doi.org/10.1145/2422436.2422451>, <https://doi.org/10.1145/2422436.2422451> [5](#), [6](#), [9](#), [26](#), [27](#), [28](#)
- [36] Yao, A.: How to generate and exchange secrets. In: FOCS (1986) [19](#)