Fully Leakage-Resilient Codes

Antonio Faonio[1] and Jesper Buus Nielsen[1]

Aarhus University

Abstract. Leakage resilient codes (LRCs) are probabilistic encoding schemes that guarantee message hiding even under some bounded leakage on the codeword. We introduce the notion of *fully* leakage resilient codes (FLRCs), where the adversary can leak some λ_0 bits from the encoding process, i.e., the message and the randomness involved during the encoding process. In addition the adversary can as usual leak from the codeword. We give a simulation-based definition requiring that the adversary's leakage from the encoding process and the codework can be simulated given just λ_0 bits of leakage from the message. For $\lambda_0 = 0$ our new simulation-based notion is equivalent to the usual game-based definition. A FLRC would be interesting in its own right and would be useful in building other leakage-resilient primitives in a composable manner. We give a fairly general impossibility result for FLRCs in the popular split-state model, where the codeword is broken into independent parts and where the leakage occurs independently on the parts. We show that if the leakage is allowed to be any poly-time function of the secret and if collision-resistant hash functions exist, then there is no FLRC for the split-state model. The result holds only when the message length can be linear in the security parameter. However, we can extend the impossibility result to FLRCs for constant-length messages under assumptions related to differing-input obfuscation. These results show that it is highly unlikely that we can build FLRCs for the split-state model when the leakage can be any poly-time function of the secret state. We then give two feasibility results for weaker models. First, we show that for NC^0-bounded leakage from the randomness and arbitrary poly-time leakage from the parts of the codeword the inner-product construction proposed by Daví *et al.* (SCN'10) and successively improved by Dziembowski and Faust (ASIACRYPT'11) is a FLRC for the split-state model. Second, we provide a compiler from any LRC to a FLRC in the common reference string model for any fixed leakage family of small cardinality. In particular, this compiler applies to the split-state model but also to many other models.

Keywords. leakage-resilient cryptography, impossibility, fully-leakage resilience, simulation-based definition, feasibility results

1 Introduction

Leakage-resilient codes (LRCs) (also known as leakage-resilient storages) allow to store safely a secret information in a physical memory that may leak some side-channel information. Since their introduction (see Davì *et al.* [DDV10]) they have found many applications either by their own or as building block for other leakage and tamper resilient primitives. To mention some, Dziembowski and Faust [DF11] proposed an efficient and continuous leakage-resilient identification scheme and a continuous leakage-resilient CCA2 cryptosystem, while Andrychowicz *et al.* [AMP15] proposed a practical leakage-resilient LPN-based version of the Lapin protocol (see Heyse *et al.* [HKL$^+$12]) both relying on LRCs based on the inner-product extractor. LRC found many applications also in the context of non-malleable codes (see Dziembowski *et al.* [DPW10]), which roughly speaking can be seen as their tamper-resilience counterpart. Faust *et al.* [FMVW14] showed a non-malleable code based on LRC, Aggarwal *et al.* [ADKO15] proposed a construction of leakage and tamper resilient code and Faust *et al.* [FMNV14] showed continuous non-malleable codes based on LRC [FMNV14] (see also Jafargholi and Wichs [JW15]).

The security requirement of LRC states that given two messages, arbitrarily but bounded length leakage on the encoding of them is indistinguishable. Ideally, a good LRC should be resilience to a leakage that can be much longer than the size of the message protected, however, to get such strong guarantee some restriction on the class of leakage allowed must be set. Intuitively, any scheme where the adversary can compute the decoding function as leakage cannot be secure. A way to fix this problem is to consider randomly chosen LRCs. As showed in [DDV10], and successively improved in [FMVW14,JW15], for

any fixed set of leakage functions, there exists a family of efficiently computable codes such that with high probability a code from this family is leakage resilient. From a cryptographic prospective, the results known in this direction can be interpreted as being in the "common reference string" model, where the leakage class is set and, then, the LRC is sampled.

Another way, more relevant for our paper, is to consider the split-state model [DP08,HL11] where the message is encoded in two (or more) codewords and the leakage happens adaptively but independently from each codewords, thus the decoding function cannot automatically be part of the allowed leakage, which opens the possibility of constructing a LRC.

It is easy to see that the encoding algorithm must be randomized, otherwise two fixed messages can be easily distinguished. However, the security of LRC does not give any guarantee when there might be leakage from the randomness used in the encoding process. In other words, while the encoded message can be stored in a leaky device the encoding must be executed in a completely leak-free environment. A stronger flavour of security where we allow some bounded leakage from also the encoding process is sometimes called *fully* leakage resilience.

1.1 Our Contributions

We generalize the notion of LRC, namely to the setting of fully leakage resilience. Roughly speaking, a fully leakage-resilient codes (FLRC) hide information about the secret message even when the adversary leaked information during the encoding process. Our contributions can be summarized as follow:

1. We provide a simulation-based definition for fully leakage-resilient codes. The definition postulates that for any adversary leaking λ_0 bit from the encoding process and λ_1 bits from the codewords there exists a simulator which provide a view that is indistinguishable. The simulator is only allowed to leak λ_0 bits of information on the underlying message. Since the adversary can always just leak λ_0 bits from the message, the simulator must of course also be allowed to leak λ_0 bits from the message, or the notion would be trivially impossible to instantiate. Our definition is therefore in some sense the minimal one suitable for the fully leakage resilience setting. As a sanity check, our new notion is implied by the indistinguishability-based definition of [DDV10] for $\lambda_0 = 0$.
2. We show that there does not exist an efficient coding scheme in the split-state model that is a fully leakage resilient code if the leakage function is allowed to be any poly-time function. Our result holds for coding schemes where the messages length can be linear in the security parameter and under the sole assumption that collision-resistant hash functions exist. We can generalise the impossibility result to the case of constant-length messages under the much stronger assumption that differing-input obfuscation exists (see [ABG$^+$13,BCP14]).
3. We provide two feasibility results for weaker models. First, we show that, if the leakage from the randomness is computable by bounded-depth constant fan-in circuits (i.e. NC^0-computable leakage), the inner-product extractor LRC of [DDV10] is fully leakage resilient. Secondly, we provide a compiler from any LRC to a fully leakage resilient code in the common reference string model for any fixed leakage family of small cardinality.

Simulation-based Security. Consider the naive fully leakage-resilient extension of the indistinguishability-based security definition of LRC. Roughly speaking, the adversary plays against a challenger and it can leak $\lambda_0 > 0$ bits from a random string $\omega \leftarrow\!\!\$\ \{0,1\}^*$, successively the adversary sends to the challenger two messages m_0, m_1, the challenger chooses a random bit b and encodes the message m_b using the randomness ω. After this, the adversary gets access to leakage from the codewords. We show an easy attack. The attacker can compute via leakage function on the randomness the encoding of both m_0 and m_1 and find a coordinate in which the two codewords differ, successively, by leaking from the codeword only one bit, it can check whether m_0 or m_1 has been encoded.

The problem is that the indistinguishability-based security definition for fully leakage resilient codes sketched above concentrates on preserving, in the presence of leakage on the randomness, the same security guarantees of the (standard) leakage resilient definition. However, the ability of leaking before and after the challenge generation, as showed for many other cryptographic primitive, gives to the adversary too much power.

Following the *leakage-tolerant* paradigm introduced by Bitansky *et al.* [BCH12], we instead consider a simulation-based notion of security. The definition postulates that for any adversary leaking λ_0 bits from the encoding process and λ_1 bits from the codewords there exists a simulator which provide a view that is indistinguishable without knowing the underling message. Namely, the adversary chooses one input message and forward it to the security game. After that the adversary can, sequentially, leak from the encoding process and from the codeword. The job of the simulator is to produce an indistinguishable view of the leakage oracles to the adversary. It is not hard to see that, without any help, the task is impossible. Since an adversary can in particular leak bits of the input message, if the input message is randomly chosen the simulator trivially cannot provide an indistinguishable view. The definition provides, therefore, oracle access on the message to the simulator. Specifically, the simulator can leak up to λ_0 bits from the message. The idea is that some information about the encoded message can unavoidably leaked from the encoding process, however the amount of information about the message, even after have seen the leakage on the codeword, does not exceed the bound on the leakage on the encoding process.

The impossibility results. We give an impossibility result for FLRCs in the split-state model. Recall that, in the split state model, the codeword is divided in two parts which are stored in two independent leaky devices. Each leakage query can be any poly-time functions of the data stored in one of the parts.

In our attack we leak from the encoding process a hash of each of the two parts of the codeword. The leakage function takes the message and the randomness, runs the encoding algorithm to compute the two parts L ans R (the left part and the right part) and leaks two hash values $h_l = h(L)$ and $h_r = h(R)$. Then we use succinct argument of knowledge systems to also leak an argument of knowledge of preimages L and R of h_l and h_r for which it holds that (L, R) decodes to m. Let λ_0 be the length of the two hashes and the succinct argument. After this leakage for the encoding process the adversary uses its access to leak from L to leak, in sequence, several succinct arguments of knowledge of L such that $h_l = h(L)$. Similarly the adversary uses its access to leak from R to leak, in sequence, several succinct arguments of knowledge of R such that $h_r = h(R)$. By setting $\lambda_1 \gg \lambda_0$ we can within the leakage bound λ_1 on L and R leak for instance $12\lambda_0$ succinct arguments of knowledge of L and R. If the code is secure there exists a simulator which can simulate the leakage of h_l and h_r and all the arguments given at most λ_0 bits of leakage on m. Since the arguments are accepting in the real world and the simulator is assumed to be good it follows that the simulated arguments are accepting too with probability close of 1. Since the simulator has access to only λ_0 bits of leakage on m it follows that for one of the $12\lambda_0$ simulated arguments produced by the simulator it uses the leakage oracle on m with probability at most $\frac{1}{4}$. This means that with probability $\frac{3}{4}$ the simulator is not even using the leakage oracle to simulate this argument, so if we remove the access to leakage from m the argument will still be acceptable with probability close to $\frac{3}{4}$. Hence if the argument systems has knowledge error just $\frac{1}{2}$ we can extract L from this argument with probability close to $\frac{1}{4}$. Similarly we can extract from one of the arguments of knowledge of R the value R with probability $\frac{1}{4}$. By collision resistance and the argument leaked from the encoding process it follows that (L, R) decodes to m. This means that we can extract from the simulator the message m with probability $\frac{1}{16}$ while using only λ_0 bits of leakage on m. If m is uniformly random and just $\lambda_0 + 5$ bits long, this is a contradiction, as there is 5 bits on min-entropy on m after leaking λ_0 bits on m and hence m cannot be guessed with probability better than 2^{-5}.

Similar proof techniques have been used already by Nielsen *et al.* [NVZ13] to prove connection between leakage resilience and adaptive security and recently by Ostrovsky *et al.* [OPV15] to prove

an impossibility result for certain flavours of leakage-resilient zero-knowledge proof systems. The way apply this type of argument here is novel. It is in particular a new idea to use many arguments of knowledge in sequence to sufficient restrict the simulators ability to leak from its leakage oracle in one of the proofs.

The definition of FLR makes sense only when the leakage parameter λ_0 is strictly smaller than the size of the message. Notice that the proposed attack needs to leak at least a collision resistant hash function of the codeword, which means that the length of the message needs to be super-logarithmic in the security parameter. Thus the technique cannot be used directly to give an impossibility result for constant-length message FLRC. We can overcome this problem relying on the concept of adaptive-secure Predictable Argument of Knowledge (PAoK) recently proposed by Faonio *et al.* [FNV15]. An adaptive-secure PAoK is an extremely succinct 2-message argument of knowledge where the prover can first see the challenge from the verifier and then decide the instance. This allows the attacker to implement the first check by just leaking a constant-length argument that the hashes of the two parts of the codeword are well formed (without actually leaking the hash values) and then, successively, leak the hash values from the codeword and check the validity of the argument. Adaptive-secure PAoK are showen to imply extractable witness encryption (see Boyle *et al.* [BCP14]) and therefore the "implausibility" result of Garg *et al.* [GGHW14] applies. We interpret our second impossibility result as a strong evidence that constant-length FLRC are hard to construct.

The feasibility results. The ability to leak collision resistant hash functions of the randomness is necessary for the impossibility result. A natural question follows. If we restrict the leakage class so that no collision resistance hash function can be computed as leakage on the randomness, can we find a coding scheme that is fully leakage resilient? We answer this question affirmatively. We consider the class NC^0 of constant-depth constant fan-in circuits and we show that the LRC based on the inner-product extractor (and more general LRCs where there is an NC^0 function that maps the randomness to the codeword) are fully leakage resilient. The intuition is that NC^0 leakage is not powerful enough to break all the "independence" between the two parts of the codeword. Technically, we are able to cast every leakage query on the randomness into two slightly bigger and independent leakage queries on the two parts of the codeword. Notice that collision resistance hash functions cannot be computed by NC^0 circuits. This is necessary. In fact, proving a similar result for a bigger complexity class automatically implies a lower bound on the complexity of computing either a collision resistant hash function or an argument of knowledge. Intuitively, this provides a strong evidence that is hard to construct FLRC even for bounded class of leakage.

A second path to avoid the impossibility results is to consider weaker models of security. We point out that the schemes proposed by [DDV10,FMVW14,JW15] in the common reference string model can be easily proved to be fully leakage resilient. Inspired by the above results we provide a compiler that maps any LRC to a FLRC in the common reference string model. In this model the adversary is not allowed to leak from the reference string during the encoding phase, while it still can leak from the randomness. In practice one can imagine a situation in which the reference string is randomly chosen and hardwired in the encoder using some leak-free components while the randomness can come from a source subject to bounded-size leakage. Unfortunately, to apply this technique we still need to put some limitation on the leakage class. However, the bound is on the cardinality of the leakage class and not on its complexity (in principle, the leakage class could contain collision resistance hash functions). The key idea is to apply a deterministic extractor on the randomness before using it inside the encoding machine. Technically, we use a result of Trevisan and Vadhan [TV00] which proves that for any fixed leakage class $\mathcal{F}$ a t-wise independent hash function (the parameter t depends on the cardinality of $\mathcal{F}$) is an extractor with high probability. The proof mostly follows the template given in [FMVW14].

1.2 Related Work

Cryptographic schemes are designed under the assumption that the adversary cannot learn any information about the secret key. However, side-channel attacks (see [Koc96,KJJ99,QS01]) have showed that this assumption does not always hold. These attacks have motivated the design of leakage-resilient cryptosystems which remain secure even against adversaries that may obtain partial information about the secret state. Starting from the groundbreaking result of Micali and Reyzin [MR04], successively either gradually stronger or different models have been consider (see for example [ADW09,DP08,FRR+10,NS09]). Fully leakage resilient schemes are known for signatures [BSW13,FNV14,MTVY11], zero-knowledge proof system [AGP14,GJS11,Pan14] and multi-party computation protocols [BDL14,BGK14]. Similar concepts of leakage resilient codes have been considered, Liu and Lysyanskaya [LL12] and successively Aggarwal *et al.* [ADKO15] constructed leakage and tamper resilient codes while Dodis *et al.* [DLWW11] constructed continual leakage resilient storage. Simulation-based definitions in the context of leakage-resilient cryptography were also adopted in the case of zero-knowledge proof (see [AGP14,GJS11,Pan14]) public-key encryption (see [HL11]) and signature schemes (see [NVZ14]). As mentioned already, our proof technique for the impossibility result is inspired by the works of Nielsen *et al.* [NVZ13] and Ostrovsky *et al.* [OPV15], however, part of the analysis diverges, and instead resembles an information theoretic argument already known in leakage-resilient cryptography (see for example [ADW09,FNV14,KV09]).

2 Preliminaries

2.1 Notation and Probability Preliminaries

We let $\mathbb{N}$ denote the naturals and $\mathbb{R}$ denote the reals. For $a, b \in \mathbb{R}$, we let $[a, b] = \{x \in \mathbb{R} \,:\, a \leq x \leq b\}$; for $a \in \mathbb{N}$ we let $[a] = \{1, 2, \ldots, a\}$. If x is a string, we denote its length by $|x|$; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the number of elements in $\mathcal{X}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow \mathcal{X}$. When $\mathcal{A}$ is an algorithm, we write $y \leftarrow \mathcal{A}(x)$ to denote a run of $\mathcal{A}$ on input x and output y; if $\mathcal{A}$ is randomized, then y is a random variable and $\mathcal{A}(x; r)$ denotes a run of $\mathcal{A}$ on input x and randomness r. An algorithm $\mathcal{A}$ is *probabilistic polynomial-time* (ppt) if $\mathcal{A}$ is allowed to use random choices and for any input $x \in \{0,1\}^*$ and randomness $r \in \{0,1\}^*$ the computation of $\mathcal{A}(x; r)$ terminates in at most $\mathsf{poly}(|x|)$ steps.

Let κ be a security parameter. A function negl is called *negligible* in κ (or simply negligible) if it vanishes faster than the inverse of any polynomial in κ. For a relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$, the language associated with $\mathcal{R}$ is $\mathcal{L}_{\mathcal{R}} = \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$.

For two ensembles $\mathcal{X} = \{X_\kappa\}_{\kappa\in\mathbb{N}}$, $\mathcal{Y} = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$, we write $\mathcal{X} \overset{c}{\approx}_\epsilon \mathcal{Y}$, meaning that every probabilistic polynomial-time distinguisher D has $\epsilon(\kappa)$ advantage in distinguishing $\mathcal{X}$ and $\mathcal{Y}$, i.e., $\frac{1}{2}|\Pr[D(\mathcal{X}_\kappa) = 1] - \Pr[D(\mathcal{Y}_\kappa) = 1]| \leq \epsilon(\kappa)$ for all sufficiently large values of κ.

We simply write $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$ when there exists a negligible function ϵ such that $\mathcal{X} \overset{c}{\approx}_\epsilon \mathcal{Y}$. Similarly, we write $\mathcal{X} \approx_\epsilon \mathcal{Y}$ (statistical indistinguishability), meaning that every unbounded distinguisher has $\epsilon(\kappa)$ advantage in distinguishing $\mathcal{X}$ and $\mathcal{Y}$.

Given two ensembles $\mathcal{X}$ and $\mathcal{Y}$ such that $\mathcal{X} \approx_\epsilon \mathcal{Y}$ the following holds:

$$\frac{1}{2}\sum_z \left|\Pr[X_\kappa = z] - \Pr[Y_\kappa = z]\right| \leqslant \epsilon(\kappa).$$

We recall the notion of (average) conditional min-entropy. We adopt the definition given in [ADW09], where the authors generalize the notion of conditional min-entropy to *interactive* predictors that participate in some randomized experiment $\mathbf{E}$. The conditional min-entropy of random variable X given any randomized experiment $\mathbf{E}$ is defined as follows:

$$\widetilde{\mathbb{H}}_\infty(X \mid \mathbf{E}) = \max_{\mathcal{B}} \left(-\log \mathbb{P}\left[\mathcal{B}()^{\mathbf{E}} = X\right]\right),$$

where the maximum is taken over all predictors without any requirement on efficiency. Note that w.l.o.g. the predictor $\mathcal{B}$ is deterministic, in fact, we can de-randomize $\mathcal{B}$ by hardwiring the random coins that maximize its outcome. Sometimes we write $\widetilde{\mathbb{H}}_\infty(X|Y)$ for a random variable Y, in this case we mean the average conditional min-entropy of X given the random experiment that gives Y as input to the predictor.

Given a string $X \in \{0,1\}^*$ and a value $\lambda \in \mathbb{N}$ let the oracle $\mathcal{O}^X_\lambda(\cdot)$ be *the leakage oracle* that accepts as input functions $f_1, f_2, \ldots$ defined as polynomial-sized circuits and outputs $f_1(X), f_2(X), \ldots$ under the restriction that $\sum_i |f_i(X)| \leqslant \lambda$.

We recall here a lemma of Alwen *et al.* (see [ADW09]) that we make use of.

Lemma 1. *For any random variable X and for any experiment $\mathbf{E}$ with oracle access to $\mathcal{O}^X_\lambda(\cdot)$, consider the experiment $\mathbf{E}'$ which is the same as $\mathbf{E}$ except that the predictor does not have oracle access to $\mathcal{O}^X_\lambda(\cdot)$, then $\widetilde{\mathbb{H}}_\infty(X \mid \mathbf{E}) \geqslant \widetilde{\mathbb{H}}_\infty(X \mid \mathbf{E}') - \lambda$.*

We recall a lemma from Bellare and Rompel [BR94].

Lemma 2. *Let $t \geqslant 4$ be an even integer. Suppose $X_1, \ldots, X_n$ are t-wise independent random variables taking values in $[0,1]$. Let $X := \sum_i X_i$ and define $\mu := \mathbb{E}[X]$ to be the expectation of the sum. Then, for any $A > 0$, $\Pr[|X - \mu| \geqslant A] \leqslant 8\left(\frac{t\mu + t^2}{A^2}\right)^{t/2}$.*

2.2 Cryptographic Primitives

Arguments of Knowledge. Our results are based on the existence of round-efficient interactive argument systems. We follow some of the notation of Wee [Wee05], the knowledge soundness definition is taken from [OPV15].

Definition 1 (Argument of knowledge). *An interactive protocol (P, V) is an argument of knowledge for a language $\mathcal{L}$ if there is a relation $\mathcal{R}$ such that $\mathcal{L} = \mathcal{L}_\mathcal{R} := \{x | \exists w : (x,w) \in \mathcal{R}\}$, and functions $\nu, s : \mathbb{N} \to [0,1]$ such that $1 - \nu(\kappa) > s(\kappa) + 1/\mathsf{poly}(\kappa)$ and the following conditions hold.*

- *(Efficiency): The length of all the exchanged messages is polynomially bounded, and both P and V are computable in probabilistic polynomial time;*
- *(Completeness): If $(x,w) \in \mathcal{R}$, then V accepts in $(P(w), V)(x)$ with probability at least $1 - \nu(|x|)$.*
- *(Knowledge Soundness): For every* ppt *prover strategy P^*, there exists an expected polynomial-time algorithm* K *(called the* knowledge extractor*) such that for every $x, z, r \in \{0,1\}^*$ if we denote by $p^*(x,z,r)$ the probability that V accepts in $(P(z;r), V)(x)$, then $p^*(x,z,r) > s(|x|)$ implies that*

$$\Pr[\mathsf{K}(P^*, x, z, r) \in \mathcal{R}(x)] \geqslant p^*(x,z,r) - s(|x|).$$

The value $\nu(\cdot)$ is called the *completeness error* and the value $s(\cdot)$ is called the *knowledge error*. We say (P, V) has perfect completeness if $\nu = 0$. The communication complexity of the argument system is the total length of all messages exchanged during an execution; the round complexity is the total number of exchanged messages. We write $\mathsf{AoK}_{\nu,s}(\rho(\kappa), \lambda(\kappa))$ to denote interactive argument on knowledge systems with completeness error ν, knowledge error s, round-complexity $\rho(\kappa)$ and communication complexity $\lambda(\kappa)$. Sometimes we also write $\lambda(\kappa) = \lambda_P(\kappa) + \lambda_V(\kappa)$ to differentiate between the communication complexity of the prover and of the verifier. The protocol is called *public-coin* when the verifier's moves consist merely of tossing coins and sending their outcomes to the prover. A public-coin argument system (P, V) is fully described by the tuple of ppt algorithms $(\mathsf{Prove}, \mathsf{Judge})$ where:

- V on input x samples uniformly random strings $y_1, \ldots, y_{\rho(\kappa)} \leftarrow\!\!\$\ \{0,1\}^\kappa$, P on inputs x, w samples uniformly random string $r_P \leftarrow\!\!\$\ \{0,1\}^\kappa$.
- For any $i \in [\rho(\kappa)]$, V sends the message y_i and P replies with the message $x_i := \mathsf{Prove}(x, w, y_1, \ldots, y_i; r_P)$.

- The verifier V executes $j := \mathsf{Judge}\big(x, y_1, \ldots, y_{\rho(\kappa)}, x_1, \ldots, x_{\rho(\kappa)}\big)$ and accepts if $j = 1$.

We say (P, V) is *succinct* if $\lambda(\kappa)$ is poly-logarithmic in the length of the witness and the statement being proven.

Remark 1. The knowledge soundness property holds for any ppt prover, however, in what follows, we execute the knowledge extractor machine on oracle machines. Abusing of notation, for a ppt oracle prover P and deterministic polynomial time oracle $\mathcal{O}$, when we say that we run the knowledge extractor on $P^{\mathcal{O}}, x, z, r$, i.e. $\mathsf{K}(P^{\mathcal{O}}, x, z, r)$, implicitly we are defining a new ppt machine $\tilde{P}$ which runs internally P and replies to the oracle queries q by running an instance of $\mathcal{O}(q)$ as subroutine.

Instantiations. Kilian [Kil92] constructs a 4-round public-coin succinct argument of knowledge for NP based on a probabilistically checkable proof (PCP) system for NP and a collision-resistant function ensemble. Gentry and Wichs [GW11] prove that non-interactive succinct arguments, so called SNARGs, cannot exist given a black-box reduction to any falsifiable assumption. In fact, the only constructions of SNARGs we know of are either based on the random oracle model of Bellare and Rogaway [BR93] (as shows Micali [Mic00] by applying the Fiat-Shamir transform [FS86] to Kilian's protocol) or under so-called "knowledge of exponent" assumptions [BCCT12] or under so-called "extractable collision-resistant hash functions" [BCC$^+$14].

We remark that for our results interactive arguments are sufficient; in particular our theorems can be based on the assumption that collision-resistant function ensembles exist.

Predictable Arguments of Knowledge. Recently, Faonio *et al.* [FNV15] put forward the notion of Predictable Arguments of Knowledge (PAoK). The primitive, strictly connected with the concept of Extractable Witness Encryption (see Ananth *et al.* [ABG$^+$13]), is a 2-message argument of knowledge system where the prover last message can be of constant size.

Specifically, we make use of an *adaptive-secure* PAoK. Such protocols are fully specified by a tuple of three ppt algorithms $\Pi = (\mathsf{Chall}, \mathsf{Resp}, \mathsf{TResp})$ as described below:

- V samples $(c, tp) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa)$ and sends c to P.
  (Notice that the generated tuple is independent of the instance.)
- P samples $a \leftarrow\!\!\$\ \mathsf{Resp}(1^\kappa, x, w, c)$ and sends a to V.
- V computes $b := \mathsf{TResp}(tp, x)$ and accepts if $a = b$, else rejects.

Roughly speaking, an adaptive-secure PAoK the malicious prover plays a game where it first see the challenge c sent from the verifier and then selects the instance x and a valid answer for the instance. The knowledge property states that for any adversary succeeding in the experiment with some non trivial probability, there exists a knowledge extractor that outputs valid witnesses for the instances produced by the malicious prover with roughly the same probability. We defer the security definition in Appendix B.

Collision Resistant Hash Functions. Let $(\mathsf{Gen}^{\mathrm{CRH}}, \mathsf{Eval}^{\mathrm{CRH}})$ be a tuple of ppt algorithms such that upon input 1^κ the algorithm Gen outputs a evaluation key h and upon inputs h and a string $x \in \{0,1\}^*$ the deterministic algorithm $\mathsf{Eval}^{\mathrm{CRH}}$ outputs a string $y \in \{0,1\}^{\ell_{\mathsf{CRH}}(\kappa)}$. We shorten the notation by writing $h(x)$ for $\mathsf{Eval}^{\mathrm{CRH}}(h, x)$.

Definition 2. *A tuple* $(\mathsf{Eval}^{CRH}, \mathsf{Gen}^{CRH})$ *is a collision-resistant hash function (family) with output length* $\ell_{\mathsf{CRH}}(\kappa)$ *if for any non-uniform polynomial time collision-finder adversary* $\mathcal{B}_{coll}$ *there exists a negligible function* negl *such that the following holds:*

$$\Pr_{h \leftarrow\!\!\$\ \mathsf{Eval}^{CRH}(1^\kappa)} \big[h(x_0) = h(x_1) \wedge x_0 \neq x_1,\ (x_0, x_1) := \mathcal{B}_{coll}(h)\big] < \mathsf{negl}(\kappa).$$

For simplicity we consider the model of non-uniform polynomial time adversaries, however, note that our results hold also if we consider the model ppt adversaries.

3 Definition

In this section we give the definition of fully leakage resilient code. The definition given is specialized for the 2-split-state model, we adopt this definition instead of a more general one for simplicity. The results given in Section 4 can be adapt to hold for the more general k-split-model (see Remark 2). LRCs of [DDV10,FMNV14,JW15] in common reference string model can be proved to have a weak form of fully-leakage resilience (see Section 5), in order to capture this we give a second weaker definition of security. The syntax given allows the scheme to depends on a common reference string, so to include the scheme of [LL12],

A (α, β)-split-coding scheme is a tuple $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of ppt algorithms with the following syntax:

- Gen on inputs 1^κ outputs a common reference string crs.
- Enc on inputs crs and a message $m \in \mathcal{M}_\kappa$ outputs a tuple $(L, R) \in \mathcal{C}_\kappa$;
- Dec is a deterministic algorithm that on inputs crs and a codeword $(L, R) \in \mathcal{C}_\kappa$ decodes to $m' \in \mathcal{M}_\kappa$.

Here $\mathcal{M}_\kappa = \{0,1\}^{\alpha(\kappa)}$, $\mathcal{C}_\kappa = \left(\{0,1\}^{\beta(\kappa)} \times \{0,1\}^{\beta(\kappa)}\right)$ and the randomness space of Enc is $\mathcal{R}_k = \{0,1\}^{p(\kappa)}$ for a fixed polynomial p.

A split-coding scheme is correct if for any κ and any $m \in \mathcal{M}_\kappa$ we have $\Pr_{\mathsf{crs},r_e}[\mathsf{Dec}(\mathsf{crs}, \mathsf{Enc}(\mathsf{crs}, m; r_e)) = m] = 1$. In what follows, whenever it is clear from the context, we will omit the security parameter κ so we will write α, β, etc. instead of $\alpha(\kappa), \beta(\kappa)$, etc.

Given a (α, β)-split-coding scheme Σ, for any $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and any function λ_0, λ_1 let $\mathsf{Real}^{\lambda_0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)$ be the following experiment:

Sampling Phase. The adversary $\mathcal{A}_0$ on input crs where $\mathsf{crs} \leftarrow\!\!\$\ \mathsf{Gen}(1^\kappa)$ and randomness r_A, outputs a message $m \in \mathcal{M}$ and a state value st'. The challenger samples $\omega \leftarrow\!\!\$\ \mathcal{R}$ and instantiates an oracle $\mathcal{O}^{\omega\|m}_{\lambda_0}$.

Encoding Phase. The adversary $\mathcal{A}_1$ gets input st, the $\widetilde{\mathsf{crs}}$ and oracle access to $\mathcal{O}^{\omega\|m}_{\lambda_0}$. The adversary notifies the challenger sending the message `encode`. The message is encoded, namely the challenger defines $(L, R) = \mathsf{Enc}(\mathsf{crs}, m; \omega)$ and instantiates the oracles $\mathcal{O}^L_{\lambda_1}$, $\mathcal{O}^R_{\lambda_1}$. The adversary $\mathcal{A}_1$ loses oracle access to $\mathcal{O}^{\omega\|m}_{\lambda_0}$ and gains oracle access to $\mathcal{O}^L_{\lambda_1}$ and $\mathcal{O}^R_{\lambda_1}$. [1]

By overloading the notation, we let $\mathsf{Real}^{\lambda_0,\lambda_1}_{\mathcal{A},\Sigma}$ be also the tuple of random variables that describe the view of $\mathcal{A}$ in the experiment:

$$\mathsf{Real}^{\lambda_0,\lambda_1}_{\mathcal{A},\Sigma} := \begin{pmatrix} r_A, \mathsf{crs}, \\ \mathsf{lk}_\omega := (\mathsf{lk}^0_\omega, \mathsf{lk}^1_\omega, \ldots, \mathsf{lk}^t_\omega), \\ \mathsf{lk}_L := (\mathsf{lk}^0_L, \mathsf{lk}^1_L, \ldots, \mathsf{lk}^{t'}_L), \\ \mathsf{lk}_R := (\mathsf{lk}^0_R, \mathsf{lk}^1_R, \ldots, \mathsf{lk}^{t''}_R) \end{pmatrix},$$

where $\mathsf{lk}^i_X := \mathcal{O}^X(f^i_X)$ with $X \in \{\omega\|m, L, R\}$ and f^i_X is the i-th oracle query made by $\mathcal{A}$ to the leakage oracle $\mathcal{O}^X$.

For any $\mathcal{A}$ and any $\mathcal{S}$ let $\mathsf{Ideal}^{\lambda_0,\lambda_1}_{\mathcal{A},\mathcal{S}}(\kappa)$ be the following experiment:

Sampling Phase. The simulator $\mathcal{S}$ on input 1^κ produces $\widetilde{\mathsf{crs}}$, the adversary $\mathcal{A}_0$ on input $\widetilde{\mathsf{crs}}$, outputs a message $m \in \mathcal{M}$ and a state value st. The experiment instantiates an oracle $\mathcal{O}^m_{\lambda_0}$. The simulator gets oracle access to $\mathcal{O}^m_{\lambda_0}$.

[1] We can also consider a strong notion, where the adversary does not lose access to $\mathcal{O}^{\omega\|m}_{\lambda_0}$ after gaining access to $\mathcal{O}^L_{\lambda_1}$ and $\mathcal{O}^R_{\lambda_1}$.

Encoding Phase. The adversary $\mathcal{A}_1$ gets input st', the $\widetilde{\mathsf{crs}}$ and oracle access to the simulator $\mathcal{S}$. Technically, the adversary $\mathcal{A}_1$ gets access first to one interface for $\mathcal{S}$ (i.e. the simulated $\mathcal{O}_{\lambda_0}^{\omega\|m}$), after the adversary sends the message `encode` to the challenger it gets access to two interfaces for $\mathcal{S}$ (i.e. simulated $\mathcal{O}_{\lambda_1}^{L}$ and simulated $\mathcal{O}_{\lambda_1}^{R}$).

As we did with $\mathsf{Real}_{\mathcal{A},\Sigma}^{\lambda_0,\lambda_1}$ we denote with $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\lambda_0,\lambda_1}$ also the tuple of random variables that describe the view of $\mathcal{A}$ in the experiment. To mark the distinction between the real experiment and ideal experiment we upper script the components of the ideal experiment with a tilde, namely:

$$\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\lambda_0,\lambda_1} = \left(\tilde{r}_A, \widetilde{\mathsf{crs}}, \widetilde{\mathsf{lk}}_\omega, \widetilde{\mathsf{lk}}_L, \widetilde{\mathsf{lk}}_R\right)$$

Given a class of leakage functions Λ we say that an adversary is Λ-bounded if it submits to $\mathcal{O}_{\lambda_0}^{\omega\|m}$ only functions $f \in \Lambda$.

Definition 3 (Simulation-based Λ-fully leakage resilient code). *A (α,β)-split-coding scheme is said to be $(\Lambda,\lambda_0,\lambda_1,\epsilon)$-FLR-sim-secure if for any* ppt *adversary $\mathcal{A}$ that is Λ-bounded there exists a* ppt *simulator $\mathcal{S}$:*

$$\left\{\mathsf{Real}_{\mathcal{A},\Sigma}^{\lambda_0,\lambda_1}(\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx}_\epsilon \left\{\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\lambda_0,\lambda_1}(\kappa)\right\}_{\kappa\in\mathbb{N}}.$$

Let $\mathsf{P}_{/\mathsf{poly}}$ be the set of all polynomial-sized circuits.

Definition 4 (Simulation-based fully leakage resilient code). *A (α,β)-split-coding scheme is said to be $(\lambda_0,\lambda_1,\epsilon)$-FLR-sim-secure if it is $(\mathsf{P}_{/\mathsf{poly}},\lambda_0,\lambda_1,\epsilon)$-FLR-sim-secure. We simply say that a split-coding scheme is (λ_0,λ_1)-FLR-sim-secure if there exists a negligible function* negl *such that the scheme is $(\lambda_0,\lambda_1,\mathsf{negl})$-FLR-sim-secure.*

In Appendix A we recall the game-based definition and we prove formally that FLR-sim-security is implied by the definition of [DDV10] for $\lambda_0 = 0$.

Given a (α,β)-split-coding scheme Σ, for any $\mathcal{A} = (\mathcal{A}_0,\mathcal{A}_1)$ and any function λ_0, λ_1 let $\mathsf{wReal}_{\mathcal{A},\Sigma}^{\lambda_0,\lambda_1}(\kappa)$ be an experiment equivalent to $\mathsf{Real}_{\mathcal{A},\Sigma}^{\lambda_0,\lambda_1}(\kappa)$ but where the Sampling Phase is substituted by the following:

Sampling Phase'. The adversary $\mathcal{A}_0$ on input 1^κ and randomness r_A, outputs a message $m \in \mathcal{M}$ and a state value st'. The challenger samples $\omega \leftarrow\!\!\$\ \mathcal{R}$ and instantiates an oracle $\mathcal{O}_{\lambda_0}^{\omega\|m}$.

Notice that crs is still included in the random variable $\mathsf{wReal}_{\mathcal{A},\Sigma}^{\lambda_0,\lambda_1}$, however the view generated before $\mathcal{A}_1$ sends the message `encode` is independent from it.

Similarly, for any $\mathcal{A}$ and any $\mathcal{S}$ let $\mathsf{wIdeal}_{\mathcal{A},\mathcal{S}}^{\lambda_0,\lambda_1}(\kappa)$ an experiment equivalent to $\mathsf{Ideal}_{\mathcal{A},\mathcal{S}}^{\lambda_0,\lambda_1}(\kappa)$ but where the Sampling Phase is substituted by the following:

Sampling Phase'. The simulator $\mathcal{S}$ on input 1^κ produces $\widetilde{\mathsf{crs}}$, the adversary $\mathcal{A}_0$ on input 1^κ outputs a message $m \in \mathcal{M}$ and a state value st. The experiment instantiates an oracle $\mathcal{O}_{\lambda_0}^{m}$. The simulator $\mathcal{S}$ gets oracle access to $\mathcal{O}_{\lambda_0}^{m}$.

Definition 5 (Simulation-based Λ-weak-fully leakage resilient code). *A (α,β)-split-coding scheme is said to be $(\Lambda,\lambda_0,\lambda_1,\epsilon)$-weak-FLR-sim-secure if for any* ppt *adversary $\mathcal{A}$ that is Λ-bounded there exists a* ppt *simulator $\mathcal{S}$:*

$$\left\{\mathsf{wReal}_{\mathcal{A},\Sigma}^{\lambda_0,\lambda_1}(\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx}_\epsilon \left\{\mathsf{wIdeal}_{\mathcal{A},\mathcal{S}}^{\lambda_0,\lambda_1}(\kappa)\right\}_{\kappa\in\mathbb{N}}.$$

4 Impossibility Results

In this section we show the main result of this paper. Throughout the section we let the class of leakage functions be $\Lambda = \mathsf{P}_{/\mathsf{poly}}$. We prove that (α, β)-split-coding schemes that are (λ_0, λ_1)-FLR-sim-secure don't exist for many interesting parameters of α, β, λ_0 and λ_1. We start with the case $\alpha(\kappa) = \Omega(\kappa)$, the impossibility results holds under the only assumption that collision resistant hash function exists. For the case $\alpha(\kappa) = O(1)$, the impossibility results holds under the stronger assumption that adaptive-secure PAoK exists.

Theorem 1. *If public-coin* $\mathsf{AoK}_{1/2,\mathsf{negl}(\kappa)}(O(1), \ell_{\mathsf{AoK}}(\kappa))$ *for* NP *and collision-resistant hash functions with output length* $\ell_{\mathsf{CRH}}(\kappa)$ *exist then for any* $\lambda_0 \geqslant \ell_{\mathsf{AoK}}(\kappa) + 2 \cdot \ell_{\mathsf{CRH}}(\kappa)$ *for any* (α, β)*-split-coding scheme* Σ *with* $\alpha(\kappa) \geqslant \lambda_0(\kappa) + \ell_{\mathsf{CRH}}(\kappa) + 7$ *and if* $\lambda_1(\kappa) \geqslant (17\lambda_0(\kappa)) \cdot \ell_{\mathsf{AoK}}(\kappa)$ *then* Σ *is not* (λ_0, λ_1)*-FLR-sim-secure.*

Proof. We first set some necessary notation. Given a random variable x we use the notation $\bar{x}$ to refer to a possible assignment of the random variable. Let Π be in $\mathsf{AoK}_{1/2,\mathsf{negl}(\kappa)}(O(1), \ell_{\mathsf{AoK}}(\kappa))$ and a public-coin argument system for NP. For concreteness let ρ be the round complexity of the Π. For any $i \in [\rho]$ let $\mathcal{P}^{\mathcal{O}}_{\mathsf{leak}}(x, y_1, \ldots, y_i; r_p)$ be a prover for Π which has oracle access to a leakage oracle $\mathcal{O}$ and which queries the oracle with the function $\mathsf{lk}(w) := \mathsf{Prove}(x, w, y_1, \ldots, y_i; r_p)$ and outputs the answer given by the oracle. Let $(\mathsf{Gen}^{\mathrm{CRH}}, \mathsf{Eval}^{\mathrm{CRH}})$ be a collision resistant hash function with output length $\ell_{\mathsf{CRH}}(\kappa)$. Consider the adversary $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1)$ that does the following:

1. Pick a collision resistant hash function $h \leftarrow \mathsf{Gen}^{\mathrm{CRH}}(1^\kappa)$;
2. Pick $m \leftarrow\!\!\$\ \mathcal{M}$ and send it to the challenger;
3. Compute $h(m)$.

This ends the code of $\mathcal{A}'_0$, formally, $\mathcal{A}'_0(1^\kappa)$ outputs m that is forwarded to the experiment which instantiates a leakage oracle $\mathcal{O}^m_{\lambda_0}$, also $\mathcal{A}'_0(1^\kappa)$ outputs the state $st := (h, h(m))$.

4. Leak from $\mathcal{O}^{\omega \| m}_{\lambda_0}$ the value $h_l = h(L), h_r = h(R)$;
5. Leak from $\mathcal{O}^{\omega \| m}_{\lambda_0}$ an argument of knowledge for the relation

$$\mathcal{R}^{\mathsf{st}} := \left\{ (x_{crs}, x_l, x_r, x_m), (w_l, w_r) : \begin{array}{c} h(w_l) = x_l \\ h(w_r) = x_r \\ h(\mathsf{Dec}(c_{crs}, w_l, w_r)) = x_m \end{array} \right\}$$

   with instance $(\mathsf{crs}, h_l, h_r, h(m))$. If the verification fails then abort.
   Technically, let r_p be a random string long enough to specify all random choices done by the prover. For $j \in [\rho]$ do the following:
   (a) Sample a random string $y_j \leftarrow\!\!\$\ \{0,1\}^\kappa$;
   (b) Execute $z_j := \mathcal{P}^{\mathcal{O}^{\omega \| m}_{\lambda_0}}_{\mathsf{leak}}\big((h_l, h_r, h(m)), y_1, \ldots, y_j; r_p \big)$.
   Let $\pi_0 := y_1, \ldots, y_\rho, z_1, \ldots, z_\rho$, if $\mathsf{Judge}\big((h_l, h_r, h(m)), \pi_0\big) = 1$ then continue, otherwise abort.
6. For $\tau := 17\lambda_0$ many times leak from $\mathcal{O}^L_{\lambda_1}$ a (succinct) AoK for the relation $\mathcal{R}^{\mathsf{hash}} := \{(y, x) : h(x) = y\}$ with instance h_l. If the verification fails then abort.
   The procedure is similar to the step above, the adversary runs the protocol as verifier with an instance of $\mathcal{P}^{\mathcal{O}^L_{\lambda_1}}_{\mathsf{leak}}(h_l)$, let π^L_i be the transcript produced, the adversary aborts if $\mathsf{Judge}(h_l, \pi^L_i) = 0$. Send the message `encode`.
7. For τ many times leak from $\mathcal{O}^R_{\lambda_1}$ a (succinct) AoK for the relation $\mathcal{R}^{\mathsf{hash}}$ with the instance h_r. If the verification fails then abort.
   As before, the adversary runs the protocol as verifier with an instance of $\mathcal{P}^{\mathcal{O}^R_{\lambda_1}}_{\mathsf{leak}}(h_r)$ the adversary aborts if $\mathsf{Judge}(h_l, \pi^R_i) = 0$ where let π^R_i is the transcript produced.

Consider the following randomized experiment $\mathbf{E}$:

- Pick uniformly random $m \leftarrow\!\!\$\ \mathcal{M}$ and $h \leftarrow\!\!\$\ \mathsf{Gen}^{\mathrm{CRH}}(1^\kappa)$ and set $st = (h, h(m))$ and forward to the predictor the state st.
- Instantiate an oracle $\mathcal{O}^m_{\lambda_0}$ and give access to the predictor to it.

Lemma 3. $\widetilde{\mathbb{H}}_\infty(m \mid \mathbf{E}) \geqslant \alpha - \ell_{\mathsf{CRH}} - \lambda_0$.

Proof. Consider the experiment $\mathbf{E}'$ which is the same as $\mathbf{E}$ except that the predictor's input is h (instead of $(h, h(m))$). We apply Lemma 1:

$$\mathbb{H}_\infty(m \mid \mathbf{E}) \geqslant \mathbb{H}_\infty\left(m \mid \mathbf{E}'\right) - \ell_{\mathsf{CRH}}.$$

Consider the experiment $\mathbf{E}''$ which is the same as $\mathbf{E}'$ except that the predictor's oracle access to $\mathcal{O}^{\omega \| m}_{\lambda_0}$ is removed. We apply Lemma 1:

$$\mathbb{H}_\infty\left(m \mid \mathbf{E}'\right) \geqslant \mathbb{H}_\infty\left(m \mid \mathbf{E}''\right) - \lambda_0.$$

In the last experiment $\mathbf{E}''$ the predictor has no information about m, therefore:

$$\mathbb{H}_\infty\left(m \mid \mathbf{E}''\right) = \log|\mathcal{M}| = \alpha.$$

□

Lemma 4. *If Σ is a (λ_0, λ_1)-FLR-sim-secure then* $\widetilde{\mathbb{H}}_\infty(m|\mathbf{E}) \leqslant 6$.

Proof. Assume that Σ is an $(\lambda_0, \lambda_1, \epsilon)$-FLR-sim-secure split-coding scheme for a negligible function ϵ. Since $\mathcal{A}'$ is ppt there exists a ppt simulator $\mathcal{S}'$ such that:

$$\{\mathsf{Real}^{\lambda_0,\lambda_1}_{\mathcal{A}',\Sigma}(\kappa)\}_\kappa \overset{c}{\approx}_{\epsilon(\kappa)} \{\mathsf{Ideal}^{\lambda_0,\lambda_1}_{\mathcal{A}',\mathcal{S}'}(\kappa)\}_\kappa. \tag{1}$$

For the sake of the proof we first build a predictor which is given $(h, h(m))$ and tries to guess m. We then use this predictor to prove the lemma. Let K be the extractor given by the knowledge soundness property of the argument of knowledge for the relation $\mathcal{R}^{\mathsf{hash}}$. Consider the following predictor $\mathcal{B}$ that takes as input $(h, h(m))$ and has oracle access to $\mathcal{O}^m_{\lambda_0}$:

1. Pick two random tapes r_a, r_s for the adversary $\mathcal{A}'_1$ and the simulator $\mathcal{S}'$ and run both of them (with the respective randomness r_a, r_s) connecting the oracle accesses of $\mathcal{A}'_1$ to the interfaces of $\mathcal{S}'$ and forwarding the queries of $\mathcal{S}'$ to the oracle $\mathcal{O}^m_{\lambda_0}$. The adversary $\mathcal{A}'_1$ starts by leaking the values h_l, h_r and an argument of knowledge for $\mathcal{R}^{\mathsf{st}}$.

2.L. The predictor tries to compute L' using the knowledge extractor K. Formally, for any $i \in [\tau]$, let $\bar{st}^L_i$ be the actual internal state of $\mathcal{S}'$ during the above run of $\mathcal{S}'$ and $\mathcal{A}'_1$ just before the i-th iteration of step 6 of $\mathcal{A}'_1$. Let $\mathcal{S}'_i$ be a copy of $\mathcal{S}'$ with internal state sets to $\bar{st}^L_i$. Abusing notation, we use $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ to denote the prover which queries the interface of $\mathcal{S}'_i$ for $\mathcal{O}^L_{\lambda_1}$, i.e., the message sent by $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ is computed by calling $\mathcal{S}'_i$ as if it was a leakage oracle and with the leakage functions being the functions computing the messages that the prover would send in a proof of knowledge of a preimage L' of h_l. Notice that when $\mathcal{S}'_i$ is run it might make an oracle query on its interface for $\mathcal{O}^m_{\lambda_0}$. When this happens $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ will intercept the query and will not try to simulate $\mathcal{O}^m_{\lambda_0}$ to $\mathcal{S}'_i$. Instead, whenever $\mathcal{S}'_i$ queries $\mathcal{O}^m_{\lambda_0}$ the program $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ will take the reply from $\mathcal{S}'_i$ to be some dummy string, say the all-zero string. This ensures that $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ makes no further leakage queries to $\mathcal{O}^m_{\lambda_0}$. The predictor runs the knowledge extractor K on the prover $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}(h_l)$, which outputs a value L' or aborts. If $h_l = h(L')$

then return L' otherwise the i-th extraction is said to abort. If all the extractions abort, the predictor aborts. [2]

2.R. The predictor computes R' using the knowledge extractor K. The procedure is the same of step 2.L of the predictor, for notational completeness let us denote with st_i^R the internal state of $\mathcal{S}'$ just before the i-th iteration of step 7.

3. The predictor outputs $m' := \mathsf{Dec}(L', R')$ as its own guess.

We compute the probability that $\mathcal{B}$ predicts m correctly. We set up some useful notation:

- Let Ext_L (resp. Ext_R) be the event that K successfully extracts a value L' (resp. R').
- Let CohSt be the event $\{h(\mathsf{Dec}(L', R')) = h(m)\}$.
- Let Coll be the event $\{h(\mathsf{Dec}(L', R')) = h(m) \wedge \mathsf{Dec}(L', R') \neq m\}$..

Recall that $m' := \mathsf{Dec}(L', R')$ is the guess of $\mathcal{B}$. We can easily derive that:

$$\Pr\left[m' = m\right] = \Pr\left[\mathsf{Ext}_L \wedge \mathsf{Ext}_R \wedge \mathsf{CohSt} \wedge \neg\mathsf{Coll}\right] \tag{2}$$

In fact, Ext_L and Ext_R imply that L' and R' are well defined and the event $(\mathsf{CohSt} \wedge \neg\mathsf{Coll})$ implies that $\mathsf{Dec}(L', R') = m$.

Claim 1 $\Pr[\mathsf{Ext}_L] \geqslant \frac{1}{4} - \mathsf{negl}(\kappa)$.

Proof. Consider the execution of step 6 between the adversary and the simulator. Let $\bar{\mathbf{st}} = \bar{st}_1^L, \ldots, \bar{st}_\tau^L \in \{0,1\}^*$ be a fixed observed value of the states of $\mathcal{S}'$ in the different rounds, i.e., $\bar{st}_i^L$ is the observed state of $\mathcal{S}'$ just before the i-th iteration in step 6.

We define a probability $\mathsf{Free}_L(\bar{st}_i^L)$ of the simulator not asking a leakage query in round i, i.e., the probability that the simulator queries its leakage oracle if run with fresh randomness starting in round i. We can assume without loss of generality that the randomness r_S of the simulator is part of $\bar{st}_i^L$. Therefore the probability is taken over just the randomness r_A of the adversary, m, h and the challenges used in the proof in round i. Notice that even though it might be fixed in $\bar{\mathbf{st}} = \bar{st}_1^L, \ldots, \bar{st}_\tau^L$ whether or not the simulator leaked in round i (this information might be contained in the final state $\bar{st}_\tau^L$), the probability $\mathsf{Free}_L(\bar{st}_i^L)$ might not be 0 or 1, as it is the probability that the simulator will leaked in round i if we would rerun round i with fresh randomness of the adversary consistent with $\bar{st}_i^L$.

Now let $\bar{\mathbf{st}} = \bar{st}_1^L, \ldots, \bar{st}_\tau^L \in \{0,1\}^*$ be a fixed observed value of the states of $\mathcal{S}'$ in the different rounds. Let $\mathsf{Good}(\bar{\mathbf{st}})$ be a function which is 1 if

$$\exists i \in [\tau] : \mathsf{Free}_L(\bar{st}_i^L) \geqslant \frac{3}{4}$$

and which is 0 otherwise. [3] After having defined $\mathsf{Good}(\bar{\mathbf{st}})$ relative to a fixed observed sequence of states, we now apply it to the random variable $\mathbf{st}$ describing the states of $\mathcal{S}'$ in a random run. When applied to this random variable, we write Good.

We can use the law of total probability to condition the extraction event to the event $\{\mathsf{Good} = 1\}$:

$$\Pr[\mathsf{Ext}_L] \geqslant \Pr[\mathsf{Ext}_L \,|\mathsf{Good}] \cdot \Pr[\mathsf{Good}] \,. \tag{3}$$

[2] To get an intuition why this might work, notice that in the real world the proof given by the prover hardcoded into the leakage function has soundness essentially 1. Hence if we ran $\mathcal{S}'_i$ and actually gave it access to query $\mathcal{O}^m_{\lambda_0}$, then $\mathcal{S}'_i$ we would also return a proof which is accepted with probability close to 1. Hence, if the probability that $\mathcal{S}'_i$ queries $\mathcal{O}^m_{\lambda_0}$ is for instance less than $\frac{1}{4}$, then $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ returns a valid proof with probability negligibly close to $\frac{3}{4}$. So, if the proof system has error soundness just $\frac{1}{2}$, then because $\frac{3}{4} > \frac{1}{2}$ we can extract a preimage L' of h_l from $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$. So, the extraction fails only if $\mathcal{S}'_i$ called the leakage oracle with probability more than $\frac{1}{4}$ in each iteration. By design this would lead it to violate the leakage bound except with negligible probability.

[3] Intuitively, Good is an indicator for a good event, that, as we will show, has overwhelming probability.

We will now focus on bounding $\Pr[\mathsf{Ext}_L \,|\mathsf{Good}] \cdot \Pr[\mathsf{Good}]$. We first bound $\Pr[\mathsf{Good}]$ and then bound $\Pr[\mathsf{Ext}_L \,|\mathsf{Good}]$.

We first prove that

$$\Pr[\mathsf{Good}] = 1 - \mathsf{negl}(\kappa) \; .$$

Too see this notice that the simulator by the rules of the game never queries its leakage oracle in more than λ_0 rounds: it is not allowed to leak more than λ_0 bits and each leakage query counts as at least one bit. Therefore there are at least $\tau - \lambda_0$ rounds in which the simulator did not query its oracle. If $\mathsf{Good} = 0$, then in each of these rounds the probability of leaking, before the round was executed, was at least $\frac{1}{4}$ and hence the probability of not leaking was at most $\frac{3}{4}$. We can use a union bound to bound the probability of observing this event

$$\Pr[\mathsf{Good} = 0] \leq \binom{\tau}{\tau - \lambda_0} \left(\frac{3}{4}\right)^{\tau - \lambda_0} \leq \binom{\tau}{\lambda_0} 2^{\log_2(3/4)(\tau - \lambda_0)} \; .$$

We now use that $\tau = 17\lambda_0$ and that it holds for any constant $c \in (0,1)$ that $\lim_{n\to\infty} \binom{n}{cn} = 2^{H_2(c)}$, where H_2 is the binary entropy function. We get that

$$\Pr[\mathsf{Good} = 0] \leq 2^{H_2(1/17)17\lambda_0} 2^{\log_2(3/4)16\lambda_0} = (2^{H_2(1/17)17 + \log_2(3/4)16})^{\lambda_0} < 2^{-\lambda_0} \; .$$

We now bound $\Pr[\mathsf{Ext}_L \,|\mathsf{Good}]$. Let $\mathsf{Ext}_L(i)$ be the event that K successfully extracts the value L' at the i-th iteration of the step 6 of the adversary $\mathcal{A}'$. Let $\mathsf{Accept}_L(i)$ be the event that $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ gives an accepting proof. It follows from knowledge soundness of Π that

$$\Pr\left[\mathsf{Ext}_L(i)|\mathsf{Good}\right] \geqslant \Pr\left[\mathsf{Accept}_L(i)|\mathsf{Good}\right] - \tfrac{1}{2} \; .$$

Let $\mathsf{Leak}_L(i)$ be the event that the simulator queries its leakage oracle in round i. Then also holds for all i that

$$\Pr\left[\mathsf{Accept}_L(i)|\mathsf{Good}\right] \geq 1 - \Pr\left[\mathsf{Leak}_L(i)|\mathsf{Good}\right] - \mathsf{negl}(\kappa) \; .$$

To see this assume that we would actually give the simulator access to the leakage oracle instead of aborting when it queries the oracle. In that case it gives acceptable proof with probability $1 - \mathsf{negl}(\kappa)$ as the prover gives an acceptable proof in the real world and the simulator simulates the real world up to negligible difference. Furthermore, aborting when the simulator queries its oracle can only make a difference when it actually queries, which happens with probability $\Pr[\mathsf{Leak}_L(i)]$

Combining the above inequalities we get that

$$\Pr\left[\mathsf{Ext}_L(i)|\mathsf{Good}\right] \geqslant 1 - \Pr\left[\mathsf{Leak}_L(i)\middle|\mathsf{Good}\right] - \mathsf{negl}(\kappa) - \tfrac{1}{2} \; .$$

When $\mathsf{Good} = 1$ there exists some round i^* such that

$$\mathsf{Free}_L(\bar{st}^L_{i^*}) \geqslant \frac{3}{4} \; ,$$

which implies that

$$\Pr\left[\mathsf{Ext}_L(i^*)|\mathsf{Good}\right] \geqslant \frac{3}{4} - \mathsf{negl}(\kappa) - \frac{1}{2} \; .$$

Clearly $\mathsf{Ext}_L(i^*)$ implies Ext_L, so we conclude that $\Pr\left[\mathsf{Ext}_L|\mathsf{Good}\right] \geqslant \frac{1}{4} - \mathsf{negl}(\kappa)$.

Claim 2 $\Pr[\mathsf{Ext}_R|\mathsf{Ext}_L] \geqslant \frac{1}{4} - \mathsf{negl}(\kappa)$.

The proof proceeds similar to the proof of the last claim, therefore it is omitted. The reason why the condition Ext_L does not matter is that proof exploits essentially only the knowledge soundness of the proof system. Whether the extraction of the of the left part succeeded or not does not remove the knowledge soundness of the proofs for the right part, which are done *after* the proof for the left part.

Claim 3 $\Pr[\mathsf{CohSt} \,|\mathsf{Ext}_L \wedge \mathsf{Ext}_R] \geqslant \frac{1}{2} - \mathsf{negl}(\kappa)$.

Proof. We reduce to the collision resistance property of h and the knowledge soundness of the argument system $\mathit{\Pi}$, Suppose that

$$\Pr[h(\mathsf{Dec}(L', R')) \neq h(m) \,|\mathsf{Ext}_L \wedge \mathsf{Ext}_R] \geqslant 1/\mathsf{poly}(\kappa)$$

Consider the following collision finder adversary $\mathcal{B}_{coll}(h)$:

- Sample uniformly random $m \leftarrow\!\!\$\ \mathcal{M}$ and random $h \leftarrow\!\!\$\ \mathsf{Gen}^{\mathrm{CRH}}(1^\kappa)$;
- Run an instance of the predictor $\mathcal{B}^{\mathcal{O}^m_{\lambda_0}}(h, h(m))$, the predictor needs oracle access to $\mathcal{O}^m_{\lambda_0}$ which can be simulated by $\mathcal{B}_{coll}(h)$.
- Let L', R' as defined by the execution of the predictor $\mathcal{B}$ and let r_a, r_s be the same randomness picked by $\mathcal{B}$ in its step 1. Simulate an execution of $\mathcal{A}'_1(h, h(m); r_a)$ and $\mathcal{S}'(1^\kappa; r_s)$ and break them just before the adversary leaks an argument of knowledge for $\mathcal{R}^{\mathsf{st}}$. Let $\mathcal{S}''$ be the a copy of the simulator with the internal state set as after the break, and oracle access to $\mathcal{O}^m_\alpha$.
- Run the knowledge extractor K_{st} on the prover $\mathcal{P}^{\mathcal{S}''}_{\mathsf{leak}}\big((h(L'), h(R'), h(m))\big)$. Let L'', R'' the witness output by the extractor.
- If $L' \neq L''$ output (L', L'') else (R', R'').

It easy to check that $\mathcal{B}_{coll}$ simulates perfectly the randomized experiment $\mathbf{E}$ therefore:

$$\begin{aligned}\Pr[h(\mathsf{Dec}(L', R')) \neq h(m)] \geqslant \\ \geqslant \Pr[h(\mathsf{Dec}(L', R')) \neq h(m) \,|\mathsf{Ext}_L \wedge \mathsf{Ext}_R] \Pr[\mathsf{Ext}_L \wedge \mathsf{Ext}_R] \geqslant \\ \geqslant 1/\mathsf{poly}(\kappa) \cdot (\tfrac{1}{16} - \mathsf{negl}(\kappa))\end{aligned}$$

On the other hand, the extractor K_{st} succeeds with probability at least $1 - \mathsf{negl}(\kappa) - \frac{1}{2}$. Therefore, L'' and R'' are such that $h(L'') = h(L')$, $h(R'') = h(R')$ and $h(\mathsf{Dec}(L'', R'')) = h(m)$. The latter implies $h(\mathsf{Dec}(L', R')) \neq h(\mathsf{Dec}(L'', R''))$ which implies that either $L'' \neq L'$ or $R'' \neq R'$. Lastly, notice that $\mathcal{B}_{coll}$ is an expected polynomial time algorithm, however we can make it polynomial time by aborting if the number of step exceed some fixed polynomial, by setting the polynomial big enough the probability of $\mathcal{B}_{coll}$ finding a collision is still noticeable.

Claim 4 $\Pr\left[\mathsf{Coll} \,|\mathsf{CohSt} \wedge \mathsf{Ext}_L \wedge \mathsf{Ext}_R\right] \leqslant \mathsf{negl}(\kappa)$.

Remind that Coll is the event that $h(m) = h(m')$ but $m \neq m'$. It can be easily verified that under collision resistance of h the claim holds, therefore the proof is omitted.

Summarizing, we have:

$$\begin{aligned}\Pr[m' = m] = \Pr\left[\mathsf{Ext}_L \wedge \mathsf{Ext}_R \wedge \mathsf{CohSt} \wedge \neg\mathsf{Coll}\right] \geqslant \\ \geqslant (\tfrac{1}{16} - \mathsf{negl}(\kappa)) \cdot (\tfrac{1}{2} - \mathsf{negl}(\kappa)) \cdot (1 - \mathsf{negl}(\kappa)) \geqslant \tfrac{1}{64}.\end{aligned}$$

which implies the statement of the lemma.

We conclude the proof of the theorem noticing that, if Σ is (λ_0, λ_1)-FLR-sim-secure split-coding scheme by the parameter given in the statement of the theorem we have that Lemma 3 and Lemma 4 are in contraction. □

Remark 2. The result can be generalized for a weaker version of the split-state model where the codeword is split in many parts. It's not hard to see that the probability that the predictor in Lemma 4 guesses the message m degrades exponentially in the number of splits (the adversary needs to leak one hash for each split and then executes step 6 for any split). Therefore, the impossibility holds when the number of splits is $o((\alpha - \lambda_0)/\ell_{\mathsf{CRH}})$. We present the theorem, as stated here, for sake of simplicity.

4.1 The case of constant-size message.

Theorem 2. *If public-coin* $\mathsf{AoK}_{1/2,\mathsf{negl}(\kappa)}(O(1), \ell_{\mathsf{AoK}}(\kappa))$ *for* NP*, adaptive-secure PAoK for* NP *with answer length* 1 *and collision-resistant hash functions with output length* 1 *exist then for any* $\lambda_0 \geqslant 1$ *for any* (α, β)*-split-coding scheme* Σ *with* $\alpha(\kappa) \geqslant 7$ *and if* $\lambda_1(\kappa) \geqslant (17\lambda_0(\kappa)) \cdot \ell_{\mathsf{AoK}}(\kappa)$ *it holds that* Σ *is not* (λ_0, λ_1)*-FLR-sim-secure.*

In what follows we give a sketch of the proof of Theorem 2 stressing mainly on the main differences with the proof of Theorem 1. The formal proof is given in Appendix B.

Proof (Sketch.). Consider the adversary $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1)$ which is the same of Theorem 1 but steps 3 and 4 are substituted by the following:

3. Sample $(c, tp) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa)$ and leak from $\mathcal{O}_{\lambda_0}^{\omega \| m}$ the function which compute the response

$$f_0(\omega \| m) := \mathsf{Resp}(\ (\mathsf{Enc}(\mathsf{crs}, m; \omega), m),\ \mathsf{Enc}(\mathsf{crs}, m; \omega),\ c),$$

   in a proof for the relation $\mathcal{R}$ as defined below:

$$\left\{ (h_l, h_r, m), (L, R) : \begin{array}{c} h(L) = h_l \\ h(R) = h_r \\ \mathsf{Dec}(L, R) = m \end{array} \right\}.$$

   Let lk_0 be the oracle answer.
4. Send the message `encode`. Leak from $\mathcal{O}_{\lambda_1}^{L}$ (resp. $\mathcal{O}_{\lambda_1}^{R}$) the value $h_l = h(L)$ (resp. $h_r = h(R)$). Check if $\mathsf{TResp}(tp,\ (h_l, h_r, m)) = \mathsf{lk}_0$, abort if not.

Consider the following randomized experiment $\mathbf{E}$:

- Pick uniformly random $m \leftarrow\!\!\$\ \mathcal{M}$ and $h \leftarrow\!\!\$\ \mathsf{Gen}^{\mathrm{CRH}}(1^\kappa)$ and forward to the predictor the value h.
- Instantiate an oracle $\mathcal{O}_{\lambda_0}^{m}$ and give the predictor access to to it.

Lemma 5. $\mathbb{H}_\infty(m|\mathbf{E}) \geqslant |m| - \lambda_0$.

The only difference from Lemma 3 is that the experiment $\mathbf{E}$ does not feed the predictor with the value $h(m)$.

Lemma 6. *If* Σ *is an* (λ_0, λ_1)*-FLR-sim-secure then* $\widetilde{\mathbb{H}}_\infty(m|\mathbf{E}) \leqslant 6$.

The proof of the lemma follows the same line of Lemma 4. Notice that in the experiment $\mathbf{E}$ the predictor does not get input m (which is trying to predict!), so it cannot fully emulate the adversary $\mathcal{A}'$, therefore we consider an adversary $\mathcal{A}''$ which is equivalent to $\mathcal{A}'$ but does not check that $\mathsf{TResp}(tp,\ (h_l, h_r, m)) = \mathsf{lk}_0$ (in the case the condition holds we say that lk_0 is *valid*). The predictor, similar to Lemma 4, simulates the interactions between $\mathcal{A}''$ and $\mathcal{S}'$ (the simulator for $\mathcal{A}'$) and then extracts the values L and R.

Notice that the description of the leakage function f_0 is independent of the message m as provided by the syntax of adaptive-secure PAoK and, conditioned on lk_0 is valid, the interactions between $\mathcal{A}'$ and $\mathcal{S}'$ and the interactions between $\mathcal{A}''$ and $\mathcal{S}'$ are indistinguishable. Moreover, the event happens with overwhelming probability.

5 Feasibility Results

In this section we give two feasibility results for weaker models of security.

5.1 The Inner-Product Extractor is a NC^0-Fully LR Code

We start by giving a well-known characterization of the class NC^0.

Lemma 7. *Let $f \in \mathsf{NC}^0$ where $f := \left(f^n : \{0,1\}^n \to \{0,1\}^{m(n)}\right)_{n\in\mathbb{N}}$ for a function m. For any n there exists a value $c = O(m)$, a set $\{i_1, \ldots, i_c\} \subseteq [n]$ of indexes and a function g such that for any $x \in \{0,1\}^n$,*

$$f(x) = g(x_{i_1}, x_{i_2}, \ldots, x_{i_c}).$$

The lemma above shows that any function in NC^0 with output length m such that $m(n)/n = \omega(1)$ cannot be collision resistant, because an adversary can guess an index $i \notin \{i_1, \ldots, i_c\}$ and output $0^n, (0^{i-1}\|1\|0^{n-i})$ as collision.

Let $\mathbb{F}$ be a finite field and let $\Phi^n_{\mathbb{F}} = (\mathsf{Enc}, \mathsf{Dec})$ be as follows:

- Enc on input $m \in \mathbb{F}$ picks uniformly random $\boldsymbol{L}, \boldsymbol{R} \leftarrow\!\!\$\ \mathbb{F}^n$ under the condition that $\langle \boldsymbol{L}, \boldsymbol{R}\rangle = m$.
- Dec on input $\boldsymbol{L}, \boldsymbol{R}$ outputs $\langle \boldsymbol{L}, \boldsymbol{R}\rangle$.

Theorem 3 (from [DF11]). *The encoding scheme $\Phi^n_{\mathbb{F}}$ as defined above for $|\mathbb{F}| = \Omega(\kappa)$ is a $(0, 0.3\cdot|\mathbb{F}^n|)$-FLR-SIM-secure for $n > 20$.*

We will show now that the scheme is also fully leakage resilient for NC^0-bounded adversaries.

Theorem 4. *For any $n \in \mathbb{N}$ and $n > 20$ there exists a positive constant $\delta \in \mathbb{R}$ such that, for any λ_0, λ_1 such that $\delta \cdot \lambda_0 + \lambda_1 < 0.3 \cdot |\mathbb{F}^n|$ the encoding scheme $\Phi^n_{\mathbb{F}}$ is $(\mathsf{NC}^0, \lambda_0, \lambda_1)$-FLR-SIM-secure.*

Proof. Given a vector $\boldsymbol{X} \in \mathbb{F}^n$ let $bit(X)_i$ be the i-th bit of a canonical bit-representation of $\boldsymbol{X}$. Given $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ we define a new adversary $\mathcal{A}'$ that works as follow:

0. Instantiate an execution of $(m, st) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa)$;
1. Execute $\mathcal{A}_1(st)$ and reply to the leakage oracle queries it makes as follow:
   - Upon leakage query f to $\mathcal{O}^{\omega\|m}_{\lambda_0}$, let I be the set of indexes such that f depends on I only. Define $I_L := I \cap [qn]$ and $I_R := I \cap [qn+1, 2qn]$. Leak from $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ (resp. from $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$) the values $bit(\boldsymbol{L})_i$ for $i \in I_L$ (resp. $bit(\boldsymbol{L})_i$ for $i \in I_R$) and hardwire such values and evaluate the function f on input m. Namely, compute $\mathsf{lk}_f := f((bit(\boldsymbol{L})_i)_{i\in I_L}, (bit(\boldsymbol{R})_i)_{i\in I_R}, m)$ and forward it back to $\mathcal{A}_1(st)$.
   - Upon leakage query f to $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ or $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$ proxy the leakage query to the appropriate oracle.

W.l.o.g. assume that every leakage query to $\mathcal{O}^{\omega\|m}_{\lambda_0}$ has output length 1 and that the adversary makes exactly λ_0 queries. By Lemma 7 there exists a constant $\delta \in \mathbb{N}$ such that for the i-th leakage query made by $\mathcal{A}$ to $\mathcal{O}^{\omega\|m}_{\lambda_0}$ the adversary $\mathcal{A}'$ leaks δ bits from $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}, \mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$. By construction:

$$\{\mathsf{Real}^{\lambda_0,\lambda_1}_{\mathcal{A},\Phi^n_{\mathbb{F}}}(\kappa)\}_{\kappa\in\mathbb{N}} \equiv \{\mathsf{Real}^{0,\lambda_1+\delta\cdot\lambda_0}_{\mathcal{A}',\Phi^n_{\mathbb{F}}}(\kappa)\}_{\kappa\in\mathbb{N}}\ .$$

Let $\mathcal{S}'$ be the simulator for the adversary $\mathcal{A}'$ as provided by Theorem 3, thus:

$$\{\mathsf{Real}^{0,\lambda_1+\delta\cdot\lambda_0}_{\mathcal{A}',\Phi^n_{\mathbb{F}}}(\kappa)\}_{\kappa\in\mathbb{N}} \approx_{\mathsf{negl}(\kappa)} \{\mathsf{Ideal}^{0,\lambda_1+\delta\cdot\lambda_0}_{\mathcal{A}',\mathcal{S}'}(\kappa)\}_{\kappa\in\mathbb{N}}.$$

Let $\mathcal{S}$ be defined as the machine that runs the adversary $\mathcal{A}'$ interacting with the simulator $\mathcal{S}'$ instead of its own oracles. Notice that:

$$\{\mathsf{Ideal}^{0,\lambda_1+\delta\cdot\lambda_0}_{\mathcal{A}',\mathcal{S}'}(\kappa)\}_{\kappa\in\mathbb{N}} \equiv \{\mathsf{Ideal}^{\lambda_0,\lambda_1}_{\mathcal{A},\mathcal{S}}(\kappa)\}_{\kappa\in\mathbb{N}}.$$

This conclude the proof of the Theorem. □

Remark 3. The proof exploits only marginally the structure of $\Phi^n_{\mathbb{F}}$. It is not hard to see that the theorem can be generalized for any coding scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where for any message $m \in \mathcal{M}$ the function $\mathsf{Enc}(m; \cdot)$ is invertible in NC^0. We present the theorem, as stated here, only for sake of concreteness.

Remark 4. The construction is secure under the slightly stronger definition where the adversary does not lose access to $\mathcal{O}^{\omega \| m}_{\lambda_0}$ after gaining access to $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ and $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$.

5.2 A Compiler from LRC to weak FLRC.

Given a (α, β)-split-coding scheme $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with randomness space $\mathcal{R}$, let $\mathcal{H}_{r,t}$ denote a family of efficiently computable t-wise independent hash function with domain $\{0,1\}^r$ and co-domain $\mathcal{R}$.

We define $\Sigma' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}' := \mathsf{Dec})$ as follow:

- Gen' on input 1^κ executes $\mathsf{crs} \leftarrow\!\!\$\, \mathsf{Gen}(1^\kappa)$ and samples a function $h \leftarrow\!\!\$\, \mathcal{H}_{r,t}$. It outputs $\mathsf{crs}' = (h, \mathsf{crs})$.
- Enc' on input a message $m \in \mathcal{M}$ and (crs, h) picks a random string $\omega \leftarrow\!\!\$\, \mathcal{R}'$ and returns as output $\mathsf{Enc}(\mathsf{crs}, m; h(\omega))$.

Theorem 5. *For any encoding scheme Σ and any leakage class $\mathcal{F}$, if Σ is $(0, \lambda_1, \epsilon)$-FLR-SIM-secure then Σ' is $(\mathcal{F}, \lambda_0, \lambda_1, 3\epsilon)$-weak-FLR-SIM-secure for any $0 \leqslant \lambda_0 < \alpha$ whenever:*

$$r \geqslant \lambda_0 + \lambda_1 + 2\log(1/\epsilon) + \log(t) + 3,$$

$$t \geqslant \lambda_0 \cdot \log|\mathcal{F}| + \alpha + \lambda_0 + \lambda_1 + 2\log(1/\epsilon).$$

Proof. Given an adversary $\mathcal{A}'$ against Σ', we define an adversary $\mathcal{A}$ against Σ. Let $\mathcal{A}$ be a ppt adversary defined as follow:

0. Pick at random $h' \leftarrow\!\!\$\, \mathcal{H}_{r,t}$, run $\mathcal{A}'_0(1^\kappa)$ which outputs m, st and pick a random string $\omega \leftarrow\!\!\$\, \mathcal{R}$.
1. Run $\mathcal{A}'_1(st, (\mathsf{crs}, h'))$, upon leakage oracle query f to $\mathcal{O}^{\omega\|m}_{\lambda_0}$ from $\mathcal{A}'_1$ reply forwarding $f(\omega\|m)$. When $\mathcal{A}'_1$ sends the message `encode` forward the message to the challenger. Upon leakage oracle queries to either $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ or $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$ from $\mathcal{A}'_1$ proxy the query to the proper oracle.

W.l.o.g. the adversary $\mathcal{A}'$ queries a sequence of λ_0 functions $f_1, \ldots, f_{\lambda_0}$ to $\mathcal{O}^{\omega\|m}_{\lambda_0}$. Let $\boldsymbol{f} \in \mathcal{F}^{\lambda_0}$ be such a sequence.

Claim 5 $\left\{\mathsf{wReal}^{\lambda_0,\lambda_1}_{\mathcal{A}',\Sigma'}(\kappa)\right\}_{\kappa\in\mathbb{N}} \approx_{2\epsilon} \left\{h', \boldsymbol{f}(\omega\|m), \mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)\right\}_{\kappa\in\mathbb{N}}$.

Before proceeding with the proof of the claim we show how the theorem follows.

Let $\mathcal{S}$ be the simulator for the adversary $\mathcal{A}$ as given by the hypothesis of the theorem:

$$\left\{\mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx}_{\epsilon(\kappa)} \left\{\mathsf{Ideal}^{0,\lambda_1}_{\mathcal{A},\mathcal{S}}(\kappa)\right\}_{\kappa\in\mathbb{N}}. \tag{4}$$

Let $\mathcal{S}'$ be defined as the adversary $\mathcal{A}$ which interacts with the simulator $\mathcal{S}$ instead of its own oracles. Notice that $\mathcal{S}'$ does not get the message m, however can forward the functions $f_i(\omega\| \ \cdot \ )$ for $i \in [\lambda_0]$ to its own oracle $\mathcal{O}^m_{\lambda_0}$, also notice that[4]:

$$\mathsf{Ideal}^{\lambda_0,\lambda_1}_{\mathcal{A}',\mathcal{S}'}(\kappa) \equiv \left(h', \boldsymbol{f}(\omega\|m), \mathsf{Ideal}^{0,\lambda_1}_{\mathcal{A},\mathcal{S}}(\kappa)\right).$$

It follows from a simple reduction to Eq. (4) that:

$$\left\{h', \boldsymbol{f}(\omega\|m), \mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)\right\} \overset{c}{\approx}_{\epsilon(\kappa)} \left\{h', \boldsymbol{f}(\omega\|m), \mathsf{Ideal}^{0,\lambda_1}_{\mathcal{A},\mathcal{S}}(\kappa)\right\}.$$

We conclude by applying the Claim 5 to the left hand side of the equation above. □

[4] To be precise, the equation is true if the variables are reordered properly.

Proof (of the claim). We prove that the event

$$\left\{\mathsf{wReal}^{\lambda_0,\lambda_1}_{\mathcal{A}',\Sigma'}(\kappa)\ \middle|h = h'\right\} \approx_\epsilon \left\{h', \boldsymbol{f}(\omega\|m), \mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)\right\}$$

holds with probability $(1-\epsilon)$ over $h' \leftarrow\!\!\$\ \mathcal{H}_{r,t}(1^\kappa)$, which implies the statement of the claim. Since we are proving statistical closeness we can de-randomize the adversary $\mathcal{A}'$ by setting the random string that maximize the distinguishability of the two random variables. Similarly we can de-randomize the common reference string generation algorithm Gen. Therefore, w.l.o.g. we can consider $\mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}$ to be the tuple of random variables $(\mathsf{lk}_L, \mathsf{lk}_R)$.

Let Bad be the event defined over the probability space $\mathcal{H}_{r,t}$ that:

$$\left\{\mathsf{wReal}^{\lambda_0,\lambda_1}_{\mathcal{A}',\Sigma'}(\kappa)\ \middle|h = h'\right\} \not\approx_\epsilon \left\{h', \boldsymbol{f}(\omega\|m), \mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)\right\}.$$

Notice that the adversary $\mathcal{A}$ defines for $\mathcal{A}'$ a hybrid environment where the leakage on the randomness is on ω but the codeword is instantiated using fresh randomness ω'. Let Hyb be such a random variable, i.e.,

$$\mathsf{Hyb} = \left(\mathsf{lk}_\omega, \mathsf{lk}_L, \mathsf{lk}_R\ |(L,R) = \mathsf{Enc}(\ (h,\mathsf{crs}), m;\ \omega')\right)\ .$$

Similarly, let $\mathsf{Real} := (\mathsf{lk}_\omega, \mathsf{lk}_L, \mathsf{lk}_R)$. Notice that both random variables depend on $h \leftarrow\!\!\$\ \mathcal{H}_{r,t}$.

$$\begin{aligned}\Pr[\mathsf{Bad}] &\leq \Pr_{h' \leftarrow\!\!\$\ \mathcal{H}_{r,t}}\left[\exists f_1,\ldots,f_{\lambda_0} \in \mathcal{F}, m \in \mathcal{M} : \mathsf{Real} \not\approx_\epsilon \mathsf{Hyb}\right] \\ &\leqslant \sum_{\boldsymbol{f}\in\mathcal{F}^{\lambda_0}} \sum_{m\in\mathcal{M}} \Pr_{h' \leftarrow\!\!\$\ \mathcal{H}_{r,t}}\Big[\sum_v \big|\Pr_\omega[\mathsf{Real}=v] - \Pr_{\omega,\omega'}[\mathsf{Hyb}=v]\big| > 2\epsilon\Big]\end{aligned}$$

Let $\lambda := \lambda_0 + \lambda_1$ and let $p_v := \Pr_{\omega,\omega'}[\mathsf{Hyb}=v]$. Define $\tilde{p}_v := \max\{p_v, 2^{-\lambda}\}$. Note that:

$$\sum_{v\in\{0,1\}^\lambda} \tilde{p}_v \leqslant \sum_v p_v + \sum_v 2^{-\lambda} \leqslant 2$$

Define the indicator random variable $Y_{\bar{\omega},v}$ for the event $\{\mathsf{Real} = v\ |\omega = \bar{\omega}\}$, where the randomness is over the choice of $h \leftarrow\!\!\$\ \mathcal{H}_{r,t}$.

For a fixed view v, the random variables $\{Y_{\bar{\omega},v}\}_{\bar{\omega}\in\{0,1\}^r}$ are t-wise independent. Moreover, $\mathbb{E}[\sum_{\bar{\omega}\in\{0,1\}^r} Y_{\bar{\omega},v}] = 2^r p_v$. In fact, for any $\bar{h}\in\mathcal{H}$, any $\bar{\omega}\in\{0,1\}^r$ and any $v\in\{0,1\}^\lambda$ it holds that $\Pr_h[\mathsf{Real}=v\ |\omega=\bar{\omega}] = \Pr_{\omega'}[\mathsf{Hyb}=v\ |\omega=\bar{\omega}, h'=\bar{h}]$.

It follows that

$$\begin{aligned}&\Pr_{h' \leftarrow\!\!\$\ \mathcal{H}_{r,t}}\Big[\sum_v \big|\Pr_\omega[\mathsf{Real}=v] - p_v\big| > 2\epsilon\Big] \\ &\quad\leqslant \Pr_{h' \leftarrow\!\!\$\ \mathcal{H}_{r,t}}\Big[\exists v : \big|\Pr_\omega[\mathsf{Real}=v] - p_v\big| > \epsilon\cdot\tilde{p}_v\Big] \\ &\quad\leqslant \sum_{v\in\{0,1\}^\lambda} \Pr_{h' \leftarrow\!\!\$\ \mathcal{H}_{r,t}}\Big[\big|\Pr_\omega[\mathsf{Real}=v] - p_v\big| > \epsilon\cdot\tilde{p}_v\Big] \\ &\quad\leqslant \sum_{v\in\{0,1\}^\lambda} \Pr_{h' \leftarrow\!\!\$\ \mathcal{H}_{r,t}}\Big[\big|\sum_{\bar{\omega}} Y_{\bar{\omega},v} - 2^r p_v\big| > 2^r\epsilon\cdot\tilde{p}_v\Big] \\ &\quad\leqslant \sum_{v\in\{0,1\}^\lambda} 8\left(\frac{t\cdot 2^r p_v + t^2}{(2^r\epsilon\cdot\tilde{p}_v)^2}\right)^{t/2} \qquad (5) \\ &\quad\leqslant \sum_{v\in\{0,1\}^\lambda} 8\left(\frac{2t\cdot 2^r \tilde{p}_v}{(2^r\epsilon\cdot\tilde{p}_v)^2}\right)^{t/2} \qquad (6) \\ &\quad\leqslant 2^\lambda\cdot 8\left(\frac{2t}{2^{r-\lambda}\cdot\epsilon^2}\right)^{t/2} \qquad (7)\end{aligned}$$

where Eq. (5) follows by Lemma 2 and Eq. (6) and Eq. (7) follow because $2^r \cdot \tilde{p}_v \geqslant 2^{r-\lambda} \geqslant t$. Combining all together we have:

$$\Pr[\mathsf{Bad}] \leqslant |\mathcal{F}|^{\lambda_0} \cdot |\mathcal{M}| \cdot 2^{\lambda_0+\lambda_1} \cdot 8 \left(\frac{2t}{2^{r-\lambda_0-\lambda_1} \cdot \epsilon^2} \right)^{t/2} .$$

To make the above negligible we can set:

$$r \geqslant \lambda_0 + \lambda_1 + 2\log(1/\epsilon) + \log(t) + 3,$$

$$t \geqslant \lambda_0 \cdot \log|\mathcal{F}| + \alpha + \lambda_0 + \lambda_1 + 2\log 1/\epsilon.$$

References

ABG+13. Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. http://ia.cr/2013/689.

ADKO15. Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In *TCC, Part I*, pages 398–426, 2015.

ADW09. J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

AGP14. Prabhanjan Ananth, Vipul Goyal, and Omkant Pandey. Interactive proofs under continual memory leakage. In *CRYPTO*, pages 164–182, 2014.

AMP15. Marcin Andrychowicz, Daniel Masny, and Edoardo Persichetti. Leakage-resilient cryptography over large finite fields: Theory and practice. In *ACNS*, 2015.

BCC+14. Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the snark. Cryptology ePrint Archive, Report 2014/580, 2014. http://ia.cr/2014/580.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

BCH12. Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In *TCC*, pages 266–284, 2012.

BCP14. Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

BDL14. Nir Bitansky, Dana Dachman-Soled, and Huijia Lin. Leakage-tolerant computation with input-independent preprocessing. In *CRYPTO*, pages 146–163, 2014.

BGK14. Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. *Distributed Computing*, 27(3):147–164, 2014.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

BR94. Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *FOCS*, pages 276–287, 1994.

BSW13. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. *J. Cryptology*, 26(3):513–558, 2013.

DDV10. Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *SCN*, pages 121–137, 2010.

DF11. Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. In *ASIACRYPT*, pages 702–721, 2011.

DLWW11. Yevgeniy Dodis, Allison B. Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. In *FOCS*, pages 688–697, 2011.

DP08. Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.

DPW10. Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.

FMNV14. Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *TCC*, pages 465–488, 2014.

FMVW14. Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *EUROCRYPT*, pages 111–128, 2014.

FNV14. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Mind your coins: Fully leakage-resilient signatures with graceful degradation. *IACR Cryptology ePrint Archive*, 2014:913, 2014.

FNV15. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. volume 2015, page 740, 2015.

FRR+10. Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *EUROCRYPT*, pages 135–156, 2010.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

GGHW14. Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, pages 518–535, 2014.
GJS11. Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO*, pages 297–315, 2011.
GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.
HKL$^+$12. Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An efficient authentication protocol based on ring-lpn. In *FSE*, pages 346–365, 2012.
HL11. Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In *TCC*, pages 107–124, 2011.
JW15. Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In *TCC, Part I*, pages 451–480, 2015.
Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992.
KJJ99. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.
Koc96. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pages 104–113, 1996.
KV09. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.
LL12. Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532, 2012.
Mic00. Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.
MR04. Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.
MTVY11. Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC*, pages 89–106, 2011.
NS09. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. *IACR Cryptology ePrint Archive*, 2009:105, 2009.
NVZ13. Jesper Buus Nielsen, Daniele Venturi, and Angela Zottarel. On the connection between leakage tolerance and adaptive security. In *PKC*, pages 497–515, 2013.
NVZ14. Jesper Buus Nielsen, Daniele Venturi, and Angela Zottarel. Leakage-resilient signatures with graceful degradation. In *Public Key Cryptography*, pages 362–379, 2014.
OPV15. Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Impossibility of black-box simulation against leakage attacks. In *CRYPTO*, pages 130–149, 2015.
Pan14. Omkant Pandey. Achieving constant round leakage-resilient zero-knowledge. In *TCC*, pages 146–166, 2014.
QS01. Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
TV00. Luca Trevisan and Salil P. Vadhan. Extracting randomness from samplable distributions. In *FOCS*, pages 32–42, 2000.
Wee05. Hoeteck Wee. On round-efficient argument systems. In *ICALP*, pages 140–152, 2005.

Supplementary Material

A Relation with the definition of [DDV10] for $\lambda_0 = 0$.

We recall the indistinghuishability-based definition:

Definition 6 (Leakage-Resilient Storage [DDV10]). *For any adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and any encoding scheme $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$, consider the following experiment $\mathsf{ExpLR}^{\mathtt{ind}}_{\mathcal{A},\Sigma}(\kappa, \lambda_1)$ between $\mathcal{A}$ and a challenger:*

1. *Upon input 1^κ the adversary $\mathcal{A}_0$ outputs a pair of messages $m_0, m_1 \in \mathcal{M}$ together with a state information st;*
2. *The challenger selects a random bit b and sets $(L, R) \leftarrow\!\!\$\ \mathsf{Enc}(m_0)$;*
3. *Upon input st and oracle access to $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ and $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$ the adversary $\mathcal{A}_1$ outputs a bit b'.*
4. *If $b = b'$ then the output of the experiment is 1, else 0.*

Given $\lambda(\kappa) \in \mathbb{N}$, a (α, β)-split-coding scheme Σ is said to be (λ_1, ϵ)-LR-ind-secure if for any unbounded adversary $\mathcal{A}$:

$$\Pr[\mathsf{ExpLR}^{\mathtt{ind}}_{\mathcal{A},\Sigma}(\kappa, \lambda_1)] \leq \tfrac{1}{2} + \epsilon(\kappa)$$

We prove that the notion above is stronger than the FLR-sim-secure notion when the leakage parameter $\lambda_0 = 0$.

Theorem 6. *Let Σ be a split-coding scheme. If Σ is (λ_1, ϵ)-LR-ind-secure then it is $(0, \lambda_1, \epsilon)$-FLR-sim-secure.*

Proof. Suppose that exists adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ such that for any simulator $\mathcal{S}$:

$$\left\{\mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\not\approx}_\epsilon \left\{\mathsf{Ideal}^{0,\lambda_1}_{\mathcal{A},\mathcal{S}}(\kappa)\right\}_{\kappa\in\mathbb{N}} \tag{8}$$

Specifically, consider the simulator $\mathcal{S}'$ that does the following:

- Upon input 1^κ encodes $(L, R) \leftarrow\!\!\$\ \mathsf{Enc}(0^\alpha)$.
- Upon query f from $\mathcal{A}$ to $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ (resp. $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$) it replies with $f(L)$ (resp. $f(R)$).

By Eq. (8) there exists a ppt distinguisher $\mathcal{D}$ such that:

$$\left|\Pr[\mathcal{D}(\mathsf{Ideal}^{0,\lambda_1}_{\mathcal{A},\mathcal{S}'}(\kappa)) = 1\,] - \Pr[\mathcal{D}(\mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}(\kappa)) = 1\,]\right| > \epsilon(\kappa)$$

Consider the following adversary $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1)$ for the experiment $\mathsf{ExpLR}^{\mathtt{ind}}_{\mathcal{A}',\Sigma}(\kappa, \lambda_1)$:

- Upon input 1^κ the adversary $\mathcal{A}'_0$ runs $\mathcal{A}_0(1^\kappa)$ which returns m and a state information st, $\mathcal{A}'_0$ sets $m_0 := m, m_1 := 0^\alpha$ and outputs (m_0, m_1) and st.
- Upon input st the adversary $\mathcal{A}'_1$ runs $\mathcal{A}_1(st)$ until eventually it terminates. Let v be the view defined by $\mathcal{A}$. The adversary $\mathcal{A}'_1$ runs the distinguisher $\mathcal{D}(v)$ and outputs what $\mathcal{D}$ returns.

We show that the adversary $\mathcal{A}'$ break the LR-ind-security of Σ:

$$\begin{aligned}
&\Pr[\mathsf{ExpLR}^{\mathtt{ind}}_{\mathcal{A}',\Sigma}(\kappa, \lambda_1) = 1] \\
&= \tfrac{1}{2}\Pr[\mathcal{D}(v) = 0 \mid b = 0] + \tfrac{1}{2}\Pr[\mathcal{D}(v) = 1 \mid b = 1] \\
&= \tfrac{1}{2}(1 - \Pr[\mathcal{D}(\mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}) = 1\,] + \Pr[\mathcal{D}(\mathsf{Ideal}^{0,\lambda_1}_{\mathcal{A},\mathcal{S}'}) = 1\,]) \\
&\geqslant \tfrac{1}{2} + \left|\Pr[\mathcal{D}(\mathsf{Ideal}^{0,\lambda_1}_{\mathcal{A},\mathcal{S}'}) = 1\,] - \Pr[\mathcal{D}(\mathsf{Real}^{0,\lambda_1}_{\mathcal{A},\Sigma}) = 1\,]\right| > \tfrac{1}{2} + \epsilon(\kappa)
\end{aligned}$$

This ends the proof of the theorem.

B Proof of Theorem 2.

We give the definition of adaptive-secure PAoK from [FNV15]. Consider the following experiment $\mathtt{Exp}^{\mathsf{adp}}_{\mathsf{P}^*,\Pi}(\kappa, z)$ between a challenger and a prover P^*:

1. The challenger picks $(c, tp) \leftarrow\!\!\$\ \Pi.\mathsf{Chall}(1^\kappa)$;
2. The prover P^* upon inputs c, the auxiliary input z and randomness $r \leftarrow\!\!\$\ \{0,1\}^*$ produces an instance x such that $|x| = \kappa$ and an answer a;
3. The challenger computes $b := \mathsf{TResp}(tp, x)$. If $a = b$ then $\mathtt{Exp}^{\mathsf{adp}}_{\mathsf{P}^*,\Pi}(\kappa, z)$ outputs accept, else 0.

Similarly, let $\mathtt{Exp}^{\mathsf{adp-e}}_{\mathsf{P}^*,\mathsf{K}}(\kappa, z)$ be the following experiment:

1. The challenger picks $(c, tp) \leftarrow\!\!\$\ \Pi.\mathsf{Chall}(1^\kappa)$;
2. The prover P^* upon inputs c, the auxiliary input z and randomness $r \leftarrow\!\!\$\ \{0,1\}^*$ produces an instance x such that $|x| = \kappa$ and an answer a;
3. The challenger computes $w \leftarrow\!\!\$\ \mathsf{K}(tp, z)$. If $(x, w) \in \mathcal{R}$ then $\mathtt{Exp}^{\mathsf{adp-e}}_{\mathsf{P}^*,\mathsf{K}}(\kappa, z)$ outputs accept, else 0.

Definition 7 (Adaptive-Secure PAoK). *Let $\Pi = (\mathsf{Chall}, \mathsf{Resp}, \mathsf{TResp})$ be as specified above, and let R be an* NP *relation. Consider the properties below.*

Completeness: *There exists a negligible function μ such that:*

$$\Pr_{c,tp}\left[\forall (x,w) \in \mathcal{R} : \mathsf{TResp}(tp, x) = \mathsf{Resp}(1^\kappa, x, w, c)\right] \geqslant 1 - \mu(\kappa),$$

where the probability is taken over the outcomes of $\mathsf{Chall}(1^\kappa)$.

Knowledge soundness with error ϵ: *For all* ppt *provers* P^* *there exists a non-uniform extractor* K *and a non-zero polynomial $q(\cdot)$ such that for any auxiliary input $z \in \{0,1\}^*$ the following holds.*

$$\Pr\left[\mathtt{Exp}^{\mathsf{adp-e}}_{\mathsf{P}^*,\mathsf{K}}(\kappa, z) = \mathrm{accept}\right] \geqslant q(p(z,\kappa) - \epsilon(\kappa))\ ,$$

where $p(\kappa, z) := \Pr[\mathtt{Exp}^{\mathsf{adp}}_{\mathsf{P}^*,\Pi}(\kappa, z) = \mathrm{accept}] > \epsilon(\kappa)$.

Let ℓ be the size of the prover's answer. We call Π a adaptive-secure predictable argument of knowledge (PAoK) for $\mathcal{R}$ if Π satisfies completeness and knowledge soundness for any efficient computable function f, and moreover $\epsilon - 2^{-\ell}$ is negligible. We call it a laconic *adaptive-secure PAoK if $\ell = 1$.*

We are now ready to give the proof. We first recall the statement of the theorem:

Theorem 2. *If public-coin $\mathsf{AoK}_{1/2,\mathsf{negl}(\kappa)}(O(1), \ell_{\mathsf{AoK}}(\kappa))$ for* NP*, adaptive-secure PAoK for* NP *with answer length* 1 *and collision-resistant hash functions with output length* 1 *exist then for any $\lambda_0 \geqslant 1$ for any (α, β)-split-coding scheme Σ with $\alpha(\kappa) \geqslant 7$ and if $\lambda_1(\kappa) \geqslant (17\lambda_0(\kappa)) \cdot \ell_{\mathsf{AoK}}(\kappa)$ it holds that Σ is not (λ_0, λ_1)-FLR-sim-secure.*

Proof. We first set some necessary notation. Given a random variable x we use the notation $\bar{x}$ to refer to a possible assignment of the random variable. Let Π be in $\mathsf{AoK}_{\mathsf{negl}(\kappa),\mathsf{negl}(\kappa)}(O(1), \ell_{\mathsf{AoK}}(\kappa))$ and a public-coin argument system for NP. For concreteness let ρ be the round complexity of the Π. For any $i \in [\rho]$ let $\mathcal{P}^{\mathcal{O}}_{\mathsf{leak}}(x, y_1, \ldots, y_i; r_p)$ be a prover for Π which has oracle access to a leakage oracle $\mathcal{O}$ and which queries the oracle with the function $\mathsf{lk}(w) := \mathsf{Prove}(x, w, y_1, \ldots, y_i; r_p)$ and outputs the answer given by the oracle. Let $(\mathsf{Gen}^{\mathrm{CRH}}, \mathsf{Eval}^{\mathrm{CRH}})$ be a collision resistant hash function with output length $\ell_{\mathsf{CRH}}(\kappa)$. Let $\Pi' := (\mathsf{Chall}, \mathsf{Resp}, \mathsf{TResp})$ be a adaptive-secure PAoK with answer length 1.
Consider the adversary $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1)$ that does the following:

1. Pick a collision resistant hash function $h \leftarrow \mathsf{Gen}^{\mathrm{CRH}}(1^\kappa)$;

2. Pick $m \leftarrow\!\!\$\ \mathcal{M}$ and send it to the challenger;
3. Compute $h(m)$.

This ends the code of $\mathcal{A}'_0$, formally, $\mathcal{A}'_0(1^\kappa)$ outputs m that is forwarded to the experiment which instantiates a leakage oracle $\mathcal{O}^m_{\lambda_0}$, also $\mathcal{A}'_0(1^\kappa)$ outputs the state $st := (h, h(m))$.

4. Sample $(c, tp) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa)$ and leak from $\mathcal{O}^{\omega\|m}_{\lambda_0}$ the function which compute the response

$$f_0(\omega\|m) := \mathsf{Resp}(\ (\mathsf{Enc}(\mathsf{crs}, m; \omega), m),\ \mathsf{Enc}(\mathsf{crs}, m; \omega),\ c),$$

in a proof for the relation $\mathcal{R}^{\mathsf{st}}$ as defined below:

$$\left\{ (h_l, h_r, m), (L, R) : \begin{array}{c} h(L) = h_l \\ h(R) = h_r \\ \mathsf{Dec}(L, R) = m \end{array} \right\}.$$

Let lk_0 be the oracle answer.

5. Send the message `encode`. Leak from $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ (resp. $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$) the value $h_l = h(L)$ (resp. $h_r = h(R)$). Check if $\mathsf{TResp}(tp,\ (h_l, h_r, m)) = \mathsf{lk}_0$, abort if not.
6. For $\tau := 17\lambda_0$ many times leak from $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$ a (succinct) AoK for the relation $\mathcal{R}^{\mathsf{hash}} := \{(y, x) : h(x) = y\}$ with instance h_l. If the verification fails then abort.
The procedure is similar to the step above, the adversary runs the protocol as verifier with an instance of $\mathcal{P}^{\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}}_{\mathsf{leak}}(h_l)$, let π^L_i be the transcript produced, the adversary aborts if $\mathsf{Judge}(h_l, \pi^L_i) = 0$. Send the message `encode`.
7. For τ many times leak from $\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$ a (succinct) AoK for the relation $\mathcal{R}^{\mathsf{hash}}$ with the instance h_r. If the verification fails then abort.
As before, the adversary runs the protocol as verifier with an instance of $\mathcal{P}^{\mathcal{O}^{\boldsymbol{R}}_{\lambda_1}}_{\mathsf{leak}}(h_r)$, the adversary aborts if $\mathsf{Judge}(h_l, \pi^R_i) = 0$ where π^R_i is the transcript produced.

Consider the following randomized experiment $\mathbf{E}$:

– Pick uniformly random $m \leftarrow\!\!\$\ \mathcal{M}$ and $h \leftarrow\!\!\$\ \mathsf{Gen}^{\mathrm{CRH}}(1^\kappa)$ and forward h to the predictor.
– Instantiate an oracle $\mathcal{O}^m_{\lambda_0}$ and give the predictor access to it.

Lemma 8. $\widetilde{\mathbb{H}}_\infty(m \mid \mathbf{E}) \geqslant \alpha - \lambda_0$.

Proof. Consider the experiment $\mathbf{E}'$ which is the same as $\mathbf{E}$ except that the predictor's oracle access to $\mathcal{O}^{\omega\|m}_{\lambda_0}$ is removed. We apply Lemma 1:

$$\mathbb{H}_\infty(m \mid \mathbf{E}) \geqslant \mathbb{H}_\infty\left(m \mid \mathbf{E}'\right) - \lambda_0.$$

In the last experiment $\mathbf{E}''$ the predictor has no information about m, therefore:

$$\mathbb{H}_\infty\left(m \mid \mathbf{E}''\right) = \log|\mathcal{M}| = \alpha.$$

□

Lemma 9. *If Σ is a (λ_0, λ_1)-FLR-sim-secure then* $\widetilde{\mathbb{H}}_\infty(m|\mathbf{E}) \leqslant 6$.

Proof. Assume that Σ is an $(\lambda_0, \lambda_1, \epsilon)$-FLR-sim-secure split-coding scheme for a negligible function ϵ. Since $\mathcal{A}'$ is ppt there exists a ppt simulator $\mathcal{S}'$ such that:

$$\{\mathsf{Real}^{\lambda_0,\lambda_1}_{\mathcal{A}',\Sigma}(\kappa)\}_\kappa \overset{c}{\approx}_\epsilon \{\mathsf{Ideal}^{\lambda_0,\lambda_1}_{\mathcal{A}',\mathcal{S}'}(\kappa)\}_\kappa. \tag{9}$$

Consider the adversary $\mathcal{A}''$ that executes the same code of $\mathcal{A}'$ but, in step 5, does not check that $\mathsf{TResp}(tp,\ (h_l, h_r, m)) = \mathsf{lk}_0$. Notice that we can implement $\mathcal{A}''_1$ without providing the input m. We have that:

$$\{\mathsf{Ideal}^{\lambda_0,\lambda_1}_{\mathcal{A}',\mathcal{S}'}(\kappa)\}_\kappa \approx \{\mathsf{Ideal}^{\lambda_0,\lambda_1}_{\mathcal{A}'',\mathcal{S}'}(\kappa)\}_\kappa. \tag{10}$$

In fact, conditioned on the event $[\mathsf{TResp}(tp,\ (h_l, h_r, m)) = \mathsf{lk}_0]$ the two distributions are exactly the same. On the other hand, Eq. (9) implies that the event happens with probability $1 - \mathsf{negl}(\kappa) - \epsilon(\kappa)$.

For the sake of the proof we first build a predictor which is given $(h, h(m))$ and tries to guess m. We then use this predictor to prove the lemma. Let K be the extractor given by the knowledge soundness property of the argument of knowledge for the relation $\mathcal{R}^{\mathsf{hash}}$. Consider the following predictor $\mathcal{B}$ that takes as input $(h, h(m))$ and has oracle access to $\mathcal{O}^m_{\lambda_0}$:

1. Pick two random tapes r_a, r_s for the adversary $\mathcal{A}''_1$ and the simulator $\mathcal{S}'$ and run both of them (with the respective randomness r_a, r_s) connecting the oracle accesses of $\mathcal{A}''$ to the interfaces of $\mathcal{S}'$ and forwarding the queries of $\mathcal{S}'$ to the oracle $\mathcal{O}^m_{\lambda_0}$. The adversary $\mathcal{A}''_1$ starts by leaking an argument of knowledge for $\mathcal{R}^{\mathsf{st}}$, then it sends the message `encode` and leaks the values h_l, h_r from $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}, \mathcal{O}^{\boldsymbol{R}}_{\lambda_1}$.

2.L. The predictor tries to compute L' using the knowledge extractor K. Formally, for any $i \in [\tau]$, let $\bar{st}^L_i$ be the actual internal state of $\mathcal{S}'$ during the above run of $\mathcal{S}'$ and $\mathcal{A}''$ just before the i-th iteration of step 6 of $\mathcal{A}''$. Let $\mathcal{S}'_i$ be a copy of $\mathcal{S}'$ with internal state set to $\bar{st}^L_i$. Abusing notation, we use $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ to denote the prover which queries the interface of $\mathcal{S}'_i$ for $\mathcal{O}^{\boldsymbol{L}}_{\lambda_1}$, i.e., the message sent by $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ is computed by calling $\mathcal{S}'_i$ as if it was a leakage oracle and with the leakage functions being the functions computing the messages that the prover would send in a proof of knowledge of a preimage L' of h_l. Notice that when $\mathcal{S}'_i$ is run it might make an oracle query on its interface for $\mathcal{O}^m_{\lambda_0}$. When this happens $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ will intercept the query and will not try to simulate $\mathcal{O}^m_{\lambda_0}$ to $\mathcal{S}'_i$. Instead, whenever $\mathcal{S}'_i$ queries $\mathcal{O}^m_{\lambda_0}$ the program $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ will take the reply from $\mathcal{S}'_i$ to be some dummy string, say the all-zero string. This ensures that $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}$ makes no further leakage queries to $\mathcal{O}^m_{\lambda_0}$. The predictor runs the knowledge extractor K on the prover $\mathcal{P}^{\mathcal{S}'_i}_{\mathsf{leak}}(h_l)$, which outputs a value L' or aborts. If $h_l = h(L')$ then return L' otherwise the i-th extraction is said to abort. If all the extractions abort, the predictor aborts.

2.R. The predictor computes R' using the knowledge extractor K. The procedure is the same of step 2.L of the predictor, for notational completeness let us denote with st^R_i the internal state of $\mathcal{S}'$ just before the i-th iteration of step 7.

3. The predictor outputs $m' := \mathsf{Dec}(L', R')$ as its own guess.

We compute the probability that $\mathcal{B}$ predicts m correctly. We set up some useful notation:

- Let Ext_L (resp. Ext_R) be the event that K successfully extracts a value L' (resp. R').
- Let CohSt be the event $\{h(\mathsf{Dec}(L', R') = m\}$.

Recall that $m' := \mathsf{Dec}(L', R')$ is the guess of $\mathcal{B}$. We can easily derive that:

$$\Pr\left[m' = m\right] = \Pr\left[\mathsf{Ext}_L \wedge \mathsf{Ext}_R \wedge \mathsf{CohSt}\right] \tag{11}$$

Claim 6 $\Pr[\mathsf{Ext}_R \wedge \mathsf{Ext}_L] \geqslant \frac{1}{16} - \mathsf{negl}(\kappa)$.

The proof proceeds similar to the proof of the correspondent lemmas in proof of Theorem 1, therefore it is omitted.

Claim 7 $\Pr[\mathsf{CohSt} \,|\mathsf{Ext}_L \wedge \mathsf{Ext}_R] \geqslant \frac{1}{2} - \mathsf{negl}(\kappa)$.

Proof. We reduce to the collision resistance property of h and the knowledge soundness of the PAoK Π', Suppose that

$$\Pr[\mathsf{Dec}(L', R') \neq m \mid \mathsf{Ext}_L \wedge \mathsf{Ext}_R] \geqslant 1/\mathsf{poly}(\kappa)$$

Consider the following collision finder adversary $\mathcal{B}_{coll}(h)$:

- Sample uniformly random $m \leftarrow\!\!\$\ \mathcal{M}$ and random $h \leftarrow\!\!\$\ \mathsf{Gen}^{\mathrm{CRH}}(1^\kappa)$;
- Run an instance of the predictor $\mathcal{B}^{\mathcal{O}^m_{\lambda_0}}(h)$, the predictor needs oracle access to $\mathcal{O}^m_{\lambda_0}$ which can be simulated by $\mathcal{B}_{coll}(h)$.
- Consider the following adversary P^* for the experiment $\mathtt{Exp}^{\mathsf{adp}}_{P^*,\Pi'}$:
  1. Let r_a, r_s be the same randomness picked by $\mathcal{B}$ in its step 1;
  2. Simulate an execution of $\mathcal{A}'_1(m, h; r_a)$ and $\mathcal{S}'(1^\kappa; r_s)$ and break them just before the adversary leaks an argument of knowledge for $\mathcal{R}^{\mathsf{st}}$;
  3. Continue the execution of $\mathcal{A}'_1(h; r_a)$ and $\mathcal{S}'(1^\kappa; r_s)$ until $\mathcal{A}'_1$ has leaked the value h_l and h_r;
  4. Output (h_l, h_r, m) as instance and lk_0 as answer.

  Run the knowledge extractor $\mathsf{K}'_{\mathsf{st}}$ given by the definition of adaptive-secure PAoK for the adversary P^*, specifically we run $\mathsf{K}'_{\mathsf{st}}$ on the same randomness and auxiliary input of P^*. Let L'', R'' the witness output by the extractor.
- If $L' \neq L''$ output (L', L'') else (R', R'').

It is easy to check that $\mathcal{B}_{coll}$ simulates perfectly the randomized experiment $\mathbf{E}$ therefore:

$$\begin{aligned}
&\Pr[\mathsf{Dec}(L', R') \neq m] \geqslant \\
&\quad \geqslant \Pr[\mathsf{Dec}(L', R') \neq m \mid \mathsf{Ext}_L \wedge \mathsf{Ext}_R] \Pr[\mathsf{Ext}_L \wedge \mathsf{Ext}_R] \geqslant \\
&\quad \geqslant 1/\mathsf{poly}(\kappa) \cdot (\tfrac{1}{16} - \mathsf{negl}(\kappa))
\end{aligned}$$

On the other hand, the extractor $\mathsf{K}'_{\mathsf{st}}$ succeeds with probability at least $q(1-\mathsf{negl}(\kappa)-\frac{1}{2})-\mathsf{negl}(\kappa)$. Notice that the definition of PAoK implies that the probability of $\mathsf{K}'_{\mathsf{st}}$ winning in $\mathtt{Exp}^{\mathsf{adp}-\mathsf{e}}$ and P^* winning in $\mathtt{Exp}^{\mathsf{adp}}$ are related, however, in general the instance for which the two algorithm wins the games can be different. This is not a problem for us, since P^* wins with overwhelming probability, therefore the ratio of the intersection of the instances for which both $\mathsf{K}'_{\mathsf{st}}$ and P^* wins is negligible close to the set of instance for which $\mathsf{K}'_{\mathsf{st}}$ wins, this explain why we lose an extra negl factor.

Therefore, L'' and R'' are such that $h(L'') = h(L')$, $h(R'') = h(R')$ and $\mathsf{Dec}(L'', R'') = m$. The latter implies $\mathsf{Dec}(L', R') \neq \mathsf{Dec}(L'', R'')$ which implies that either $L'' \neq L'$ or $R'' \neq R'$. Lastly, notice that $\mathcal{B}_{coll}$ is an expected polynomial time algorithm, however we can make it polynomial time by aborting if the number of steps exceed some fixed polynomial. By setting the polynomial big enough the probability of $\mathcal{B}_{coll}$ finding a collision is still noticeable.

Summarizing, we have:

$$\begin{aligned}
\Pr[m' = m] = \Pr\left[\mathsf{Ext}_L \wedge \mathsf{Ext}_R \wedge \mathsf{CohSt}\right] \geqslant \\
\geqslant (\tfrac{1}{16} - \mathsf{negl}(\kappa)) \cdot (\tfrac{1}{2} - \mathsf{negl}(\kappa)) \geqslant \tfrac{1}{64}
\end{aligned}$$

which implies the statement of the lemma.

We conclude the proof of the theorem noticing that if Σ is (λ_0, λ_1)-FLR-sim-secure split-coding scheme by the parameter given in the statement of the theorem we have that Lemma 8 and Lemma 9 are in contraction. □