

# Fully Leakage-Resilient Codes

Antonio Faonio<sup>1</sup> and Jesper Buus Nielsen<sup>1</sup>

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  $\lambda_0$  bits from the encoding process, namely, 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 codeword can be simulated given just  $\lambda_0$  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. Unfortunately, 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 when the message length is at least linear in the security parameter. We can extend the impossibility result to FLRCs with 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  $\text{NC}^0$ -bounded leakage from the randomness and arbitrary poly-time leakage from the parts of the codeword the inner-product construction proposed by Davi *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* where the leakage on the encoding comes from a fixed leakage family of small cardinality. In particular, this compiler applies to the split-state model but also to 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 Davi *et al.* [DDV10]) they have found many applications either by their own or as building blocks 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<sup>+</sup>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 Jafarholi 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 resilient 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

---

\* Both the authors are supported by European Research Council Starting Grant 279447.

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 is 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 leakage from the encoding process is usually called *fully leakage resilience*.

## 1.1 Our Contributions

We generalize the notion of LRC to the setting of fully leakage resilience. Roughly speaking, a fully leakage-resilient code (FLRC) hides 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 of fully leakage-resilient codes. The definition postulates that for any adversary leaking  $\lambda_0$  bit from the encoding process and  $\lambda_1$  bits from the codewords there exists a simulator which provide a view that is indistinguishable. Our definition is, in some sense, the minimal one suitable for the fully leakage resilience setting. As a sanity check, we show that 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 are at least linear in the security parameter and under the sole assumption that collision-resistant hash functions exist. We can generalize the impossibility result to the case of constant-length messages under the much stronger assumption that differing-input obfuscation exists (see [ABG<sup>+</sup>13,BCP14]).
3. We provide two feasibility results for weaker models. 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. We provide a weaker definition of fully-leakage resilience in the common reference string model and show a compiler from any LRC to a fully leakage resilient code in this weaker model for any fixed leakage-from-the-encoding 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  $\omega$ . After this, the adversary gets access to leakage from the codewords. We show an easy attack on this definition. 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 presence of leakage on the randomness, the same security

guarantees as the (standard) leakage resilient definition. However, the ability of leaking before and after the challenge generation, as shown 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  $\lambda_0$  bits from the encoding process and  $\lambda_1$  bits from the codewords there exists a simulator which provide a view that is indistinguishable. Namely, the adversary chooses one input message and forwards it to the challenger of 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, given only leakage oracle access to the message. It is not hard to see that, without any help, the task would be impossible. Since an adversary could in particular leak bits of the input message, if the input message is randomly chosen the simulator trivially cannot provide an indistinguishable view. The simulator can leak up to  $\lambda_0(1 + \gamma)$  bits from the message for a “slack parameter”  $\gamma \geq 0$ . The idea is that some information about the encoded message can unavoidably leak from the encoding process, however the amount of information about the message, even after have seen the leakage on the codeword, does not exceed to much the bound on the leakage on the encoding process. The slack parameter is often considered as a reasonable weakening of the model in the context of fully leakage resilience (see for example [GJS11,FNV14,HL11,NVZ14]), we include it in our model to make the impossibility results stronger. For the feasibility results we will instead ignore it.

**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 function of the data stored in one of the parts.

Here we give the intuition behind the attacker. For simplicity let us set the slack parameter  $\gamma$  equal to 0. 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  $L$  and  $R$  (the left part and the right part) and leaks two hash values  $h_l = h(L)$  and  $h_r = h(R)$ . Then we use a succinct argument of knowledge systems to also leak an argument of knowledge of pre-images  $L$  and  $R$  of  $h_l$  and  $h_r$  for which it holds that  $(L, R)$  decodes to  $m$ . Let  $\lambda_0$  be the length of the two hashes and the succinct argument. After this the message can be encoded. 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  $\lambda_1$  on  $L$  and  $R$  leak  $17\lambda_0$  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  $\lambda_0$  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 to 1. Since the simulator has access to only  $\lambda_0$  bits of leakage on  $m$  it follows that for one of the  $17\lambda_0$  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 close to  $\frac{1}{4}$ . By collision resistance and soundness of the first 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 negligibly close to  $\frac{1}{16}$  while using only  $\lambda_0$  bits of leakage on  $m$ . If  $m$  is uniformly random and just  $\lambda_0 + 6$  bits long, this is a contradiction, as there is 6 bits on min-entropy on  $m$  after leaking  $\lambda_0$  bits on  $m$  and hence  $m$  cannot be guessed with probability better than  $2^{-6}$ .

Similar proof techniques have been used already by Nielsen *et al.* [NVZ13] to prove a connection between leakage resilience and adaptive security and recently by Ostrovsky *et al.* [OPV15] to prove an impossibility result for certain flavors of leakage-resilient zero-knowledge proof systems. The way we apply this type of argument here is novel. It is in particular a new idea to use many arguments of knowledge in sequence to sufficiently restrict the simulator's ability to leak from its leakage oracle in one of the proofs.

The proposed attack needs to leak at least a collision-resistant hash function of the codeword. On the other hand, the definition of FLRC makes sense only when the leakage parameter  $\lambda_0$  is strictly smaller than the size of the message, therefore 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 Predictable ZAP (PZAP) recently proposed by Faonio *et al.* [FNV15]. A PZAP 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. PZAPs are shown 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 an evidence that constant-length FLRCs are hard to construct as such a code would not only make diO implausible, but it would prove diO *impossible* under the only assumption that collision-resistant hash functions exist.

**The feasibility results.** The ability to leak collision-resistant hash functions of the randomness is necessary for the impossibility result. The natural question that follows is: If we restrict the leakage class so that no collision-resistant 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  $\text{NC}^0$  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  $\text{NC}^0$  function that maps the randomness to the codeword) are fully leakage resilient. The intuition is that  $\text{NC}^0$  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-resistant hash functions cannot be computed by  $\text{NC}^0$  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 it is hard to construct FLRC even for bounded classes of leakage.

Another important property that we exploit in the impossibility result is that, given access to the leakage oracle on the randomness, we can compute the codeword. A second path to avoid the impossibility results is to consider weaker models of security where this is not permitted. 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 weaker form of FLRC.

We remark that the (full-fledged) definition of FLRC already assumes a CRS (this to include in our model the result of Liu and Lysyanskaya [LL12]). The difference between the two definitions is that, in the weaker, the adversary first leaks from the encoding process and then receives the CRS, in this way the leakage functions on the encoding process cannot depend on the common reference string, and therefore the codeword cannot be computed using the leakage oracle.

The key idea of our compiler is to apply a deterministic extractor on the randomness before using it inside the encoding machine. Unfortunately, to apply our 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). 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 a deterministic 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 considered (see for example [ADW09,DP08,FRR<sup>+</sup>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]).

In [FMNV15] the authors present a RAM model of computation where a CPU is connected to some constant number of memories, paralleling the split-state model that we use here. The memories and buses are assumed to be leaky, but the CPU is assumed to be leakage free. Besides leakage, the paper also shows how to handle tampering, like moving around codewords in the memories. They show how to use a leakage-resilient and tamper-resilient code to securely compute on this platform. In each step the CPU will read from the disks a number of codewords, decode these, do a computation on the plaintext, re-encode the results and write the codewords back in the memories. One should wonder if it is possible to get a similar result for the more realistic model where there is a little leakage from the CPU? It is clear that if the CPU can leak, then it can also leak from the plaintexts it is working on. This can be handled by having the computation that is done on the plaintexts being leakage resilient in itself. The challenging part is then to show that the leakage from the CPU during re-encoding of the results to be stored in the memories can be simulated given just a little leakage on the results themselves. This would in particular require that the code is fully leakage-resilient in the sense we define in this paper. Our negative results therefore do not bode well for this proof strategy. On the other hand, our positive results open up the possibility of tolerating some simple leakage from the CPU or getting a result for weaker models, like the random oracle model. Note, however, that the code would have to be tamper-resilient in addition to being fully leakage resilient, so there still seem to be significant obstacles towards proving such a result.

## 1.3 Roadmap

In Section 2 we introduce the necessary notation for probability and cryptographic tools we make use of. In Section 3 we provide the simulation-based definition for Fully Leakage-Resilient Codes while in Section 3.1 we introduce the weaker variant. In Section 4 we state and prove the main impossibility result for linear-size message spaces. In Section 4.1 we sketch the second impossibility result for constant-size message space, the full proof is deferred in Appendix B. In Section 5 we provide the two feasibility

results, specifically, in Section 5.1 we give a FLR code for the class  $\text{NC}^0$  and in Section 5.2 we give a compiler from (standard) Leakage-Resilient Codes to Weak Fully Leakage-Resilient Codes.

## 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  $[a] = \{1, 2, \dots, a\}$ . 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  $\text{poly}(|x|)$  steps.

Let  $\kappa$  be a security parameter. A function  $\text{negl}$  is called *negligible* in  $\kappa$  (or simply negligible) if it vanishes faster than the inverse of any polynomial in  $\kappa$ . 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  $\mathcal{X} \stackrel{c}{\approx}_{\epsilon} \mathcal{Y}$ , 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  $\kappa$ .

We simply write  $\mathcal{X} \stackrel{c}{\approx} \mathcal{Y}$  when there exists a negligible function  $\epsilon$  such that  $\mathcal{X} \stackrel{c}{\approx}_{\epsilon} \mathcal{Y}$ . 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:

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

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:

$$\tilde{\mathbb{H}}_{\infty}(X | \mathbf{E}) = \max_{\mathcal{B}} (-\log \Pr[\mathcal{B}(\cdot)^{\mathbf{E}} = X]),$$

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  $\tilde{\mathbb{H}}_{\infty}(X|Y)$  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  $\mathcal{O}_{\lambda}^X(\cdot)$  be *the leakage oracle* that accepts as input functions  $f_1, f_2, \dots$  defined as circuits and outputs  $f_1(X), f_2(X), \dots$  under the restriction that  $\sum_i |f_i(X)| \leq \lambda$ .

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

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

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

**Lemma 2.** Let  $t \geq 4$  be an even integer. Suppose  $X_1, \dots, X_n$  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  $A > 0$ ,  $\Pr[|X - \mu| \geq A] \leq 8 \left( \frac{t\mu + t^2}{A^2} \right)^{t/2}$ .

## 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]. A public-coin argument system  $(P(w), V)(x)$  with round complexity  $\rho(\kappa)$  is fully described by the tuple of ppt algorithms (Prove, Judge) where:

- $V$  on input  $x$  samples uniformly random strings  $y_1, \dots, y_{\rho(\kappa)} \leftarrow_{\$} \{0, 1\}^{\kappa}$ ,  $P$  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  $x_i := \text{Prove}(x, w, y_1, \dots, y_i; r_P)$ .
- The verifier  $V$  executes  $j := \text{Judge}(x, y_1, \dots, y_{\rho(\kappa)}, x_1, \dots, x_{\rho(\kappa)})$  and accepts if  $j = 1$ .

**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  $\mathcal{L} = \mathcal{L}_{\mathcal{R}} := \{x \mid \exists w : (x, w) \in \mathcal{R}\}$ , and functions  $\nu, s : \mathbb{N} \rightarrow [0, 1]$  such that  $1 - \nu(\kappa) > s(\kappa) + 1/\text{poly}(\kappa)$  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

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

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  $\text{AoK}_{\nu, s}(\rho(\kappa), \lambda(\kappa))$  to denote interactive argument on knowledge systems with completeness error  $\nu$ , 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. We say  $(P, V)$  is *succinct* if  $\lambda(\kappa)$  is poly-logarithmic in the length of the witness and the statement being proven.

*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<sup>+</sup>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<sup>+</sup>13]), is an argument of knowledge system with two messages only and where the prover message can be of constant size. Specifically, we make use of Predictable ZAPs, namely two-message adaptive-sound arguments of knowledge. Such protocols are fully specified by a tuple of three ppt algorithms  $\Pi = (\text{Chall}, \text{Resp}, \text{Predict})$  as described below:

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

Roughly speaking, the adaptive knowledge-soundness of a Predictable ZAP postulates that the malicious prover can select instances adaptively as function of the challenge  $c$ . 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 give the security definition in Appendix B.

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

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

$$\Pr_{h \leftarrow_{\$} \text{Gen}^{\text{CRH}}(1^\kappa)} [h(x_0) = h(x_1) \wedge x_0 \neq x_1 \mid (x_0, x_1) := \mathcal{B}_{\text{coll}}(h)] < \text{negl}(\kappa).$$

For simplicity we consider the model of non-uniform polynomial time adversaries. Note, however, 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 Codes. 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 adapted to hold for the more general  $k$ -split model (see Remark 1). LRCs of [DDV10, FMNV14, JW15] in the 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 weaker definition of security in Section 3.1. The syntax given allows the scheme to depend on a common reference string to include the scheme of [LL12].

An  $(\alpha, \beta)$ -split-coding scheme is a tuple  $\Sigma = (\text{Gen}, \text{Enc}, \text{Dec})$  of ppt algorithms with the following syntax:

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



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

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

Given an  $(\alpha, \beta)$ -split-coding scheme  $\Sigma$ , for any  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$  and any function  $\lambda_0, \lambda_1$  let  $\text{Real}_{\mathcal{A}, \Sigma}^{\lambda_0, \lambda_1}(\kappa)$  be the following experiment:

**Sampling Phase.** The experiment runs the adversary  $\mathcal{A}_0$  on input  $\text{crs} \leftarrow_{\$} \text{Gen}(1^\kappa)$  and randomness  $r_A \leftarrow_{\$} \{0, 1\}^{p(\kappa)}$  for a polynomial  $p$  that bounds the running time of  $\mathcal{A}_0$ . The adversary outputs a message  $m \in \mathcal{M}_\kappa$  and a state value  $st$ . The experiment samples  $\omega \leftarrow_{\$} \mathcal{R}_\kappa$  and instantiates a leakage oracle  $\mathcal{O}_{\lambda_0}^{\omega \| m}$ .

**Encoding Phase.** The experiment runs the adversary  $\mathcal{A}_1$  on input  $st$  and  $\text{crs}$ . Moreover, the experiment sets an index  $i := 0$ .

- Upon query  $(\text{rand}, f)$  from the adversary where  $f$  is the description of a function with domain  $\mathcal{R}_\kappa \times \mathcal{M}_\kappa$ , the experiment sets  $i := i + 1$ , computes  $\text{lk}_\omega^i := \mathcal{O}_{\lambda_0}^{\omega \| m}(f)$  and returns the value to the adversary.

- Eventually, the adversary notifies the experiment by sending the message `encode`.

The message is encoded, namely the experiment defines  $(L, R) := \text{Enc}(\text{crs}, m; \omega)$  and instantiates the oracles  $\mathcal{O}_{\lambda_1}^L, \mathcal{O}_{\lambda_1}^R$ . Moreover, the experiment sets two indexes  $l := 0$  and  $r := 0$ .

- Upon query  $(L, f)$  from the adversary where  $f$  is the description of a function with domain  $\mathcal{C}_\kappa$ , the experiment sets  $l := l + 1$ , computes  $\text{lk}_L^l := \mathcal{O}_{\lambda_1}^L(f)$  and returns the value to the adversary.

- Upon query  $(R, f)$  from the adversary where  $f$  is the description of a function with domain  $\mathcal{C}_\kappa$ , the experiment sets  $r := r + 1$ , computes  $\text{lk}_R^r := \mathcal{O}_{\lambda_1}^R(f)$  and returns the value to the adversary.

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

$$\text{Real}_{\mathcal{A}, \Sigma}^{\lambda_0, \lambda_1} := \left( \begin{array}{l} r_A, \text{crs}, \\ \text{lk}_\omega := (\text{lk}_\omega^1, \text{lk}_\omega^2, \dots, \text{lk}_\omega^i), \\ \text{lk}_L := (\text{lk}_L^1, \text{lk}_L^2, \dots, \text{lk}_L^l), \\ \text{lk}_R := (\text{lk}_R^1, \text{lk}_R^2, \dots, \text{lk}_R^r) \end{array} \right),$$

Given an adversary  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ , a simulator  $\mathcal{S}$  and a slack parameter  $\gamma(\kappa)$  such that  $0 \leq \gamma(\kappa) < \frac{\alpha(\kappa)}{\lambda_0(\kappa)} - 1$  let  $\text{Ideal}_{\mathcal{A}, \mathcal{S}, \gamma}^{\lambda_0, \lambda_1}(\kappa)$  be the following experiment:

**Sampling Phase.** The experiment runs the adversary  $\mathcal{A}_0$  on input  $\text{crs} \leftarrow_{\$} \text{Gen}(1^\kappa)$  and randomness  $r_A \leftarrow_{\$} \{0, 1\}^{p(\kappa)}$  for a polynomial  $p$  that bounds the running time of  $\mathcal{A}_0$ . The adversary outputs a message  $m \in \mathcal{M}_\kappa$  and a state value  $st$ . The experiment instantiates an oracle  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$ .

**Encoding Phase.** The experiment runs the adversary  $\mathcal{A}_1$  on input  $st$  and  $\text{crs}$ , and the simulator  $\mathcal{S}$  on input  $\text{crs}$ .

- Upon query  $(X, f)$  from the adversary where  $X \in \{\text{rand}, L, R\}$  the experiment forwards the query to the simulator  $\mathcal{S}$  which returns an answer to the adversary.

- Upon query  $(\text{msg}, f)$  from the simulator the experiment computes  $\text{lk}_m := \mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m(f)$  and returns an answer to the simulator.

As we did with  $\text{Real}_{\mathcal{A}, \Sigma}^{\lambda_0, \lambda_1}$  we denote with  $\text{Ideal}_{\mathcal{A}, \mathcal{S}, \gamma}^{\lambda_0, \lambda_1}$  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 “simulated” components of the ideal experiment with a tilde, namely:

$$\text{Ideal}_{\mathcal{A}, \mathcal{S}, \gamma}^{\lambda_0, \lambda_1} = (r_A, \text{crs}, \tilde{\text{lk}}_\omega, \tilde{\text{lk}}_L, \tilde{\text{lk}}_R)$$

Given a class of leakage functions  $\Lambda$  we say that an adversary is  $\Lambda$ -bounded if it submits only queries  $(\text{rand}, f)$  where the function  $f \in \Lambda$ .

**Definition 3 (Simulation-based  $\Lambda$ -fully leakage resilient code).** An  $(\alpha, \beta)$ -split-coding scheme is said to be  $(\Lambda, \lambda_0, \lambda_1, \epsilon)$ -FLR-sim-secure with slack parameter  $0 \leq \gamma < \alpha/\lambda_0 - 1$  if for any ppt adversary  $\mathcal{A}$  that is  $\Lambda$ -bounded there exists a ppt simulator  $\mathcal{S}$ :

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

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

**Definition 4 (Simulation-based fully leakage resilient code).** An  $(\alpha, \beta)$ -split-coding scheme is said to be  $(\lambda_0, \lambda_1, \epsilon)$ -FLR-sim-secure with slack parameter  $\gamma$  if it is  $(\mathbb{P}_{/\text{poly}}, \lambda_0, \lambda_1, \epsilon)$ -FLR-sim-secure with slack parameter  $\gamma$ . We simply say that a split-coding scheme is  $(\lambda_0, \lambda_1)$ -FLR-sim-secure if there exists a negligible function  $\text{negl}$  and a constant  $\gamma < \alpha/\lambda_0 - 1$  such that the scheme is  $(\lambda_0, \lambda_1, \text{negl})$ -FLR-sim-secure with slack parameter  $\gamma$ .

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

### 3.1 Weak Fully-Leakage Resilient Codes

Given a  $(\alpha, \beta)$ -split-coding scheme  $\Sigma$ , for any  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$  and any function  $\lambda_0, \lambda_1$  let  $\text{wReal}_{\mathcal{A}, \Sigma}^{\lambda_0, \lambda_1}(\kappa)$  be the following experiment:

**Sampling Phase.** The experiment samples  $\omega \leftarrow_{\$} \mathcal{R}$ , instantiates an oracle  $\mathcal{O}_{\lambda_0}^{\omega}$  and runs the adversary  $\mathcal{A}_0$  on inputs  $1^{\kappa}$ . Moreover, the experiment instantiates an index  $i := 0$ .

- Upon query  $(\text{rand}, f)$  from the adversary where  $f$  is the description of a function with domain  $\mathcal{R}_{\kappa} \times \mathcal{M}_{\kappa}$ , the experiment sets  $i := i + 1$ , computes  $\text{lk}_{\omega}^i := \mathcal{O}_{\lambda_0}^{\omega}(f)$  and returns the value to the adversary.
- Eventually, the adversary notifies the experiment by sending  $\text{getCRS}$ . The experiment replies with  $\text{crs} \leftarrow_{\$} \text{Gen}(1^{\kappa})$  and turns off the oracle  $\mathcal{O}_{\lambda_0}^{\omega}$ .

The adversary  $\mathcal{A}_0$  outputs a message  $m \in \mathcal{M}_{\kappa}$  and a state value  $st$ .

**Encoding Phase.** The message is encoded, namely the experiment defines  $(L, R) := \text{Enc}(\text{crs}, m; \omega)$  and instantiates the oracles  $\mathcal{O}_{\lambda_1}^L, \mathcal{O}_{\lambda_1}^R$ . Moreover, the experiment sets two indexes  $l := 0$  and  $r := 0$ .

- Upon query  $(L, f)$  from the adversary where  $f$  is the description of a function with domain  $\mathcal{C}_{\kappa}$ , the experiment sets  $l := l + 1$ , computes  $\text{lk}_L^l := \mathcal{O}_{\lambda_1}^L(f)$  and returns the value to the adversary.
- Upon query  $(R, f)$  from the adversary where  $f$  is the description of a function with domain  $\mathcal{C}_{\kappa}$ , the experiment sets  $r := r + 1$ , computes  $\text{lk}_R^r := \mathcal{O}_{\lambda_1}^R(f)$  and returns the value to the adversary.

Notice that  $\text{crs}$  is still included in the random variable  $\text{wReal}_{\mathcal{A}, \Sigma}^{\lambda_0, \lambda_1}$ , however the leakage functions of the randomness cannot be chosen as function of the common reference string. We recall that for our feasibility results we will ignore the slack parameter.

Similarly, for any  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$  and any  $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$  let  $\text{wIdeal}_{\mathcal{A}, \mathcal{S}}^{\lambda_0, \lambda_1}(\kappa)$  be the following experiment:

**Sampling Phase.** The experiment samples  $\text{crs} \leftarrow_{\$} \text{Gen}(1^{\kappa})$ , runs the adversary  $\mathcal{A}_0$  on input  $1^{\kappa}$  and the simulator  $\mathcal{S}$  on input  $\text{crs}$ .

- Upon query  $(\text{rand}, f)$  from the adversary the experiment forwards the query to the simulator  $\mathcal{S}$  which returns an answer to the adversary.
- Eventually, the adversary notifies the experiment sending  $\text{getCRS}$ . The experiment replies with  $\text{crs}$ .

The adversary  $\mathcal{A}_0$  outputs a message  $m \in \mathcal{M}_{\kappa}$  and a state value  $st$ . The simulator  $\mathcal{S}_0$  outputs a state  $st_{\mathcal{S}}$ .

**Encoding Phase.** The experiment instantiates the oracle  $\mathcal{O}_{\lambda_0}^m$  and runs the simulator  $\mathcal{S}_1$  on inputs  $\text{crs}$  and  $st_{\mathcal{S}}$  and oracle access to  $\mathcal{O}_{\lambda_0}^m$ .

- Upon query  $(X, f)$  from the adversary where  $X \in \{\text{L}, \text{R}\}$  the experiment forwards the query to the simulator  $\mathcal{S}$  which returns an answer to the adversary.

**Definition 5 (Simulation-based  $\Lambda$ -weak-fully leakage resilient code).** A  $(\alpha, \beta)$ -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  $\Lambda$ -bounded there exists a ppt simulator  $\mathcal{S}$ :

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

## 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  $\Lambda = \text{P}_{/\text{poly}}$ . We prove that  $(\alpha, \beta)$ -split-coding schemes that are  $(\lambda_0, \lambda_1)$ -FLR-sim-secure don't exist for many interesting parameters of  $\alpha, \beta, \lambda_0$  and  $\lambda_1$ . 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  $\text{AoK}_{1/2, \text{negl}(\kappa)}(O(1), \ell_{\text{AoK}}(\kappa))$  for NP and collision-resistant hash functions with output length  $\ell_{\text{CRH}}(\kappa)$  exist then for any  $\lambda_0 \geq \ell_{\text{AoK}}(\kappa) + 2 \cdot \ell_{\text{CRH}}(\kappa)$  for any  $\gamma \geq 0$  and for any  $(\alpha, \beta)$ -split-coding scheme  $\Sigma$  with  $\alpha(\kappa) \geq \lambda_0(\kappa) \cdot (1 + \gamma) + \ell_{\text{CRH}}(\kappa) + 7$  and if  $\lambda_1(\kappa) \geq 17\lambda_0(\kappa) \cdot (1 + \gamma) \cdot \ell_{\text{AoK}}(\kappa)$  then  $\Sigma$  is not  $(\lambda_0, \lambda_1)$ -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  $(\text{Gen}^{\text{CRH}}, \text{Eval}^{\text{CRH}})$  be a collision resistant hash function with output length  $\ell_{\text{CRH}}(\kappa)$ .

*Leakage-aided Prover.* Let  $\Pi = (\text{Prove}, \text{Judge})$  be in  $\text{AoK}_{1/2, \text{negl}(\kappa)}(O(1), \ell_{\text{AoK}}(\kappa))$  and a public-coin argument system for NP. For concreteness let  $\rho$  be the round complexity of the  $\Pi$ . We say that an attacker leaks an argument of knowledge for  $x \in \mathcal{L}_{\mathcal{R}}$  from  $X \in \{\text{rand}, \text{L}, \text{R}\}$  if the attacker proceeds with the following sequence of instructions and leakage-oracle queries:

- Let  $r_p$  be a random string long enough to specify all random choices done by the prover of  $\Pi$ . For  $j \in [\rho]$  do the following:
  1. Sample a random string  $y_j \leftarrow_{\$} \{0, 1\}^{\kappa}$ ;
  2. Send the query  $(X, \text{Prove}(x, \cdot, y_1, \dots, y_j; r_p))$  and let  $z_j$  be the answer to such query.
- Let  $\pi := y_1, \dots, y_p, z_1, \dots, z_p$  be the leaked transcript, compute the value  $j := \text{Judge}(x, \pi)$ , if  $j = 1$  we say that the leaked argument of knowledge is accepting.

Consider the adversary  $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1)$  that does the following:

1. Pick a collision resistant hash function  $h \leftarrow \text{Gen}^{\text{CRH}}(1^{\kappa})$ ;
2. Pick  $m \leftarrow_{\$} \mathcal{M}_{\kappa}$  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  $\mathcal{O}_{\lambda_0, (1+\gamma)}^m$ , also  $\mathcal{A}'_0(1^{\kappa})$  outputs the state  $st := (h, h(m))$ . Here starts the code of  $\mathcal{A}'_1(h, h(m))$ :

4. **Leak Hashed Values.** Define the following function:

$$f_0(\omega \| m) := (h(L), h(R)) \text{ where } L, R = \text{Enc}(\text{crs}, m; \omega);$$

Send the query  $(\text{rand}, f_0)$ . Let  $(h_l, h_r)$  be the answer to the query.

5. **Leak Argument of Knowledge of Consistency.** Consider the following relation:

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

Leak an argument of knowledge for  $(\text{crs}, h_l, h_r, h(m)) \in \mathcal{L}_{\mathcal{R}^{\text{st}}}$  from  $\text{rand}$ . Notice that a witness for the instance can be defined as function of  $(\omega \| m)$ . If the leaked argument is not accepting then abort. Let  $\pi_0$  be the leaked transcript.

6. Send the message `encode`.

7. **Leak Arguments of Knowledge of the Left part.** Consider the following relation:

$$\mathcal{R}^{\text{hash}} := \{(y, x) : h(x) = y\}$$

Let  $\tau := 17\lambda_0 \cdot (1 + \gamma)$ , for all  $i \in [\tau]$  leak an argument of knowledge for  $h_l \in \mathcal{L}_{\mathcal{R}^{\text{hash}}}$  from L. If the leaked argument is not accepting then abort. Let  $\pi_i^L$  be the leaked transcript.

8. **Leak Arguments of Knowledge of the Right part.** For all  $i \in [\tau]$  leak an argument of knowledge for  $h_r \in \mathcal{L}_{\mathcal{R}^{\text{hash}}}$  from R. If the leaked argument is not accepting then abort. Let  $\pi_i^R$  be the leaked transcript.

Consider the following randomized experiment **E**:

- Pick uniformly random  $m \leftarrow_{\$} \mathcal{M}_{\kappa}$  and  $h \leftarrow_{\$} \text{Gen}^{\text{CRH}}(1^{\kappa})$  and set  $st = (h, h(m))$  and forward to the predictor the state  $st$ .
- Instantiate an oracle  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$  and give the predictor access to it.

**Lemma 3.**  $\tilde{\mathbb{H}}_{\infty}(m \mid \mathbf{E}) \geq \alpha - \ell_{\text{CRH}} - \lambda_0 \cdot (1 + \gamma)$ .

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

$$\tilde{\mathbb{H}}_{\infty}(m \mid \mathbf{E}) \geq \tilde{\mathbb{H}}_{\infty}(m \mid \mathbf{E}') - \ell_{\text{CRH}}.$$

Consider the experiment **E''** which is the same as **E'** except that the predictor's oracle access to  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$  is removed. We apply Lemma 1:

$$\tilde{\mathbb{H}}_{\infty}(m \mid \mathbf{E}') \geq \tilde{\mathbb{H}}_{\infty}(m \mid \mathbf{E}'') - \lambda_0 \cdot (1 + \gamma).$$

In the last experiment **E''** the predictor has no information about  $m$  and moreover  $h$  is independently chosen with respect to  $m$ , therefore:

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

□

**Lemma 4.** *If  $\Sigma$  is  $(\lambda_0, \lambda_1)$ -FLR-sim-secure then  $\tilde{\mathbb{H}}_{\infty}(m \mid \mathbf{E}) \leq 6$ .*

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

$$\{\text{Real}_{\mathcal{A}', \Sigma}^{\lambda_0, \lambda_1}(\kappa)\}_{\kappa} \stackrel{c}{\approx}_{\epsilon(\kappa)} \{\text{Ideal}_{\mathcal{A}', \mathcal{S}', \gamma}^{\lambda_0, \lambda_1}(\kappa)\}_{\kappa}. \quad (1)$$

For the sake of the proof we first build a predictor which tries to guess  $m$ . We then use this predictor to prove the lemma. Let  $\mathbb{K}$  be the extractor given by the knowledge soundness property of the argument of knowledge for the relation  $\mathcal{R}^{\text{hash}}$ . Consider the following predictor  $\mathcal{B}$  that takes as input  $(h, h(m))$  and has oracle access to  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$ :

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  $r_a, r_s$ ) forwarding all the queries from  $\mathcal{A}'_1$  to  $\mathcal{S}'$  and from  $\mathcal{S}'$  to  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$ . (The adversary  $\mathcal{A}'_1$  starts by leaking the values  $h_l, h_r$  and an argument of knowledge for  $(h_l, h_r) \in \mathcal{L}_{\mathcal{R}^{\text{st}}}$ . Eventually the adversary sends the message encode.)
- 2.L. **Extract**  $(h_l, L') \in \mathcal{R}^{\text{hash}}$  **using the knowledge extractor**  $\mathsf{K}$ . For any  $i \in [\tau]$ , let  $\bar{st}_i^L$  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 7 of  $\mathcal{A}'_1$ .  
Let  $\mathcal{P}_{\text{leak}}$  be a prover of  $\Pi$  for  $\mathcal{R}^{\text{hash}}$  that upon input the instance  $h_l$ , randomness  $r_p$  and auxiliary input  $\bar{st}_i^L$  does the following:
  - Run an new instance  $\mathcal{S}'_i$  of  $\mathcal{S}'$  with the internal state set to  $\bar{st}_i^L$ .
  - Upon message  $y_j$  with  $j \in [\rho]$  from the verifier, send to  $\mathcal{S}'_i$  the message  $(\text{L}, \text{Prove}(h_l, \cdot, y_1, \dots, y_j), r_p)$ .
  - Upon message  $(\text{msg}, f')$  from the simulator  $\mathcal{S}'_i$  reply  $\perp$  to  $\mathcal{S}'_i$ .
 Notice that  $\mathcal{P}_{\text{leak}}$  makes no leakage oracle queries.
  - i) If the value  $L'$  is unset, run the knowledge extractor  $\mathsf{K}$  on the prover  $\mathcal{P}_{\text{leak}}^{\mathcal{S}'_i}$  on input  $h_l$  and auxiliary input  $\bar{st}_i^L$  and proper randomness<sup>1</sup>. The knowledge extractor  $\mathsf{K}$  outputs a value  $L'$  or aborts. If  $h_l = h(L')$  then set  $L'$  otherwise we say that the  $i$ -th extraction aborts.
  - ii) Keep on running  $\mathcal{A}'_1$  and  $\mathcal{S}'$  as in the simulated experiment until reaching the next iteration.
 If all the extractions abort, the predictor aborts.
- 2.R. **Extract**  $(h_r, R') \in \mathcal{R}^{\text{hash}}$  **using the knowledge extractor**  $\mathsf{K}$ . The procedure is the same as 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 8.
3. The predictor outputs  $m' := \text{Dec}(L', R')$  as its own guess.

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

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

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

$$\Pr [m' = m] = \Pr [\text{Ext}_L \wedge \text{Ext}_R \wedge \text{CohSt} \wedge \neg \text{Coll}] \quad (2)$$

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

**Claim 1**  $\Pr[\text{Ext}_L] \geq \frac{1}{4} - \text{negl}(\kappa)$ .

*Proof.* Consider the execution of step 7 between the adversary and the simulator. Let  $\bar{\mathbf{st}} = \bar{st}_1^L, \dots, \bar{st}_\tau^L \in \{0, 1\}^*$  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 7.

We define a probability  $\text{Free}_L(\bar{st}_i^L)$  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  $\bar{\mathbf{st}} = \bar{st}_1^L, \dots, \bar{st}_\tau^L$  whether or not the simulator leaked in round  $i$  (this information might be contained in the final state  $\bar{st}_\tau^L$ ), the probability  $\text{Free}_L(\bar{st}_i^L)$  might not be 0 or 1, as it is the probability that the simulator leaked in round  $i$  if we would rerun round  $i$  with fresh randomness of the adversary consistent with  $\bar{st}_i^L$ .

<sup>1</sup> The randomness for  $\mathcal{P}_{\text{leak}}^{\mathcal{S}'_i}$  is implicitly defined in the random string  $r_a$ .

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

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

and which is 0 otherwise.<sup>2</sup> After having defined  $\text{Good}(\bar{\mathbf{st}})$  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  $\text{Good}$ .

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

$$\Pr[\text{Ext}_L] \geq \Pr[\text{Ext}_L | \text{Good} = 1] \cdot \Pr[\text{Good} = 1]. \quad (3)$$

We will now focus on bounding  $\Pr[\text{Ext}_L | \text{Good} = 1] \cdot \Pr[\text{Good} = 1]$ . We first bound  $\Pr[\text{Good} = 1]$  and then bound  $\Pr[\text{Ext}_L | \text{Good} = 1]$ .

We first prove that

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

To see this notice that the simulator by the rules of the experiment never queries its leakage oracle in more than  $\lambda_0 \cdot (1 + \gamma)$  rounds: it is not allowed to leak more than  $\lambda_0 \cdot (1 + \gamma)$  bits and each leakage query counts as at least one bit. Therefore there are at least  $\tau - \lambda_0 \cdot (1 + \gamma)$  rounds in which the simulator did not query its oracle. If  $\text{Good} = 0$ , 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  $\frac{3}{4}$ . Set  $\lambda' := \lambda \cdot (1 + \gamma)$ , we can use a union bound to bound the probability of observing this event

$$\Pr[\text{Good} = 0] \leq \binom{\tau}{\tau - \lambda'} \left(\frac{3}{4}\right)^{\tau - \lambda'} \leq \binom{\tau}{\lambda'} 2^{\log_2(3/4)(\tau - \lambda')}. \quad (4)$$

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

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

We now bound  $\Pr[\text{Ext}_L | \text{Good} = 1]$ . Let  $\text{Ext}_L(i)$  be the event that  $\mathcal{K}$  successfully extracts the value  $L'$  at the  $i$ -th iteration of the step 7 of the adversary  $\mathcal{A}'$ . Let  $\text{Accept}_L(i)$  be the event that  $\mathcal{P}_{\text{leak}}$  on input  $h_i$  and auxiliary input  $st_i^L$  gives an accepting proof. It follows from knowledge soundness of  $\Pi$  that

$$\Pr[\text{Ext}_L(i) | \text{Good} = 1] \geq \Pr[\text{Accept}_L(i) | \text{Good} = 1] - \frac{1}{2}.$$

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

$$\Pr[\text{Accept}_L(i) | \text{Good} = 1] \geq 1 - \Pr[\text{Leak}_L(i) | \text{Good} = 1] - \text{negl}(\kappa).$$

To see this assume that  $\mathcal{P}_{\text{leak}}$  upon message  $(\text{msg}, f')$  from  $\mathcal{S}'_i$  would send to the simulator  $f'(\omega \| m)$  instead of  $\perp$ . In that case it gives an acceptable proof with probability  $1 - \text{negl}(\kappa)$  as the adversary leaks an acceptable proof in the real world and the simulator simulates the real world up to negligible difference. Furthermore, sending  $\perp$  when the simulator queries its oracle can only make a difference when it actually sends a query, which happens with probability  $\Pr[\text{Leak}_L(i)]$

Combining the above inequalities we get that

$$\Pr[\text{Ext}_L(i) | \text{Good} = 1] \geq 1 - \Pr[\text{Leak}_L(i) | \text{Good} = 1] - \text{negl}(\kappa) - \frac{1}{2}.$$

<sup>2</sup> Intuitively,  $\text{Good}$  is an indicator for a good event, that, as we will show, has overwhelming probability.

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

$$\text{Free}_L(\bar{st}_{i^*}^L) \geq \frac{3}{4},$$

which implies that

$$\Pr[\text{Ext}_L(i^*) | \text{Good} = 1] \geq \frac{3}{4} - \text{negl}(\kappa) - \frac{1}{2}.$$

Clearly  $\text{Ext}_L(i^*)$  implies  $\text{Ext}_L$ , so we conclude that  $\Pr[\text{Ext}_L | \text{Good} = 1] \geq \frac{1}{4} - \text{negl}(\kappa)$ .

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

The proof proceeds similar to the proof of the last claim, therefore it is omitted. The reason why the condition  $\text{Ext}_L$  does not matter is that the proof exploits 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, as they are done *after* the proofs for the left part.

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

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

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

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

1. Sample uniformly random  $m \leftarrow_s \mathcal{M}$  and random  $h \leftarrow_s \text{Gen}^{\text{CRH}}(1^\kappa)$ ;
2. Run an instance of the predictor  $\mathcal{B}^{\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m}(h, h(m))$ . The predictor needs oracle access to  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$  which can be simulated by  $\mathcal{B}_{\text{coll}}(h)$ .
3. Let  $L', R'$  be 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  $\mathcal{R}^{\text{st}}$ . Let  $st'$  be the internal state of  $\mathcal{S}'(1^\kappa; r_s)$ .

Let  $\mathcal{P}'_{\text{leak}}$  be a prover for  $\Pi$  for the relation  $\mathcal{R}^{\text{st}}$  that upon input the instance  $(\text{crs}, h(L'), h(R'), h(m))$  and auxiliary input  $z := (st', m)$  does the following:

- Run an  $\mathcal{S}'$  with the internal state set to  $st'$ . Sample a random string  $r_p$  long enough to specify all random choices done by the prover of  $\Pi$ .
  - Upon message  $y_j$  with  $j \in [\rho]$  from the verifier, send to  $\mathcal{S}'$  the message  $(\text{rand}, \text{Prove}((\text{crs}, h(L'), h(R'), h(m)), E$  (The next-message function of the prover of  $\Pi$  that uses as input the witness  $\text{Enc}(\text{crs}, m; \omega)$  and the internal randomness set to  $r_p$ .)
  - Upon message  $(\text{msg}, f')$  from the simulator  $\mathcal{S}'$  reply forwarding  $f'(m)$ .
4. Run the knowledge extractor  $\text{K}_{\text{st}}$  on the prover  $\mathcal{P}'_{\text{leak}}$  on input  $(\text{crs}, h(L'), h(R'), h(m))$  and auxiliary input  $z$ . Let  $L'', R''$  be the witness output by the extractor.
  5. If  $L' \neq L''$  output  $(L', L'')$  else  $(R', R'')$ .

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

$$\begin{aligned} \Pr[h(\text{Dec}(L', R')) \neq h(m)] &\geq & (5) \\ &\geq \Pr[h(\text{Dec}(L', R')) \neq h(m) | \text{Ext}_L \wedge \text{Ext}_R] \Pr[\text{Ext}_L \wedge \text{Ext}_R] \geq \\ &\geq 1/\text{poly}(\kappa) \cdot \left(\frac{1}{16} - \text{negl}(\kappa)\right) \end{aligned}$$

On the other hand, the extractor  $\text{K}_{\text{st}}$  succeeds with probability at least  $1 - \text{negl}(\kappa) - \frac{1}{2}$ . Therefore,  $L''$  and  $R''$  are such that  $h(L'') = h(L')$ ,  $h(R'') = h(R')$  and  $h(\text{Dec}(L'', R'')) = h(m)$ .

Combining the latter and the statement of the event in Eq. (5), we have  $h(\text{Dec}(L', R')) \neq h(m) = h(\text{Dec}(L'', R''))$  which implies that either  $L'' \neq L'$  or  $R'' \neq R'$ . Lastly, notice that  $\mathcal{B}_{\text{coll}}$  is an expected polynomial time algorithm. However we can make it polynomial time by aborting if the number of step exceeds some fixed polynomial. By setting the polynomial big enough the probability of  $\mathcal{B}_{\text{coll}}$  finding a collision is still noticeable.

**Claim 4**  $\Pr[\text{Coll} \mid \text{CohSt} \wedge \text{Ext}_L \wedge \text{Ext}_R] \leq \text{negl}(\kappa)$ .

Recall 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:

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

which implies the statement of the lemma.

We conclude the proof of the theorem noticing that, if  $\Sigma$  is  $(\lambda_0, \lambda_1)$ -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.  $\square$

*Remark 1.* The result can be generalized for a weaker version of the split-state model where the code-word is split in many parts. 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 7 for any split). Therefore, the impossibility holds when the number of splits is  $o((\alpha - \lambda_0(1 + \gamma))/\ell_{\text{CRH}})$ . We present the theorem, as stated here, for sake of simplicity.

#### 4.1 The case of constant-size message

**Theorem 2.** *If public-coin  $\text{AoK}_{1/2, \text{negl}(\kappa)}(O(1), \ell_{\text{AoK}}(\kappa))$  for NP, Predictable ZAP for NP with answer length 1 and collision-resistant hash functions with output length  $\ell_{\text{CRH}}(\kappa)$  exist then for any  $\lambda_0 \geq 1$  for any  $\gamma \geq 0$  and for any  $(\alpha, \beta)$ -split-coding scheme  $\Sigma$  with  $\alpha(\kappa) \geq 7 + \gamma$  and if  $\lambda_1(\kappa) \geq 17\lambda_0(\kappa) \cdot (1 + \gamma) \cdot \ell_{\text{AoK}}(\kappa) + 2\ell_{\text{CRH}}(\kappa)$  it holds that  $\Sigma$  is not  $(\lambda_0, \lambda_1)$ -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 as that of Theorem 1 but steps 4, 5 and 6 are substituted by the following:

4. **Leak Predictable ZAP of Consistency.** Let  $\Pi = (\text{Chall}, \text{Resp}, \text{Predict})$  be a PZAP for the relation  $\mathcal{R}$  as defined below:

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

Sample  $(c, tp) \leftarrow_s \text{Chall}(1^\kappa)$  and define the functions:

$$\begin{aligned} f_L(\omega \| m) &:= (h(L) \text{ where } L, R = \text{Enc}(\text{crs}, m; \omega)); \\ f_R(\omega \| m) &:= (h(R) \text{ where } L, R = \text{Enc}(\text{crs}, m; \omega)); \\ f_0(\omega \| m) &:= \text{Resp}(1^\kappa, (\text{crs}, f_L(\omega \| m), f_R(\omega \| m), m), \text{Enc}(\text{crs}, m; \omega), c). \end{aligned}$$

Send the query  $(\text{rand}, f_0)$ . Let  $\text{lk}_0$  be the answer to the query.



5. Send the message `encode`.

6. **Leak Hashed Values and Verify Consistency.** Send the queries  $(L, h), (R, h)$ . Let  $h_l, h_r$  be the answers to the queries. Check if  $\text{Predict}(tp, (h_l, h_r, m)) = \text{lk}_0$ , abort if not.

Consider the following randomized experiment **E**:

- Pick uniformly random  $m \leftarrow_{\$} \mathcal{M}$  and  $h \leftarrow_{\$} \text{Gen}^{\text{CRH}}(1^\kappa)$  and forward to the predictor the value  $h$ .
- Instantiate an oracle  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$  and give the predictor access to it.

**Lemma 5.**  $\mathbb{H}_\infty(m|\mathbf{E}) \geq |m| - \lambda_0 \cdot (1 + \gamma)$ .

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

**Lemma 6.** *If  $\Sigma$  is  $(\lambda_0, \lambda_1)$ -FLR-sim-secure then  $\widetilde{\mathbb{H}}_\infty(m|\mathbf{E}) \leq 6$ .*

The proof of the lemma follows the same line of Lemma 4. Notice that in the experiment **E** the predictor does not get input  $m$  (which it 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  $\text{Predict}(tp, (\text{crs}, h_l, h_r, m)) = \text{lk}_0$  (in the case the condition holds we say that  $\text{lk}_0$  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 Predictable ZAP and, conditioned on  $\text{lk}_0$  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 $\text{NC}^0$ -Fully LR Code

We start by giving a well-known characterization of the class  $\text{NC}^0$ .

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

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

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

Let  $\mathbb{F}$  be a finite field and let  $\Phi_{\mathbb{F}}^n = (\text{Enc}, \text{Dec})$  be as follows:

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

**Theorem 3 (from [DF11]).** *The encoding scheme  $\Phi_{\mathbb{F}}^n$  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  $\text{NC}^0$ -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  $\lambda_0, \lambda_1$  such that  $\delta \cdot \lambda_0 + \lambda_1 < 0.3 \cdot |\mathbb{F}^n|$  the encoding scheme  $\Phi_{\mathbb{F}}^n$  is  $(\text{NC}^0, \lambda_0, \lambda_1)$ -FLR-SIM-secure.

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

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 message  $(\text{rand}, f)$  from  $\mathcal{A}_1$ , let  $I$  be the set of indexes such that  $f$  depends on  $I$  only. Define  $I_L := I \cap [qn]$  and  $I_R := I \cap [qn + 1, 2qn]$ . Define the functions:

$$\begin{aligned} f_L(\mathbf{L}) &:= (\text{bit}(\mathbf{L})_i \text{ for } i \in I_L); \\ f_R(\mathbf{R}) &:= (\text{bit}(\mathbf{R})_i \text{ for } i \in I_R). \end{aligned}$$

Send the queries  $(\mathbf{L}, f_L)$  and  $(\mathbf{R}, f_R)$  and let  $\text{lk}_L$  and  $\text{lk}_R$  be the answers to the queries. Hardwire such values and evaluate the function  $f$  on input  $m$ . Namely, compute  $\text{lk}_f := f(f_L(\mathbf{L}), f_R(\mathbf{R}), m)$  and send it back to  $\mathcal{A}_1(st)$ .

- Upon message  $(\mathbf{X}, f)$  where  $\mathbf{X} \in \{\mathbf{L}, \mathbf{R}\}$  from  $\mathcal{A}_1$  forward the message.

W.l.o.g. assume that every leakage query to  $\mathcal{O}_{\lambda_0}^{\omega||m}$  has output length 1 and that the adversary makes exactly  $\lambda_0$  queries. By Lemma 7 there exists a constant  $\delta \in \mathbb{N}$  such that for the  $i$ -th leakage query made by  $\mathcal{A}_1$  to  $\mathcal{O}_{\lambda_0}^{\omega||m}$  the adversary  $\mathcal{A}'$  leaks  $\delta$  bits from  $\mathcal{O}_{\lambda_1}^{\mathbf{L}}, \mathcal{O}_{\lambda_1}^{\mathbf{R}}$ . By construction:

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

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

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

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

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

This conclude the proof of the theorem.  $\square$

*Remark 2.* The proof exploits only marginally the structure of  $\Phi_{\mathbb{F}}^n$ . It is not hard to see that the theorem can be generalized for any coding scheme  $(\text{Gen}, \text{Enc}, \text{Dec})$  where for any message  $m \in \mathcal{M}$  the function  $\text{Enc}(m; \cdot)$  is invertible in  $\text{NC}^0$ . We present the theorem, as stated here, only for sake of concreteness.

*Remark 3.* The construction is secure under the slightly stronger definition where the adversary does not lose access to  $\mathcal{O}_{\lambda_0}^{\omega||m}$  after having sent the message  $\text{encode}$ .

## 5.2 A Compiler from LRC to weak FLRC.

Given a  $(\alpha, \beta)$ -split-coding scheme  $\Sigma = (\text{Gen}, \text{Enc}, \text{Dec})$  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  $\Sigma' = (\text{Gen}', \text{Enc}', \text{Dec}' := \text{Dec})$  as follow:

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

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

$$\begin{aligned} r &\geq \lambda_0 + \lambda_1 + 2 \log(1/\epsilon) + \log(t) + 3, \\ t &\geq \lambda_0 \cdot \log |\mathcal{F}| + \alpha + \lambda_0 + \lambda_1 + 2 \log(1/\epsilon). \end{aligned}$$

*Proof.* Given an adversary  $\mathcal{A}'$  against  $\Sigma'$ , we define a ppt adversary  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$  against  $\Sigma$  as follow:

- **Adversary  $\mathcal{A}_0$ :** On input  $\text{crs}$ , it picks at random  $h \leftarrow_{\mathcal{S}} \mathcal{H}_{r,t}$ , a random string  $r \leftarrow_{\mathcal{S}} \{0, 1\}^{p(\kappa)}$  for a polynomial  $p$  that bounds the running time of  $\mathcal{A}'$  and runs  $\mathcal{A}'_0(1^\kappa; r)$  and a random string  $\omega \leftarrow_{\mathcal{S}} \mathcal{R}$ . Upon leakage oracle query  $f$  to  $\mathcal{O}_{\lambda_0}^\omega$  from  $\mathcal{A}'_0$ , it replies  $f(\omega)$ . Upon notification  $\text{getCRS}$  from  $\mathcal{A}'_0$ , it replies by forwarding the tuple  $(h, \text{crs})$ . Eventually, the adversary  $\mathcal{A}'_0$  outputs a message  $m \in \mathcal{M}$  and a state value  $st$ ,  $\mathcal{A}_0$  outputs  $m$  and  $st' = (st, h)$ .
- **Adversary  $\mathcal{A}_1$ :** On inputs  $st' = (st, h)$  and  $\text{crs}$ , it runs  $\mathcal{A}'_1(st, (h, \text{crs}))$  and forwards all the queries made by  $\mathcal{A}'_1$ .

W.l.o.g. the adversary  $\mathcal{A}_0$  makes the sequence  $(\text{rand}, f_1), (\text{rand}, f_2), \dots, (\text{rand}, f_{\lambda_0})$  of queries. Let  $\mathbf{f} := (f_1, \dots, f_{\lambda_0}) \in \mathcal{F}^{\lambda_0}$ , therefore view of  $\mathcal{A}'$  in the real experiment is:

$$\text{wReal}_{\mathcal{A}', \Sigma'}^{\lambda_0, \lambda_1}(\kappa) = (r, \mathbf{f}(\omega), (h, \text{crs}), \text{lk}_L, \text{lk}_R)$$

On the other hand, by definition of the adversary  $\mathcal{A}$ , the view provided to  $\mathcal{A}'$  is  $(r, \mathbf{f}(\omega), (h, \text{crs}), \text{lk}_{L'}, \text{lk}_{R'})$  where  $L', R' = \text{Enc}(\text{crs}, m; \omega')$  and  $\omega, \omega' \leftarrow_{\mathcal{S}} \mathcal{R}$ .

**Claim 5**  $\{\text{wReal}_{\mathcal{A}', \Sigma'}^{\lambda_0, \lambda_1}(\kappa)\}_{\kappa \in \mathbb{N}} \approx_{2\epsilon(\kappa)} \{(r, \mathbf{f}(\omega), (h, \text{crs}), \text{lk}_L, \text{lk}_R)\}_{\kappa \in \mathbb{N}}$ .

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:

$$\{\text{Real}_{\mathcal{A}, \Sigma}^{0, \lambda_1}(\kappa)\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx}_{\epsilon(\kappa)} \{\text{Ideal}_{\mathcal{A}, \mathcal{S}}^{0, \lambda_1}(\kappa)\}_{\kappa \in \mathbb{N}}. \quad (6)$$

Let  $\mathcal{S}'$  be defined as the adversary  $\mathcal{A}$  interacting with the simulator  $\mathcal{S}$ . Since  $\text{Ideal}_{\mathcal{A}, \mathcal{S}}^{0, \lambda_1}(\kappa) = ((h, r, \omega), \text{crs}, \tilde{\text{lk}}_L, \tilde{\text{lk}}_R)$  it holds that:

$$\text{wIdeal}_{\mathcal{A}', \mathcal{S}'}^{\lambda_0, \lambda_1}(\kappa) = (r, \mathbf{f}(\omega), (h, \text{crs}), \tilde{\text{lk}}_L, \tilde{\text{lk}}_R).$$

It follows from a simple reduction to Eq. (6) that:

$$\{(r, \mathbf{f}(\omega), (h, \text{crs}), \text{lk}_{L'}, \text{lk}_{R'})\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx}_{\epsilon(\kappa)} \{\text{wIdeal}_{\mathcal{A}', \mathcal{S}'}^{\lambda_0, \lambda_1}(\kappa)\}_{\kappa \in \mathbb{N}}.$$

We conclude by applying Claim 5 to equation above.  $\square$

*Proof (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  $\text{Gen}$ . Therefore, w.l.o.g., we can consider them fixed in the views.

Recall that the adversary  $\mathcal{A}$  defines for  $\mathcal{A}'$  a hybrid environment where the leakage on the randomness is on  $\omega$  but the codeword is instantiated using fresh randomness  $\omega'$ . We prove the stronger statement that the two views are statistical close with high probability over the choice of the  $t$ -wise hash function  $h$ . For convenience, we define two tuples of random variables:

$$\begin{aligned} \text{Real}_h &:= ( \mathbf{f}(\omega), \text{lk}_L, \text{lk}_R \mid (L, R) = \text{Enc}(\text{crs}, m; h(\omega)) ) \\ \text{Hyb}_h &:= ( \mathbf{f}(\omega), \text{lk}_{L'}, \text{lk}_{R'} \mid (L', R') = \text{Enc}(\text{crs}, m; h(\omega')) ) \end{aligned}$$

We prove that

$$\Pr[\text{Hyb}_h \approx_\epsilon \text{Real}_h] \geq 1 - \epsilon,$$

where the probability is over the choice of  $h \leftarrow \mathcal{H}_{r,t}$ . Let  $\text{Bad}$  be the event  $\{\text{Hyb}_h \not\approx_\epsilon \text{Real}_h\}$ .

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

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

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

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

For any 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  $\Pr_h[\text{Real}_h = v \mid \omega = \bar{\omega}] = \Pr_{\omega'}[\text{Hyb}_h = v \mid \omega = \bar{\omega}, h = \bar{h}]$ . It follows that

$$\begin{aligned} &\Pr_{h \leftarrow \mathcal{H}_{r,t}} \left[ \sum_v \left| \Pr_{\omega}[\text{Real}_h = v] - p_v \right| > 2\epsilon \right] \\ &\leq \Pr_{h \leftarrow \mathcal{H}_{r,t}} \left[ \exists v : \left| \Pr_{\omega}[\text{Real}_h = v] - p_v \right| > \epsilon \cdot \tilde{p}_v \right] \\ &\leq \sum_{v \in \{0,1\}^\lambda} \Pr_{h \leftarrow \mathcal{H}_{r,t}} \left[ \left| \Pr_{\omega}[\text{Real}_h = v] - p_v \right| > \epsilon \cdot \tilde{p}_v \right] \\ &\leq \sum_{v \in \{0,1\}^\lambda} \Pr_{h \leftarrow \mathcal{H}_{r,t}} \left[ \left| \sum_{\bar{\omega}} Y_{\bar{\omega},v} - 2^r p_v \right| > 2^r \epsilon \cdot \tilde{p}_v \right] \\ &\leq \sum_{v \in \{0,1\}^\lambda} 8 \left( \frac{t \cdot 2^r p_v + t^2}{(2^r \epsilon \cdot \tilde{p}_v)^2} \right)^{t/2} \tag{7} \end{aligned}$$

$$\leq \sum_{v \in \{0,1\}^\lambda} 8 \left( \frac{2t \cdot 2^r \tilde{p}_v}{(2^r \epsilon \cdot \tilde{p}_v)^2} \right)^{t/2} \tag{8}$$

$$\leq 2^\lambda \cdot 8 \left( \frac{2t}{2^{r-\lambda} \cdot \epsilon^2} \right)^{t/2} \tag{9}$$

where Eq. (7) follows by Lemma 2 and Eq. (8) and Eq. (9) follow because  $2^r \cdot \tilde{p}_v \geq 2^{r-\lambda} \geq t$ . Combining all together we have:

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

To make the above negligible we can set:

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

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

## References

- ABG<sup>+</sup>13. Prabhajan 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. Prabhajan 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<sup>+</sup>14. Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinfeld, 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 pre-processing. 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 Davi, 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.
- FMNV15. Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient von neumann architecture. In *PKC*, pages 579–603, 2015.
- 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. 2015. <http://ia.cr/2015/740>.
- FRR<sup>+</sup>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<sup>+</sup>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 indistinguishability-based definition:

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

1. Upon input  $1^\kappa$  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  $(L, R) \leftarrow_{\$} \text{Enc}(m_0)$ ;
3. Upon input  $st$  and oracle access to  $\mathcal{O}_{\lambda_1}^L$  and  $\mathcal{O}_{\lambda_1}^R$  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  $(\alpha, \beta)$ -split-coding scheme  $\Sigma$  is said to be  $(\lambda_1, \epsilon)$ -LR-ind-secure if for any unbounded adversary  $\mathcal{A}$ :

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

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

**Theorem 6.** Let  $\Sigma$  be a split-coding scheme. If  $\Sigma$  is  $(\lambda_1, \epsilon)$ -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  $\mathcal{S}$ :

$$\{\text{Real}_{\mathcal{A}, \Sigma}^{0, \lambda_1}(\kappa)\}_{\kappa \in \mathbb{N}} \not\stackrel{c}{\approx}_{\epsilon} \{\text{Ideal}_{\mathcal{A}, \mathcal{S}}^{0, \lambda_1}(\kappa)\}_{\kappa \in \mathbb{N}} \quad (10)$$

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

- Upon input  $1^\kappa$  encodes  $(L, R) \leftarrow_{\$} \text{Enc}(0^\alpha)$ .
- Upon message  $(X, f)$  from  $\mathcal{A}$  where  $X \in \{L, R\}$  it replies with  $f(L)$  if  $X = L$  otherwise  $f(R)$ .

By Eq. (10) there exists a ppt distinguisher  $\mathcal{D}$  such that:

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

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

- Upon input  $1^\kappa$  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  $\Sigma$ :

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

This ends the proof of the theorem.

## B Proof of Theorem 2.

We give the definition of Predictable ZAPs from [FNV15].

**Definition 7 (Predictable ZAP).** Let  $\Pi = (\text{Chall}, \text{Resp}, \text{Predict})$  and let  $R$  be an NP relation. Consider the properties below.

**Completeness:** There exists a negligible function  $\nu : \mathbb{N} \rightarrow [0, 1]$  such that for all  $(x, w) \in \mathcal{R}$ :

$$\Pr_{c, tp} [\text{Predict}(1^\kappa, tp, x) \neq \text{Resp}(1^\kappa, x, w, c) : (c, tp) \leftarrow \text{Chall}(1^\kappa)] \leq \nu(\kappa).$$

**(Adaptive) Knowledge soundness with error  $\epsilon$ :** For all ppt provers  $\mathcal{P}^*$  making polynomially many queries to its oracle, there exists a ppt extractor  $K$  and a non-zero polynomial  $q(\cdot)$  such that for any auxiliary input  $z \in \{0, 1\}^*$  the following holds. Whenever

$$p_{z,r}(\kappa) := \Pr \left[ \begin{array}{l} (c, tp) \leftarrow \text{Chall}(1^\kappa), \\ a = b : (x, a) \leftarrow \mathcal{P}^*(c, z; r) \text{ where } |x| = \kappa, \\ b := \text{Predict}(1^\kappa, tp, x). \end{array} \right] > \epsilon(\kappa),$$

we have

$$\Pr \left[ \begin{array}{l} (c, tp) \leftarrow \text{Chall}(1^\kappa), \\ (x, w) \in \mathcal{R} : (x, a) \leftarrow \mathcal{P}^*(c, z; r) \text{ where } |x| = \kappa, \\ w \leftarrow K(1^\kappa, x, z, r, \mathcal{Q}). \end{array} \right] \geq q(p_{z,r}(\kappa) - \epsilon(\kappa)).$$

In the above equations, we denote by  $V(1^\kappa, tp, \cdot, \cdot)$  the oracle machine that upon input a query  $(x, a)$  computes  $b := \text{Predict}(1^\kappa, tp, x)$  and outputs 1 iff  $a = b$ ; we also write  $\mathcal{Q}$  for the list  $\{(x_i, a_i), d_i\}$  of oracle queries (and answers to these queries) made by  $\mathcal{P}^*$ .

Let  $\ell$  be the size of the prover's answer, we call  $\Pi$  a predictable ZAP (PZAP) for  $\mathcal{R}$  if  $\Pi$  satisfies completeness and adaptive knowledge soundness with error  $\epsilon$ , and moreover  $\epsilon - 2^{-\ell}$  is negligible.

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

**Theorem 2.** If public-coin  $\text{AoK}_{1/2, \text{negl}(\kappa)}(O(1), \ell_{\text{AoK}}(\kappa))$  for NP, Predictable ZAP for NP with answer length 1 and collision-resistant hash functions with output length  $\ell_{\text{CRH}}(\kappa)$  exist then for any  $\lambda_0 \geq 1$  for any  $\gamma \geq 0$  and for any  $(\alpha, \beta)$ -split-coding scheme  $\Sigma$  with  $\alpha(\kappa) \geq 7 + \gamma$  and if  $\lambda_1(\kappa) \geq 17\lambda_0(\kappa) \cdot (1 + \gamma) \cdot \ell_{\text{AoK}}(\kappa) + 2\ell_{\text{CRH}}(\kappa)$  it holds that  $\Sigma$  is not  $(\lambda_0, \lambda_1)$ -FLR-sim-secure.

*Proof.* We first recall the necessary notation. Given a random variable  $x$  we use the notation  $\bar{x}$  to refer to a possible assignment of the random variable. Let  $\Pi$  be in  $\text{AoK}_{\text{negl}(\kappa), \text{negl}(\kappa)}(O(1), \ell_{\text{AoK}}(\kappa))$  and a public-coin argument system for NP. For concreteness let  $\rho$  be the round complexity of the  $\Pi$ . Let  $(\text{Gen}^{\text{CRH}}, \text{Eval}^{\text{CRH}})$  be a collision resistant hash function with output length  $\ell_{\text{CRH}}(\kappa)$ . Let  $\Pi' := (\text{Chall}, \text{Resp}, \text{Predict})$  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  $h \leftarrow \text{Gen}^{\text{CRH}}(1^\kappa)$ ;
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  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$ , also  $\mathcal{A}'_0(1^\kappa)$  outputs the state  $st := (h, h(m))$ .



4. **Leak Predictable ZAP of Consistency.** Let  $\Pi = (\text{Chall}, \text{Resp}, \text{Predict})$  be a PZAP for the relation  $\mathcal{R}$  as defined below:

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

Sample  $(c, tp) \leftarrow_s \text{Chall}(1^\kappa)$  and define the functions:

$$\begin{aligned} f_L(\omega \| m) &:= (h(L) \text{ where } L, R = \text{Enc}(\text{crs}, m; \omega)); \\ f_R(\omega \| m) &:= (h(R) \text{ where } L, R = \text{Enc}(\text{crs}, m; \omega)); \\ f_0(\omega \| m) &:= \text{Resp}(1^\kappa, (\text{crs}, f_L(\omega \| m), f_R(\omega \| m), m), \text{Enc}(\text{crs}, m; \omega), c). \end{aligned} \quad (11)$$

Send the query  $(\text{rand}, f_0)$ . Let  $\text{lk}_0$  be the answer to the query.

5. Send the message `encode`.

6. **Leak Hashed Values and Verify Consistency.** Send the queries  $(L, h), (R, h)$ . Let  $h_l, h_r$  be the answers to the queries. Check if  $\text{Predict}(tp, (h_l, h_r, m)) = \text{lk}_0$ , abort if not.

7. **Leak Arguments of Knowledge of the Left part.** Consider the following relation:

$$\mathcal{R}^{\text{hash}} := \{(y, x) : h(x) = y\}$$

Let  $\tau := 17\lambda_0 \cdot (1 + \gamma)$ , for any  $i \in [\tau]$  leak an argument of knowledge for  $h_l \in \mathcal{L}_{\mathcal{R}^{\text{hash}}}$  from L. If the leaked argument is not accepting then abort. Let  $\pi_i^L$  be the leaked transcript.

8. **Leak Arguments of Knowledge of the Right part.** for any  $i \in [\tau]$  leak an argument of knowledge for  $h_r \in \mathcal{L}_{\mathcal{R}^{\text{hash}}}$  from R. If the leaked argument is not accepting then abort. Let  $\pi_i^R$  be the leaked transcript.

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

- Pick uniformly random  $m \leftarrow_s \mathcal{M}$  and  $h \leftarrow_s \text{Gen}^{\text{CRH}}(1^\kappa)$  and forward  $h$  to the predictor.
- Instantiate an oracle  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$  and give the predictor access to it.

**Lemma 8.**  $\tilde{\mathbb{H}}_\infty(m \mid \mathbf{E}) \geq \alpha - \lambda_0 \cdot (1 + \gamma)$ .

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

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

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

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

□

**Lemma 9.** If  $\Sigma$  is a  $(\lambda_0, \lambda_1)$ -FLR-sim-secure then  $\tilde{\mathbb{H}}_\infty(m \mid \mathbf{E}) \leq 6$ .

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

$$\{\text{Real}_{\mathcal{A}', \Sigma}^{\lambda_0, \lambda_1}(\kappa)\}_\kappa \stackrel{c}{\approx}_\epsilon \{\text{Ideal}_{\mathcal{A}', \mathcal{S}', \gamma}^{\lambda_0, \lambda_1}(\kappa)\}_\kappa. \quad (12)$$

Consider the adversary  $\mathcal{A}''$  that executes the same code of  $\mathcal{A}'$  but, in step 6, does not check that  $\text{Predict}(tp, (h_l, h_r, m)) = \text{lk}_0$ . Notice that we can implement  $\mathcal{A}''$  without providing the input  $m$ . We have that:

$$\{\text{Ideal}_{\mathcal{A}', \mathcal{S}', \gamma}^{\lambda_0, \lambda_1}(\kappa)\}_{\kappa} \approx \{\text{Ideal}_{\mathcal{A}'', \mathcal{S}'}^{\lambda_0, \lambda_1}(\kappa)\}_{\kappa}. \quad (13)$$

In fact, conditioned on the event  $\{\text{Predict}(tp, (h_l, h_r, m)) = \text{lk}_0\}$  the two distributions are exactly the same. On the other hand, Eq. (12) implies that the event happens with probability  $1 - \text{negl}(\kappa) - \epsilon(\kappa)$  (where the negligible factor  $\text{negl}$  comes from the completeness of the PZAP).

For the sake of the proof we first build a predictor which is given  $h$  and tries to guess  $m$ . We then use this predictor to prove the lemma. Let  $\text{K}$  be the extractor given by the knowledge soundness property of the argument of knowledge for the relation  $\mathcal{R}^{\text{hash}}$ . Consider the following predictor  $\mathcal{B}$  that takes as input  $h$  and has oracle access to  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$ :

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  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$ . (The adversary  $\mathcal{A}''_1$  starts by leaking a PZAP for  $\mathcal{R}^{\text{st}}$ , then it sends the message  $\text{encode}$  and leaks the values  $h_l, h_r$  from  $\mathcal{O}_{\lambda_1}^L, \mathcal{O}_{\lambda_1}^R$ .)
- 2.L. **Extract**  $(h_l, L') \in \mathcal{R}^{\text{hash}}$  **using the knowledge extractor**  $\text{K}$ . For any  $i \in [\tau]$ , let  $\bar{st}_i^L$  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 7 of  $\mathcal{A}''_1$ . Let  $\mathcal{P}_{\text{leak}}$  be a prover of  $\Pi$  for  $\mathcal{R}^{\text{hash}}$  that upon input the instance  $h_l$ , randomness  $r_p$  and auxiliary input  $\bar{st}_i^L$  does the following:
  - Run a new instance  $\mathcal{S}'_i$  of  $\mathcal{S}'$  with the internal state set to  $\bar{st}_i^L$ .
  - Upon message  $y_j$  with  $j \in [\rho]$  from the verifier, send to  $\mathcal{S}'_i$  the message  $(L, \text{Prove}(h_l, \cdot, y_1, \dots, y_j); r_p)$ .
  - Upon message  $(\text{msg}, f')$  from the simulator  $\mathcal{S}'_i$  reply forwarding  $\perp$  to  $\mathcal{S}'_i$ .

Notice that  $\mathcal{P}_{\text{leak}}$  makes no leakage oracle queries.

- i) If the value  $L'$  is unset, run the knowledge extractor  $\text{K}$  on the prover  $\mathcal{P}_{\text{leak}}^{\mathcal{S}'_i}$  on input  $h_l$  and auxiliary input  $\bar{st}_i^L$  and proper randomness. The knowledge extractor  $\text{K}$  outputs a value  $L'$  or aborts. If  $h_l = h(L')$  then set  $L'$  otherwise we say that the  $i$ -th extraction aborts.
- ii) Keep on running  $\mathcal{A}''_1$  and  $\mathcal{S}'$  as in the simulated experiment until reach the next iteration.

If all the extractions abort, the predictor aborts.

- 2.R. **Extract**  $(h_r, R') \in \mathcal{R}^{\text{hash}}$  **using the knowledge extractor**  $\text{K}$ . 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 8.
3. The predictor outputs  $m' := \text{Dec}(L', R')$  as its own guess.
  - Let  $\text{Ext}_L$  (resp.  $\text{Ext}_R$ ) be the event that  $\text{K}$  successfully extracts a value  $L'$  (resp.  $R'$ ).
  - Let  $\text{CohSt}$  be the event  $\{h(\text{Dec}(L', R')) = m\}$ .

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

$$\Pr[m' = m] = \Pr[\text{Ext}_L \wedge \text{Ext}_R \wedge \text{CohSt}] \quad (14)$$

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

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

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

*Proof.* We reduce to the collision resistance property of  $h$  and the adaptive knowledge soundness of the PZAP  $\Pi'$ , Suppose that

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

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

- Sample uniformly random  $m \leftarrow_{\$} \mathcal{M}$  and random  $h \leftarrow_{\$} \text{Gen}^{\text{CRH}}(1^\kappa)$ ;
  - Run an instance of the predictor  $\mathcal{B}^{\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m}(h)$ , the predictor needs oracle access to  $\mathcal{O}_{\lambda_0 \cdot (1+\gamma)}^m$  which can be simulated by  $\mathcal{B}_{\text{coll}}(h)$ .
  - Consider the following prover  $\mathcal{P}^*$ :
    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  $\mathcal{S}'(1^\kappa; r_s)$ :
      - The adversary sends  $(\text{rand}, f_0)$  as defined in Eq. (11), let  $\text{lk}_0$  be the answer given by the simulator. The simulator possibly send messages of the form  $(\text{msg}, f')$  reply with  $f'(m)$ ;
      - 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  $h_r$ ;
    3. Output  $(\text{crs}, h_l, h_r, m)$  as instance and  $\text{lk}_0$  as answer.
- Run the knowledge extractor  $K'$  given by the definition of PZAP for the adversary  $\mathcal{P}^*$ , specifically we run  $K'$  on the same randomness and auxiliary input of  $\mathcal{P}^*$ . 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  $\mathcal{B}_{\text{coll}}$  simulates perfectly the randomized experiment  $\mathbf{E}$  therefore:

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

On the other hand, the extractor  $K'$  succeeds with probability at least  $q(1 - \text{negl}(\kappa) - \frac{1}{2}) - \text{negl}(\kappa)$ . Notice that the definition of PAoK implies that the probability of  $K'$  extracting and  $\mathcal{P}^*$  succeeding 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  $\mathcal{P}^*$  wins with overwhelming probability, therefore the ratio of the intersection of the instances for which both  $K'$  extracts and  $\mathcal{P}^*$  succeeds is negligible close to the set of instance for which  $K'$  extracts, this explain why we lose an extra  $\text{negl}$  factor.

Therefore,  $L''$  and  $R''$  are such that  $h(L'') = h(L')$ ,  $h(R'') = h(R')$  and  $\text{Dec}(L'', R'') = m$ . The latter implies  $\text{Dec}(L', R') \neq \text{Dec}(L'', R'')$  which implies that either  $L'' \neq L'$  or  $R'' \neq R'$ . Lastly, notice that  $\mathcal{B}_{\text{coll}}$  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  $\mathcal{B}_{\text{coll}}$  finding a collision is still noticeable.

Summarizing, we have:

$$\begin{aligned} \Pr[m' = m] &= \Pr[\text{Ext}_L \wedge \text{Ext}_R \wedge \text{CohSt}] \geq \\ &\geq (\frac{1}{16} - \text{negl}(\kappa)) \cdot (\frac{1}{2} - \text{negl}(\kappa)) \geq \frac{1}{64} \end{aligned}$$

which implies the statement of the lemma.

We conclude the proof of the theorem noticing that if  $\Sigma$  is  $(\lambda_0, \lambda_1)$ -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.  $\square$