After-the-Fact Leakage in Public-Key Encryption

Shai Halevi
IBM Research

Huijia Lin*
Cornell University

January 6, 2011

Abstract

What does it mean for an encryption scheme to be leakage-resilient? Prior formulations require that the scheme remains semantically secure even in the presence of leakage, but only considered leakage that occurs *before the challenge ciphertext is generated*. Although seemingly necessary, this restriction severely limits the usefulness of the resulting notion.

In this work we study after-the-fact leakage, namely leakage that the adversary obtains after seeing the challenge ciphertext. We seek a "natural" and realizable notion of security, which is usable in higher-level protocols and applications. To this end, we formulate *entropic leakage-resilient PKE*. This notion captures the intuition that as long as the entropy of the encrypted message is higher than the amount of leakage, the message still has some (pseudo) entropy left. We show that this notion is realized by the Naor-Segev constructions (using hash proof systems).

We demonstrate that entropic leakage-resilience is useful by showing a simple construction that uses it to get semantic security in the presence of after-the-fact leakage, in a model of bounded memory leakage from a split state.

1 Introduction

In the traditional view of cryptography, some parts of the system are designated as secret, and these parts are kept beyond reach for the attackers and only interact with the non-secret parts via well defined interfaces under the control of the designers. In contrast, in reality many times attackers can "design their own interfaces" for accessing the secret state. For example, they may get parts of the secret state via a myriad of side-channels (e.g., timing, radiation, etc.), read it off some backup tape or physical memory, or maybe bribe people who have access to parts of the secret state (or install a virus on their machines). Recent years saw many advances in our ability to reason formally about such unintended leakage and to construct schemes that resist broad class of leakage attacks (e.g., [14, 16, 10, 1, 17, 4, 15, 11] and others). This line of work is typically referred to as *leakage-resilient cryptography*.

The general theme in formulating leakage resilience of some primitive is that in addition to the usual interfaces that are available by design, the adversary can also choose arbitrary leakage functions (from some broad class) and get the result of applying these functions to the secret state of the scheme. We then require that the scheme still meets the original notion of security, even in the presence of this more powerful adversary. This approach was successfully applied to model leakage resilience of many cryptographic schemes, such as pseudorandom generators, signature schemes, etc.

*Supported by a Microsoft Research Fellowship.Part of this work was done while visiting IBM Research.

The same approach was also applied to leakage-resilience of encryption schemes, e.g., in [1, 17, 4], but here there seems to be a problem: Basic notions of security for encryption schemes require that they hide the content of the plaintext even from an adversary that knows many things about that plaintext. In particular, semantic security of encryption requires that an adversary that knows two messages m_0, m_1 and sees a ciphertext c encrypting one of them, will not be able to tell which of the two messages is encrypted in c. But if we let the adversary learn arbitrary functions of the secret key, then it could ask for a function that decrypts the "challenge ciphertext" c and outputs 0 if it is decrypted to m_0 and 1 otherwise. In other words, *given the challenge ciphertext the adversary can design a leakage function that will leak to it exactly the one bit that we try to hide using encryption.*

Prior work on leakage-resilient PKE [1, 17, 4] bypassed this definitional difficulty by only considering before-the-fact leakage. Namely, the adversary could only ask for leakage on the secret key before it sees the challenge ciphertext, and the scheme was deemed leakage resilient if it remained semantically secure in face of such leakage. This approach indeed bypasses the technical problem, but pays dearly in terms of the meaning and applicability of the resulting notions. Indeed this solution means that as soon as even one bit of the secret key is leaked, we cannot say anything about the secrecy of any message that was encrypted before that bit was leaked.

Consider for example trying to address memory leakage (such as the "cold boot attacks" [12]) by using leakage-resilient encryption. In a memory-leakage attack, an attacker may get a laptop where the disk is encrypted. Lacking the password to access the decryption functionality, the attacker may try to read the decryption key directly off the physical memory. In this setting, the adversary could first see the encrypted disk (hence getting access to the ciphertext), and then try to design a method of measuring the memory specifically for the purpose of decrypting this ciphertext. Existing notions of before-the-fact leakage-resilient encryption say nothing about the secrecy of the plaintext under this attack. This definitional problem was acknowledged in prior work, but no solutions were offered. For example, Naor and Segev wrote in [17] that "It will be very interesting to find an appropriate framework that allows a certain form of challenge-dependent leakage."

1.1 Our contributions

In this work we study after-the-fact leakage, where the adversary obtains leakage information after seeing the challenge ciphertext. Our main contribution is formulating the notion of *entropic leakage-resilient PKE* and showing how to meet it. Intuitively, this notion says that even if the adversary designs its leakage function according to the challenge ciphertext to leak the things it wants to know, if it only leaks k bits then it cannot "amplify" them to learn more than k bits about the plaintext. Technically, our notion can be viewed as an extension of HILL entropy [13] to the interactive setting. Namely, our notion would say that the message still looks like it has some min-entropy, even to the interactive adversary that participated in the game of semantic-security with leakage.

We remark that this notion is not trivial: Indeed it is not hard to construct "contrived" encryption schemes that are semantically secure (even with respect to before-the-fact leakage), but such that leaking (say) $\sqrt{n}$ bits after the fact lets the adversary recover n bits of plaintext. On the other hand, we show that the same construction that Naor and Segev used in [17] for obtaining leakage-resilient encryption from hash proof systems, in fact realizes also the stronger notion of entropic leakage-resilience relative to after-the-fact leakage.

To demonstrate the usefulness of entropic leakage-resilience we show that in some cases it can be used to get full semantic security, even in the presence of after-the-fact leakage. For this, we of course have to limit the type of leakage functions that the adversary has access to. Specifically, we

consider a model where the key is broken into several parts, and the adversary can get access to leakage from every part separately, but not to a global leakage from the entire secret state. (This model is often used in conjunction with the only-computation-leaks axiom of Micali and Reyzin [16].)

To get semantic security in that model, we use two instances of an entropic leakage resilient encryption scheme. To encrypt a message m, we choose two random long strings x_1, x_2, encrypt each x_i under a different copy of the entropic scheme, and use a two-source extractor to compute $Ext2(x_1, x_2) \oplus m$. To decrypt we recover the two strings x_1, x_2 (which we can do by working separately with the two secret keys) and then recover m. The intuition is that as long as the adversary can only get leakage functions from the two keys separately, the entropic leakage resilience of the underlying scheme implies that x_1, x_2 still have a lot of entropy, and hence $Ext2(x_1, x_2)$ still hides the message. (We remark that we view this construction more as an example to the usefulness of our new notion than as a stand-alone application.)

Discussion: on defining useful leakage primitives. On some level, our notion of entropic leakage-resilience departs from the usual theme described above for defining leakage-resilience primitives. Namely, we no longer insist that the scheme retains its original security properties even in the face of leakage. In the face of the impossibility of achieving the strong notion of semantic security, we are willing to settle on a weaker achievable notion so long as it is useful in higher-level applications. It is interesting to formulate such useful weakened notions also for other primitives, such as commitment, key-agreement, etc.

In this context we note that when thinking about encryption as part of a communication system, our notion only captures leakage at the receiver side (i.e., from the secret key) and not at the sender side (i.e., from the encryption randomness). It is interesting to find ways of simultaneously addressing leakage at both ends.

1.2 Recent Related Work

We mention that two recent works by Goldwasser and Rothblum [11], and Juma and Vahlis [15] implicitly also considered after-the-fact leakage for encryption schemes. They presented general methods for compiling any circuit with secret components into one that resists continuous leakage (in the only-computation-leaks model), using leakage-free hardware. Their transformations use as a technical tool encryptions schemes that remain semantically secure even at the presence of after-the-fact leakage of the secret key, provided that the adversary sees *only part of the challenge ciphertext*. (Such notion of semantic-security with respect to adversaries that cannot see the entire challenge ciphertext, if defined as a stand-alone primitive, could be another example of a useful weaker leakage primitive.)

2 Preliminaries

We denote random variables by uppercase English letters. For a random variable A we (slightly) abuse notation and denote by A also the probability distribution on the support of this variable. We write $A \in D$ to denote that A is drawn from domain D. We use U_t to denote the uniform distribution on t-bit binary strings. We write $x \leftarrow A$ to denote the random variable A assuming the value x. We will rely on the following fact about independent random variables, which is proved in the full version of [9].

Lemma 1 *Let A, B be independent random variables and consider a sequence $V_1, \ldots, V_m$ of random variables, where for some function ϕ, $V_i = \phi(V_1, \ldots, V_{i-1}, C_i)$, with each C_i being either A or B. Then A and B are independent conditioned on $V_1, \ldots, V_m$.*

2.1 Min-Entropy and Average Min-Entropy

The statistical distance between two random variables A and B over a finite domain Ω is $\mathsf{SD}\big(A,\ B\big) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr[A=\omega] - \Pr[B=\omega]|$. Two variables are ε-close if their statistical distance is at most ε. The min-entropy of a random variable A is $H_\infty(A) = -\log(\max_x \Pr[A=x])$. The notion of average min-entropy, formalized in [7], captures the remaining unpredictability of the random variable A conditioned on the value of another random variable B. Formally,

$$\begin{aligned}\tilde{H}_\infty(A|B) &= -\log\left(\mathbb{E}_{y\leftarrow B}\left[\max_x \Pr\left[A=x \mid B=y\right]\right]\right)\\ &= -\log\left(\mathbb{E}_{y\leftarrow B}\left[2^{-H_\infty(A|B=y)}\right]\right)\end{aligned}$$

We will rely on the following useful properties of average min-entropy.

Lemma 2 ([7]) *Let A, B, C be random variables. If B has at most 2^λ possible values, then $\tilde{H}_\infty(A|(B,C)) \geq \tilde{H}_\infty(A|C) - \lambda$.*

Lemma 3 *Let A be a random variable with domain Ω, and U the random variable describing a uniformly sampled element from Ω; and let B a random variable. For any $\varepsilon \in [0,1]$, if $\mathsf{SD}\big((A,B),\ (U,B)\big) \leq \varepsilon$, then $\tilde{H}_\infty(A|B) \geq -\log\left(\frac{1}{|\Omega|} + \varepsilon\right)$.*

This lemma follows directly from the definition of average min-entropy; we omit the proof here.

2.2 Seeded Extractors

Definition 1 ([19]) *A function $Ext : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$ is a (worst-case) (k,ε)-strong extractor if for every random variable $A \in \{0,1\}^n$, such that, $H_\infty(A) \geq k$, it holds that, $\mathsf{SD}\big((Ext(A,S),S),\ (U_m,S)\big) \leq \varepsilon$, where S is uniform on $\{0,1\}^r$.*

Dodis et al. [7] generalized the definition above to the setting of average min-entropy, and showed the following generalized variant of the leftover hash lemma, stating that any family of pairwise independent hash functions is an average-case strong extractor.

Definition 2 *A function $Ext : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^m$ is an average-case (k,ε)-strong extractor if for all pairs of random variables A and B, such that, $A \in \{0,1\}^n$ and $\tilde{H}_\infty(A|B) \geq k$, it holds that, $\mathsf{SD}\big((Ext(A,S),S,B),\ (U_m,S,B)\big) \leq \varepsilon$, where S is uniform on $\{0,1\}^r$.*

Lemma 4 *Assume $\left\{H_x : \{0,1\}^n \to \{0,1\}^l\right\}$ is a family of universal hash functions. Then for any random variables A and B, such that $A \in \{0,1\}^n$ and $\tilde{H}_\infty(A|B) \geq m$, $\mathsf{SD}\big((H_x(A),X,B),\ (U_l,X,B)\big) \leq \varepsilon$ whenever $l \leq m - 2\log\frac{1}{\varepsilon}$.*

2.3 Two-Source Extractors

The extractors defined in the last section require the use of a short but *truly random* seed, which sometimes might be hard to obtain. The notion of two-source extractor [21, 22, 5] eliminates the use of a truly random seed, and instead extracts random bits from two *independent* sources of randomness.

Definition 3 *A function* $Ext2 : (\{0,1\}^t)^2 \to \{0,1\}^m$ *is a (worst-case)* (s, ε) *two-source extractor, if for all independent random variables* $A, B \in \{0,1\}^t$ *with min-entropy* s*, it holds that* $\mathsf{SD}\big(Ext2(A,B),\ U_m\big) \leq \varepsilon$.

Definition 4 *A function* $Ext2 : (\{0,1\}^t)^2 \to \{0,1\}^m$ *is an average-case* (s, ε)*-two source extractor, if for all random variables* $A, B \in \{0,1\}^t$ *and* C*, such that, conditioned on* C*,* A *and* B *are independent and have average min-entropy* s*, it holds that* $\mathsf{SD}\big((Ext2(A,B), C),\ (U_m, C)\big) \leq \varepsilon$.

It follows from the same proof of Lemma 2.3 in [7] that any worst-case two-source extractor is also an average-case two-source extractor.

Lemma 5 *For any* $\delta > 0$*, if* $Ext2 : (\{0,1\}^t)^2 \to \{0,1\}^m$ *is a (worst-case)* $(s - \log \frac{1}{\delta}, \varepsilon)$*-two-source extractor, then* $Ext2$ *is also an average-case* $(s, \varepsilon + 2\delta)$*-two-source extractor.*

2.4 Hash Proof Systems

Hash proof systems were introduced by Cramer and Shoup [6]. We briefly recall the presentation in [17], which views the hash proof systems as key encapsulation mechanisms.

Smooth Projective Hashing. All the notations below should be thought of as relying on an implicit security parameter (and maybe some other system parameters, such as the underlying algebraic groups). Let $\mathcal{SK}, \mathcal{PK}$ be the domains of secret and public keys, let $\mathcal{K}$ be the space of encapsulated symmetric keys, $\mathcal{C}$ be the space of ciphertexts and $\mathcal{V} \subset \mathcal{C}$ be the space of valid ciphertexts.

Let $F = \{F_{sk} : \mathcal{C} \to \mathcal{K}\}_{sk \in \mathcal{SK}}$ be a collection of hash functions with domain $\mathcal{C}$ and range $\mathcal{K}$, and let $\mu : \mathcal{SK} \to \mathcal{PK}$ be a projection function. Let $\mathcal{F}$ be the construction which is described by all these sets, $\mathcal{F} = (\mathcal{SK}, \mathcal{PK}, \mathcal{C}, \mathcal{V}, \mathcal{K}, F, \mu)$.

Definition 5 *The construction* $\mathcal{F}$ *is a projective hash family if for all* $v \in \mathcal{V}$*, and for all* $sk_1, sk_2 \in \mathcal{SK}$ *such that* $\mu(sk_1) = \mu(sk_2)$*, it holds that* $F_{sk_1}(v) = F_{sk_2}(v)$.

In other words, for all indexes $sk \in \mathcal{SK}$, the actions of F_{sk} on elements in $\mathcal{V}$ are uniquely determined by $\mu(sk)$. On the other hand, for elements not in $\mathcal{V}$, we require the hash function F_{sk} to behave almost "randomly". Formally,

Definition 6 *The construction* $\mathcal{F}$ *is* ε*-smooth, if it holds that*

$$\mathsf{SD}\big((pk, c, F_{sk}(c)),\ (pk, c, k)\big) \leq \varepsilon,$$

where $sk \in \mathcal{SK}$*,* $c \in \mathcal{C}/\mathcal{V}$*, and* $k \in \mathcal{K}$ *are sampled uniformly at random and* $pk = \mu(sk)$.

Hash Proof System. A hash proof system is roughly a construction of smooth-projective hash functions with several efficient associated algorithms. Specifically, we assume an efficient parameter-generating algorithm $Param$ that given the security parameter outputs the description of $\mathcal{F} = (\mathcal{SK}, \mathcal{PK}, \mathcal{C}, \mathcal{V}, \mathcal{K}, F, \mu)$, such that $\mathcal{V}$ is an NP language and there is an efficient algorithm for sampling $c \leftarrow \mathcal{V}$ together with a witness w. We also assume that there are efficient algorithms for sampling $sk \leftarrow \mathcal{SK}$ and $c \leftarrow \mathcal{C} \setminus \mathcal{V}$.

We also have two algorithms for computing the hash function F_{sk}. One is a *private evaluation algorithm* $Priv(sk, c)$ that on input a secret key $sk \in \mathcal{SK}$ and a ciphertext $c \in \mathcal{C}$, outputs the encapsulated key $k = F_{sk}(c)$. The other is a *public evaluation algorithm* Pub that computes the same given the public key, but only on valid ciphertexts and only when it is also given a witness of validity. Namely, for every $sk \in SK$ and $pk = \mu(sk)$ and for every $c \in \mathcal{V}$ with witness w, it holds that $Pub(pk, c, w) = F_{sk}(c)$.

Cramer and Shoup noted that a hash proof system immediately implies a KEM mechanism, where key-generation consists of running the parameter generating routine, then choosing a random secret key $sk \leftarrow \mathcal{SK}$ and computing the corresponding public key $pk = \mu(sk)$. Encapsulating a key is done by choosing at random $c \leftarrow \mathcal{V}$ together with a witness w. Then the ciphertext is c and the corresponding encapsulated key is computed by the sender using the public evaluation algorithm, setting $k = Pub(pk, c, w)$. On the receiving side, the same key is recovered using the private evaluation algorithm, setting $k = Priv(sk, c)$. Security of this scheme follows from the smoothness of the construction, in conjunction with the hardness subset membership problem, as defined below.

Subset Membership Problem. A hash proof system as above is said to have a hard subset membership problem if a randomly generated valid ciphertext is computationally indistinguishable from a randomly generated invalid ciphertext. Formally, the following two ensembles are indistinguishable

$$\begin{aligned} \mathsf{VALID} &= \{\mathcal{F} = (\mathcal{SK}, \mathcal{PK}, \mathcal{C}, \mathcal{V}, \mathcal{K}, F, \mu) \leftarrow Param(1^n),\ c \leftarrow \mathcal{V} \ :\ (\mathcal{F}, c)\}_{n \in N} \\ \mathsf{INVALID} &= \{\mathcal{F} = (\mathcal{SK}, \mathcal{PK}, \mathcal{C}, \mathcal{V}, \mathcal{K}, F, \mu) \leftarrow Param(1^n),\ c \leftarrow \mathcal{C} \setminus \mathcal{V} \ :\ (\mathcal{F}, c)\}_{n \in N} \end{aligned}$$

3 Entropic Security against After-the-Fact Leakage

Roughly speaking, we say that an encryption scheme is entropic leakage resilient if a message M with high min-entropy still "looks random" to the adversary even after it sees an encryption of it and some leakage information (even if this leakage was obtained after seeing the ciphertext). This is formulated by postulating the existence of a simulator that generates a view, which on one hand is indistinguishable from the real adversary view and on the other hand still leaves M with high min-entropy.

More formally, we define two games, one "real" and the other "simulated". Both games depend on several parameters: k is the a-priory min-entropy of the message, and $\ell_{\text{pre}}, \ell_{\text{post}}$ control the amount of leakage in various parts of the games (namely the pre- and post- challenge-ciphertext leakage). All of these parameters are of course functions of the security parameter n. (For simplicity, in the definition below we assume that the message M is a uniform random k-bit string. This is all we need for our application in Section 4 to the only-computation-leak model, and extending the definition to arbitrary high-min-entropy distributions is easy.)

The "real" game. Given the parameters $(k, \ell_{\rm pre}, \ell_{\rm post})$ and the encryption scheme $\Psi = (Gen, Enc, Dec)$, the real game is defined as follows:

Key Generation: The challenger chooses at random a message $m \leftarrow U_k$. The challenger also generates $(sk, pk) \leftarrow Gen(1^n)$, and sends pk to the adversary.

Pre-Challenge Leakage: The adversary makes a pre-challenge leakage query, specifying a function $f^{\rm pre}(\cdot)$. If the output length of $f^{\rm pre}$ is at most $\ell_{\rm pre}$ then the challenger replies with $f^{\rm pre}(sk)$. (Else the challenger ignores the query.)

Challenge: Upon a challenge query, the challenger encrypts the message m and sends the ciphertext $c = Enc(pk, m)$ to the adversary.

Post-Challenge Leakage: The adversary makes a post-challenge leakage query, specifying another function $f^{\rm post}(\cdot)$. If the output length of $f^{\rm post}$ is at most $\ell_{\rm post}$ then the challenger replies with $f^{\rm post}(sk)$. (Else the challenger ignores the query.)

We let $\mathsf{View}^{\rm rl}_A(\Psi) = (\text{randomness}, pk, f^{\rm pre}(sk), c, f^{\rm post}(sk))$ be the random variable describing the view of the adversary A in the game above, and by $M^{\rm rl}$ we denote the message that was chosen at the onset of this game. (We view them as correlated random variables, namely when we write $(M^{\rm rl}, \mathsf{View}^{\rm rl}_A(\Psi))$ we mean the joint distribution of the message $M^{\rm rl}$ and A's view in a real game with $M^{\rm rl}$).

The "simulated" game. In the simulated game we replace the challenger from above by a simulator Simu that interacts with A in any way that it sees fit. Simu gets a uniformly chosen message $M^{\rm sm}$ as input, and it needs to simulate the interaction conditioned on this $M^{\rm sm}$. The view of A when interacting with S is denoted $\mathsf{View}^{\rm sm}_A(\mathsf{Simu})$.

Below we say that Ψ is entropic leakage-resilient (with respect to all the parameters) if on one hand the distributions $\mathsf{View}^{\rm rl}_A(\Psi)$, $\mathsf{View}^{\rm sm}_A(\mathsf{Simu})$ are indistinguishable even given the message M, and on the other hand $M^{\rm sm}$ has high min-entropy given $\mathsf{View}^{\rm sm}_A(\mathsf{Simu})$.

Definition 7 *Let $k, \ell_{\rm pre}, \ell_{\rm post}$ be parameters as above, and let δ be another "slackness parameter." A public-key encryption scheme $\Psi = (Gen, Enc, Dec)$ is entropic leakage resilient with respect to these parameters if there exists a simulator Simu, such that, for every $\mathcal{PPT}$ adversary A the following two conditions hold:*

- *The two ensembles $\big(M^{\rm rl}, \mathsf{View}^{\rm rl}_A(\Psi)\big)$, $\big(M^{\rm sm}, \mathsf{View}^{\rm sm}_A(\mathsf{Simu})\big)$ (indexed by the security parameter) are computationally indistinguishable.*
- *The average min-entropy of $M^{\rm sm}$ given $\mathsf{View}^{\rm sm}_A(\mathsf{Simu})$ is*

$$\tilde{H}_\infty(M^{\rm sm} \mid \mathsf{View}^{\rm sm}_A(\mathsf{Simu})) \geq k - \ell_{\rm post} - \delta.$$

Intuitively, in the simulated game the message $M^{\rm sm}$ retains its initial entropy, except for the $\ell_{\rm post}$ post-challenge leakage bits and possibly also some "overhead" of δ bits. And since the simulated game cannot be distinguished from the real game, then $M^{\rm rl}$ has the same number of pseudo-entropy bits also in the real game.

What happened to ℓ_{pre}? Note that the min-entropy of M^{sm} is only reduced by the amount of the post-challenge leakage (and "overhead") irrespective of the pre-challenge leakage. This is reminiscent of the prior results that can tolerate pre-challenge leakage while maintaining semantic security (hence "not losing any entropy" of the message). Indeed, the security notions from [1, 17] can be obtained as a special case of our definition with $\ell_{\text{post}} = \delta = 0$.

Adaptiveness. It was pointed out in [1] that the pre-challenge leakage can be made adaptive without effecting the definition. The same holds also for the post-challenge leakage.

3.1 Constructing Entropic Leakage-Resilient Scheme

We show that the generic construction of Naor and Segev for pre-challenge leakage resilient encryption from hash proof systems [17], is actually entropic secure against bounded after-the-fact leakage. The encryption algorithm (overly simplified) samples a valid ciphertext c of the hash proofs system, and uses the key encapsulated in c to hide the message. To show entropic security, the entropic simulator proceed the same as the encryption algorithm except that it uses invalid ciphertexts. It follows from the indistinguishability of the valid and invalid ciphertexts that the real and the simulated games are indistinguishable. Furthermore, due to smoothness the key encapsulated in an invalid ciphertext has high min-entropy, and hence the message is well "hidden", and has high average min-entropy.

In more details, we need an ε-smooth hash proof system $\mathcal{F} = (\mathcal{SK}, \mathcal{PK}, \mathcal{C}, \mathcal{V},$
$\mathcal{K}, F, \mu)$, where the symmetric encapsulated keys are assumed (w.l.o.g.) to be just t_1-bit strings, $\mathcal{K} = \{0,1\}^{t_1}$. We also need a function $Ext : \{0,1\}^{t_1} \times \{0,1\}^{t_2} \to \{0,1\}^{t_3}$ which is an average-case (t_4, δ) strong extractor. Namely, it has t_1-bit inputs, t_2-bit seeds and t_3-bit outputs, and for a random seed and input with t_4 bits of min entropy, the output is δ-away from a uniform t_3-bit string. Then, the encryption scheme $\Psi = (Gen, Enc, Dec)$ proceeds as follows:

Key Generation: The key generation algorithm, on input a security parameter 1^n, generates an instance of a projective hash family $\mathcal{F} = (\mathcal{SK}, \mathcal{PK}, \mathcal{C}, \mathcal{V}, \mathcal{K}, F,$
$\mu) \leftarrow Param(1^n)$, samples a secret key $sk \leftarrow \mathcal{SK}$, and computes the corresponding public key $pk = \mu(sk)$.

Encryption: The encryption algorithm, on input a message $m \in \{0,1\}^{t_3}$, samples a valid ciphertext together with a corresponding witness $(c, w) \leftarrow \mathcal{V}$, and computes the encapsulated key k using the public evaluation algorithm, i.e., $k = Pub(pk, c, w)$. It then samples a random seed $s \in \{0,1\}^{t_2}$, and computes $\psi = Ext(k, s) \oplus m$. Finally, it outputs the ciphertext $\hat{c} = (c, s, \psi)$.

Decryption: The decryption algorithm on input a ciphertext (c, s, ψ), computes the encapsulated key k using the private evaluation algorithm, i.e., $k = Priv(sk, c)$, and outputs the message $m = Ext(k, s) \oplus \psi$.

It follows using the same proof as in [17] that the encryption scheme Ψ is a correct public-key encryption scheme. Namely the decryption algorithm always recovers the original message m correctly. Next, we proceed to prove the entropic leakage resilience of Ψ against after-the-fact leakage.

Lemma 6 *The public-key encryption scheme Ψ from above is entropic leakage-resilient with respect*

to leakage $\ell_{\text{pre}}, \ell_{\text{post}}$ and "overhead" δ', as long as these parameters satisfy the following constraints:

$$\ell_{\text{pre}} \leq \log\left(\frac{1}{\frac{1}{|\mathcal{K}|}+\varepsilon}\right) - t_4 \quad \textit{and} \quad \delta' \leq t_3 - \log\frac{1}{2^{-t_3}+\delta}$$

To interpret these parameters, it is useful to think of a "very smooth" hash proof system ($\varepsilon \ll 1/|\mathcal{K}| = 2^{-t_1}$), and a very good extractor that can work with inputs that have min-entropy $t_4 \ll \log|\mathcal{K}| = t_1$ and produces outputs whose distance from uniform t_3-bit strings is $\delta < 2^{-t_3}$. For such building blocks we can tolerate pre-challenge leakage of $\ell_{\text{pre}} \approx t_1 - t_4 = t_1(1-o(1))$, and our overhead is $\delta' < 1$ bits.

Proof: To prove Lemma 6 we need to describe a simulator, whose answers to the adversary are indistinguishable from the real game but at the same time leave many bits of min-entropy in the message m.

In our case, the simulator S proceeds almost identically to the challenger in the real game, except that to generate the ciphertext $\hat{c} = (c, s, \psi)$ it samples an *invalid ciphertext* for the hash-prof system, $c \leftarrow \mathcal{C}/\mathcal{V}$, then it computes $k = Priv(sk, c)$ using the secret key sk that it knows, and outputs the ciphertext $\hat{c} = (c, s, \psi)$, where $\psi = Ext(k, s) \oplus m$.

It follows directly from the indistinguishability of the valid and invalid ciphertexts of the hash proof system that the simulated view is indistinguishable from the real one even given the message m. It only remains to show the min-entropy condition.

On a high-level, the proof consists of two steps. The first step shows that conditioned on all the information that the adversary receives till the end of the Challenge Phase, the message m still has high average min-entropy, namely at least $t_3 - \delta'$.

To see this, note that by ε-smoothness the encapsulated key k has almost t_1 bits of min-entropy even given pk and c, and therefore almost $t_1 - \ell_{\text{pre}}$ bits of min-entropy even given pk, c and the pre-challenge leakage. Specifically k has at least $\log\left(\frac{1}{\frac{1}{|\mathcal{K}|}+\varepsilon}\right) - \ell_{\text{pre}} \geq t_4$ bits of min-entropy, and therefore the bits extracted from k using the extractor Ext are statistically close to random (even given pk, c the pre-challenge leakage and the seed s). Thus the message m is δ-close to a uniform t_3-bit string, even given pk, c, the pre-challenge leakage, the seed s, and the value ψ. (So far this is exactly the same argument as in the proof of the Naor-Segev construction.) Hence upto this phase, the message m has at least $t_3 - \delta'$ bits of min entropy.

Next, by further relying on the fact that the post-challenge leakage is bounded by ℓ_{post} bits, the min-entropy of m is reduced by at most this much, so it retains at least $t_3 - \ell_{\text{post}} - \delta'$ bits of average min-entropy.

. ■

4 Semantic Security in a Split-State Model

We next demonstrate how Definition 7 can be used in a "higher level protocol". Specifically, we consider a split-state model, where the secret state of the cryptosystem at hand is partitioned to a few parts and the adversary can obtain leakage of its choice on every part separately but not a global leakage function from the entire secret state.

This model is often used in conjunction with the only-computation-leaks axiom (OCL) of Micali and Reyzin [16]. In our case we talk only about CPA security and there is no decryption oracle,

hence after-the-fact there isn't any computation to leak from and talking about only-computation-leaks does not make sense. (An extension of this construction to get CCA-security may be applicable to the OCL model, but this is beyond the scope of the current work.)

Definition 8 *A 2-split-state encryption scheme is a public-key encryption scheme* $\Pi = (Gen, Enc, Dec)$ *that has the following structure:*

- *The secret key consists of a pair of strings* $S = (S_1, S_2)$*, and similarly the public key consists of a pair* $P = (P_1, P_2)$*.*
- *The key generation algorithm Gen consists of two subroutines* Gen_1 *and* Gen_2*, where* Gen_i *for* $i \in \{1, 2\}$ *outputs* (P_i, S_i)*.*
- *The decryption algorithm Dec also consists of two partial decryption subroutines* Dec_1 *and* Dec_2 *and a combining subroutine Comb. Each* Dec_i *takes as input the ciphertext and* S_i *and outputs partial decryption* t_i*, and the combining subroutines Comb takes the ciphertext and the pair* (t_1, t_2) *and recovers the plaintext.*

In the split-state model, we assume that information is leaked independently from the two parts. Semantic security for such a 2-split-state scheme in the presence of After-the-Fact Leakage in this model is defined below. Let $\ell_{\text{pre}}, \ell_{\text{post}}$ be parameters as before, and we consider the following game:

Key Generation: The challenger chooses $r_1, r_2 \in \{0,1\}^*$ at random, generates $(sk_b, pk_b) \leftarrow Gen(1^n, r_b)$ for $b = 1, 2$, and sends (pk_1, pk_2) to the adversary.

Pre-Challenge Leakage: The adversary makes an arbitrary number of leakage queries $(f^{\text{pre}}_{1,i}, f^{\text{pre}}_{2,i})$ adaptively. Upon receiving the i^{th} leakage query the challenger sends back $(f^{\text{pre}}_{1,i}(sk_1), f^{\text{pre}}_{2,i}(sk_2))$, provided that the total output length of all the pre-challenge queries so far does not exceed ℓ_{pre} in each coordinate. (Otherwise the challenger ignores the query.)

Challenge: The adversary sends two messages $m_0, m_1 \in \{0,1\}^n$. The challenger chooses a random bit σ, encrypts the message m_σ, and returns the ciphertext $c = Enc(pk, m_\sigma)$.

Post-Challenge Leakage: The adversary can submit an arbitrary number of leakage queries $(f^{\text{post}}_{1,i}, f^{\text{post}}_{2,i})$ adaptively. Upon receiving the i^{th} leakage query the challenger sends back $(f^{\text{post}}_{1,i}(sk_1), f^{\text{post}}_{2,i}(sk_2))$, provided that the total output length of all the post-challenge queries so far does not exceed ℓ_{post} in each coordinate. (Otherwise the challenger ignores the query.)

Output: The adversary outputs a bit σ'.

Definition 9 *A 2-split-state encryption scheme* $\Psi = (Gen, Enc, Dec)$ *is* resilient to $(\ell_{\text{pre}}, \ell_{\text{post}})$ leakage in the split-state model*, if for every* $\mathcal{PPT}$ *adversary A that participates in an experiment as above, there is a negligible function* negl *such that* $\Pr[\sigma' = \sigma] < 1/2 + \mathsf{negl}(n)$*.*

4.1 Our Construction

As defined above, our 2-split-state scheme maintains a split secret key (S_1, S_2) (and a corresponding split public key (P_1, P_2)), where S_i is generated by Gen_i and used by Dec_i. Due to the the restriction on the leakage in the split-state model, the adversary can never obtain leakage on S_1 and S_2 jointly, so even after leakage we can hope that each of the two parts still has sufficient entropy *and moreover*

they are independent. Hence we can use two-source extractors to get a close-to-uniform string from these two parts, and use it to mask the message.

In the scheme below, we do not try to extract from the secret keys themselves, but rather in each encryption we encrypt two random strings, one under each of the keys, and extract randomness from these two ephemeral random strings. We argue that if the two copies are implemented using an entropic leakage-resilient scheme, then we get semantic security in the split-state model. Intuitively, the reason is that the entropic security ensures that the two ephemeral strings still have high (pseudo)entropy even given the leakage, and the split-state model ensures that they are independent, so the two-source extraction should give us what we want.

The formal proof roughly follows this intuitive reasoning, with just one additional complication, related to adaptivity: In the post-challenge leakage, the adversary can choose the leakage functions from the two key parts after it already saw the value that was extracted from the two random strings, causing a circularity in the argument. We solve this issue essentially by "brute force": We argue below that if the extracted value has only u bits, then the adaptivity issue can increase the advantage of the adversary by at most a 2^u factor, and set our parameters to get around this factor.[1]

The construction. Let n be the security parameter, and let u be the bit-length of the messages that we want to encrypt. Also let $t, v, \ell_{\text{pre}}, \ell_{\text{post}}$ be some other parameters (to be defined later). Let $\Psi = (Gen^{\mathsf{Ent}}, Enc^{\mathsf{Ent}}, Dec^{\mathsf{Ent}})$ be a entropic secure encryption for t-bit messages, resilient to leakage $(\ell_{\text{pre}}, \ell_{\text{post}})$, with overhead of one bit.

Also, let $Ext2 : \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^u$ be an average-case (v, ε)-two-source extractor, with $\varepsilon = 2^{-u-\omega(\log n)}$. Namely both inputs to $Ext2$ are of length t, and as long as they are independent and both have more than v min entropy, the output of $Ext2$ is at most ε away from a uniform u-bit string.[2] Given these ingredients, our 2-split-state encryption scheme $\Pi = (Gen, Enc, Dec)$ proceeds as follows:

Key Generation: The key generation algorithm runs two subroutines Gen_1 and Gen_2, where Gen_i for $i \in \{1,2\}$ on input 1^n generates a public and secret key pair $(S_i, P_i) \leftarrow Gen^{\mathsf{Ent}}(1^n)$ of the entropic encryption scheme Ψ. The public key is $P = (P_1, P_2)$ and the secret key is $S = (S_1, S_2)$.

Encryption: The encryption algorithm, on input a message $m \in \{0,1\}^u$, chooses two random strings $x_1, x_2 \in \{0,1\}^t$ and encrypts the two strings using the two public keys P_1 and P_2 respectively; set $c_i = Enc^{\mathsf{Ent}}(P_i, x_i)$. Then, it computes $\psi = Ext2(x_1, x_2) \oplus m$, and outputs the ciphertext $\hat{c} = (c_1, c_2, \psi)$.

Decryption: The decryption algorithm, on input a ciphertext (c_1, c_2, ψ), executes the following three subroutines sequentially.

- The subroutine Dec_1 decrypts c_1 using S_1 and outputs the plaintext $x_1 = Dec^{\mathsf{Ent}}(S_1, c_1)$.
- The subroutine Dec_2 decrypts c_2 using S_2 and outputs the plaintext $x_2 = Dec^{\mathsf{Ent}}(S_2, c_2)$.
- The subroutine $Comb$ on input x_1, x_2 and ψ, outputs the message $M = Ext2(x_1, x_2) \oplus \psi$.

[1] We remark that we *do not* make exponential hardness assumptions to achieve this. See proof of Claim 8 for more details.

[2] Note that we set ε so that it remain negligible even if we multiply it by 2^u, this is needed for the adaptivity issue in the proof.

Lemma 7 *The 2-split-state scheme* Π *from above is semantically secure with respect to leakage* $(\ell'_{\text{pre}}, \ell'_{\text{post}})$ *in the split-state model, as long as the parameters satisfy the following constraints:*

$$\ell'_{\text{pre}} \leq \ell_{\text{pre}} \quad \textit{and} \quad \ell'_{\text{post}} \leq \min(\ell_{\text{post}} - u,\ t - v - 1).$$

Proof: We need to show that $\mathcal{PPT}$ adversaries only have negligible advantage in guessing the choice bit σ in the semantic security game. To that end, fix a semantic-security adversary $\mathcal{A}^{\mathsf{ss}}$, and let Simu be the entropic simulator that exists for the underlying entropic scheme Ψ[3]. We now consider the hybrid experiments hyb_1 and hyb_2, which are defined as follows:

Hybrid hyb_1: In this hybrid, the challenger generates the ciphertext c_2 and answers leakage queries against S_2 just as in the real game. However, it uses the entropic simulator Simu to generate the ciphertext c_1 and to answer leakage queries against S_1.

In more details, the challenger chooses x_1, x_2 at random, then generates (S_2, P_2) using the key-generation of Ψ, but it gets P_1 by running $\mathsf{Simu}(x_1)$. (Recall that the entropic simulator expects a random plaintext string in its input.) Then to answer a pre-challenge query $(f^{\text{pre}}_{1,i}, f^{\text{pre}}_{2,i})$, the challenger forward $f^{\text{pre}}_{1,i}$ to Simu and gets the answer from it, computes the answer $f^{\text{pre}}_{2,i}(S_2)$ by itself, and send both answer to $\mathcal{A}^{\mathsf{ss}}$.

When $\mathcal{A}^{\mathsf{ss}}$ makes a challenge query (m_0, m_1), the challenger asks Simu for the first ciphertext c_1, and computes c_2 by itself $c_2 = Enc(P_2, x_2)$. (Recall again that Simu was run with input x_1, so the ciphertext that it returns is supposed to simulate an encryption of x_1.)

Next, **the challenger makes a direct post-challenge leakage query to** Simu, querying with the function $h_1(S_1) = Ext2(Dec(S_1, c_1), x_2)$ (that has u bits of output). Getting some answer r', the challenger just discards that answer, instead computing $r = Ext2(x_1, x_2)$, choosing a random bit σ, setting $\psi = r \oplus m_\sigma$ and sending (c_1, c_2, ψ) to $\mathcal{A}^{\mathsf{ss}}$.

After that, post-challenge queries of $\mathcal{A}^{\mathsf{ss}}$ are handled just like the pre-challenge queries, with the challenger asking Simu for the first part of the answer (for the query against S_1) and computing the answer to the query against S_2 by itself.

Hybrid hyb_2: In this hybrid the challenger still chooses x_1, x_2 at random, but now both parts of the game are handled by the simulator, running as $\mathsf{Simu}(x_1)$ to answer the first part of all the queries (and to get c_1) and as $\mathsf{Simu}(x_2)$ to answer the second part of all the queries (and to get c_2).

The challenger makes direct post-challenge queries to both copies of the simulator, asking the first for $r' = h_1(S_1) = Ext2(Dec(S_1, c_1), x_2)$ and the second for $r'' = h_2(S_2) = Ext2(x_1, Dec(S_2, c_2))$. The challenger still ignores both answers, computing instead $r = Ext2(x_1, x_2)$ and setting the ψ component of the Π-ciphertext as $r \oplus m_\sigma$.

Before proceeding with the proof, we point out that the direct post-challenge leakage queries that the challenger makes are expected to return the same value that the challenger computes itself, $r' = r'' = r$. (Indeed we prove below that they almost always do). The reason that the challenger still makes them is to ensure that the entropic simulators see the same queries in these hybrids as in the reductions that we use below. One consequence of these direct queries is that the entropic simulators need to answer more post-challenge queries than what the semantic-security adversary asks. Specifically, it needs to answer u more bits, hence the constraint $\ell'_{\text{post}} \leq \ell_{\text{post}} - u$.

[3]Our Definition 7 has only one pre- and one post-challenge query. Below we assume for convenience that the entropic-security adversary can make adaptive queries, it was noted in [1] that these definition are equivalent.

We now prove that the event $\sigma' = \sigma$ holds in the hybrids with essentially the same probability as in the real game, by reducing to the indistinguishability property of the entropic simulator.

The hybrid hyb_1. Assume toward contradiction that the event $\sigma' = \sigma$ happens in the real game with probability which is larger than in the first hybrid hyb_1 by a noticeable amount ρ. We describe an entropic adversary $\mathcal{A}^{\mathsf{ent}}$ and distinguisher $\mathcal{D}^{\mathsf{ent}}$ that break this indistinguishability property. (In fact, for the same entropic adversary $\mathcal{A}^{\mathsf{ent}}$ we describe two distinguishes $\mathcal{D}_1^{\mathsf{ent}}$, $\mathcal{D}_2^{\mathsf{ent}}$, and prove that at least one of them has advantage $\rho/2$ or more.)

- The entropic adversary $\mathcal{A}^{\mathsf{ent}}$, on input public key P_1, chooses (P_2, S_2) and x_2 in the same way as the hyb_1 challenger, and sends (P_1, P_2) to the semantic-security adversary $\mathcal{A}^{\mathsf{ss}}$. It then proceeds similarly to the hyb_1 challenger, answering the first part of every query using its oracle and computing the answer to the second part by itself.

  The only difference between $\mathcal{A}^{\mathsf{ent}}$ and the hyb_1 challenger is in the way that the ψ component of the ciphertext is computed. Once $\mathcal{A}^{\mathsf{ent}}$ gets c_1 from its oracle and computes $c_2 = Enc(P_2, x_2)$, it makes a post-challenge leakage query to its oracle asking for $r' = h_1(S_1) = Ext2(Dec(S_1, c_1), x_2)$. Since $\mathcal{A}^{\mathsf{ent}}$ does not have x_1, it does not discard the answer but rather uses it for setting $\psi = r' \oplus m_\sigma$.

- The first distinguisher $\mathcal{D}_1^{\mathsf{ent}}$ gets the view of $\mathcal{A}^{\mathsf{ent}}$, which includes x_2 and r', and also the string x_1 (which was supposed to be encrypted in c_1). $\mathcal{D}_1^{\mathsf{ent}}$ simply verifies that $r' = Ext2(x_1, x_2)$, outputting 1 if they are equal and 0 otherwise.

- The second distinguisher $\mathcal{D}_2^{\mathsf{ent}}$ gets the view of $\mathcal{A}^{\mathsf{ent}}$, which includes σ and σ', and outputs 1 if they are equal and 0 otherwise.

Clearly, if the oracle of $\mathcal{A}^{\mathsf{ent}}$ is the real encryption scheme Ψ then the transcript that $\mathcal{A}^{\mathsf{ss}}$ sees is identical to the real semantic-security game. In particular, the ciphertext c_1 is indeed an encryption of x_1, and therefore we have $r' = Ext2(x_1, x_2)$ with probability 1.

If the oracle of $\mathcal{A}^{\mathsf{ent}}$ is the simulator $\mathsf{Simu}(x_1)$, then we have two possible cases: either the event $r' \neq Ext2(x_1, x_2)$ happens with probability at least $\rho/2$, or it happens with smaller probability. In the first case, the distinguisher $\mathcal{D}_1^{\mathsf{ent}}$ clearly has an advantage at least $\rho/2$ in distinguishing between the real scheme Ψ and the simulator Simu.

In the second case, the transcript that $\mathcal{A}^{\mathsf{ss}}$ sees is the same as in the hybrid hyb_1, except for an event of probability less than $\rho/2$. Since the probability of $\sigma = \sigma'$ in the real game is larger by ρ than this probability in hyb_1, then it is larger by more than $\rho/2$ than this probability in the interaction with $\mathcal{A}^{\mathsf{ent}}$. Hence the distinguisher $\mathcal{D}_2^{\mathsf{ent}}$ has advantage more than $\rho/2$. ∎

The hybrid hyb_2. The proof of indistinguishability between hyb_1 and hyb_2 is essentially the same as the proof of indistinguishability between the real game and hyb_1, and is omitted here. ∎

The advantage in hyb_2. Having shown that the probability of $\sigma = \sigma'$ in the second hybrid hyb_2 is negligibly close to the probability in the real game, we now proceed to bound it. For that purpose, we consider another mental experiment $\tilde{\mathsf{hyb}}$ as follows:

Hybrid $\tilde{\mathsf{hyb}}$: hybrid $\tilde{\mathsf{hyb}}$ proceeds the same as hyb_2, except that, in the Challenge Phase, instead of sending the adversary $\mathcal{A}^{\mathsf{ss}}$ the complete ciphertext $\hat{c} = (c_1, c_2, \psi)$, the challenger sends only c_1 and c_2, and defers sending ψ until after the Post-Challenge Leakage Phase.

Of course, the mental experiment $\tilde{\mathsf{hyb}}$ is very much distinguishable from hyb_2. Moreover, compared with the adversary in $\tilde{\mathsf{hyb}}$, the adversary in hyb_2 has the advantage of choosing the leakage functions in the Post-Challenge Leakage Phase based on ψ. Still, we argue that this advantage is limited, up to an exponential factor in u. Namely, we show in Claim 8 that if $\mathcal{A}^{\mathsf{ss}}$ has advantage α in guessing the bit σ in hyb^2, then there is another adversary $\tilde{A}$ that has advantage at least $\alpha/2^u$ in guessing the bit σ in the mental experiment $\tilde{\mathsf{hyb}}$.

Claim 8 *If for some $\alpha > 0$ there exists an adversary $\mathcal{A}^{\mathsf{ss}}$ for which $\Pr_{\mathsf{hyb}_2}[\sigma = \sigma'] \geq \frac{1}{2} + \alpha$, then there exists another adversary $\tilde{A}$ for which $\Pr_{\tilde{\mathsf{hyb}}}[\sigma = \sigma'] \geq \frac{1}{2} + \frac{\alpha}{2^u}$.*

Proof: We present a generic construction of $\tilde{A}$ given $\mathcal{A}^{\mathsf{ss}}$. The adversary $\tilde{A}$ in $\tilde{\mathsf{hyb}}$ runs $\mathcal{A}^{\mathsf{ss}}$ internally, and forward messages externally to the Challenger in $\tilde{\mathsf{hyb}}$. Except that in the Challenge Phase, $\tilde{A}$ randomly chooses some string $\psi' \in \{0,1\}^u$ and sends it to $\mathcal{A}^{\mathsf{ss}}$ in lieu of ψ. Later, when $\tilde{A}$ gets the "real ψ" from the challenger, it aborts if it guessed wrong, $\psi' \neq \psi$, and proceeds just like $\mathcal{A}^{\mathsf{ss}}$ if the guess was correct. Since the guess is correct with probability 2^{-u}, it follows that the advantage of $\tilde{A}$ is exactly $\alpha/2^u$. ∎

The advantage in $\tilde{\mathsf{hyb}}$. We are now ready to use the min-entropy property of the simulator S to prove that the advantage pf $\tilde{A}$ in $\tilde{\mathsf{hyb}}$ is at most 2ε. Since we set $\varepsilon = 2^{-u-\omega(\log n)}$, then by Claim 8 it follows that the advantage of $\mathcal{A}^{\mathsf{ss}}$ in hyb_2 is at most $2\varepsilon \cdot 2^u = 2^{1-\omega(\log n)} = \mathsf{negl}(n)$, as needed.

In the mental experiment $\tilde{\mathsf{hyb}}$, let Γ be the (partial) transcript of messages that $\tilde{A}$ receives till the end of the Post-Challenge Leakage Phase (i.e., before it gets ψ). We show that the average min-entropy of each of the two seeds x_1, x_2, conditioned on Γ is at least v. Let $\Gamma = (\Gamma_1, \Gamma_2)$, where Γ_1 denote the partial transcript including the public key P_1, the simulated encryption c_1 of x_1, and all the leakage on S_1, and Γ_2 the partial transcript including P_2, c_2 and all the leakage on S_2. By the entropic security of Ψ in the simulated game, and the fact that $\ell'_{\text{post}} \leq t - v - 1$, we have that $\tilde{H}_\infty(x_1|\Gamma_1) \geq t - \ell'_{\text{post}} - 1 \geq v$. Furthermore, since conditioned on Γ_1, x_1 and Γ_2 are independent, we get $\tilde{H}_\infty(x_1|\Gamma) \geq v$. Similarly, it also holds that $\tilde{H}_\infty(x_2|\Gamma) \geq v$.

Since both x_1, x_2 have min-entropy more than v, and furthermore, by Lemma 1, are independent conditioned on Γ (as in Γ no function computes on both x_1 and x_2), the output of the average-case (v, ε) two-source extractor $Ext2(x_1, x_2)$ is at most ε away from uniform. Therefore the two distribution $Ext2(x_1, x_2) \oplus m_0$ and $Ext2(x_1, x_2) \oplus m_1$ are at most 2ε apart (since each is at most ε away from uniform). Therefore the advantage of $\tilde{A}$ is at most 2ε. ∎

4.2 Instantiations and Parameters

Naor and Segev presented some instances of their construction [17] based on the DDH assumption (or DDH and the d-linear assumption), and the same constructions work for our case too. This gives entropic leakage resilient scheme Ψ with respect to any leakage ℓ_{pre}, ℓ_{post} and overhead 1, as long as ℓ_{pre} is bounded by $(1 - o(1))L' - 3t$, where L' and t are respectively the lengths of the secret key and plaintext of the scheme Ψ. Therefore, we only need to focus on instantiating the two-source extractor $Ext2$ with exponentially small error $\varepsilon = 2^{-u-\omega(\log n)}$ in the length u of the output. In the work of Bouragin [3] it was shown how to extract randomness from two independent sources with min-entropy rate slightly less than half.

Theorem 9 ([3]) *There exists a universal constant $\gamma < 1/2$ and a polynomial time computable function* $\mathsf{Bou} : \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^{u'}$ *that is a (v, ε)-two-source extractor, with $v = \gamma t$, $\varepsilon = 2^{-\Omega(u')}$, and $u' = \Omega(t)$.*

It follow from Lemma 5 that Bou is also an average-case extractor as needed. Furthermore, this construction lets us get two-source extractors with statistical distance as small as we want. Namely, to get $\varepsilon = 2^{-u-\omega(\log n)}$ we simply use it with u' sufficiently larger than u. Then we can truncate the output to length u without increasing the statistical distance, thus getting the parameters that we need.

Remark 1 *The scheme* Π *uses a two-source extractor. We show that the construction can be easily modified to use a c-source extractor, for any $c > 2$: instead of having two secret keys, maintain c secret keys $S_1, \ldots, S_c$; each key S_i is used to encrypt and decrypt a random seed X_i sampled independently, and the message is hidden using the random bits extracted from X_i's—we call it a c-split-state encryption scheme. It follows from the same proof as above that, this scheme is secure in the split-state model.*

Furthermore, since we know how to extract randomness from multiple independent random sources more efficiently, the c-split-state encryption scheme can tolerate more leakage than the 2-split-state scheme. For instance, it has been shown in [20, 2] that for any arbitrarily small constant $\delta > 0$, there exists an $O(1/\delta)$-source extractor that extracts from independent sources with min-entropy $k = n^\delta$, with error $\varepsilon = 2^{-k^{\Omega(1)}}$. Plugging in this extractor yields a c-split-state encryption scheme resilient to any $(\ell_{\text{pre}}, \ell_{\text{post}})$ leakage on each secret key S_i *satisfying $\ell'_{\text{post}} + (1 - o(1))\ell'_{\text{pre}} \leq (1 - o(1))L - 3u - \omega(\log n)$, which means almost all of each secret key can be leaked. We omit the details here.*

5 Conclusion and Future Work

In this paper, we study after-the-fact leakage for public-key encryption schemes. We show that a meaningful notion of security, namely, entropic security, can be achieved even at the presence of arbitrary (but bounded) leakage after the ciphertext is generated, and furthermore, the full fledged semantic security can be retained if considering some restricted form of leakage, namely a split-state model.

It is, of course, very interesting to explore other notions of security and other models in the context of after-the-fact leakage. For instance, Naor and Segev [17] showed that PKE that is semantically secure resilient to before-the-fact leakage can be transformed into a scheme that is CCA2-secure resilient to before-the-fact leakage, following the Naor-Yung "double encryption" paradigm [8, 18]. It is interesting to see if a similar transformation can be done even with after-the-fact leakage.

Furthermore, recently, there has been some developments in leakage resilient cryptography in the continuous leakage model. One question studied in [15, 11] is how to transform any circuit with a secret hard-coded, into another one that hides the secret even at the presence of arbitrary leakage during the computation of the circuit, *in the OCL model*. It would be interesting to investigate if their techniques can be applied to our scheme to make it secure even in the continuous leakage model.

Another interesting question is to handle leakage from the encryption randomness, not just the secret key. Perhaps the dense-model theorem from [10] can be used to prove ressitance at least to logarithmically many leakage bits.

Beyond just encryption, it is interesting to see if there are "natural" and useful relaxations of other primitives that can be achieved in the presence of After-the-Fact Leakage, for example commitment, key-agreement, etc.

Acknowledgement We would like to thank Daniel Wichs for many delightful discussions and the anonymous referees of TCC 2011 for their helpful comments and suggestions.

References

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC'09*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.

[2] B. Barak, A. Rao, R. Shaltiel, and A. Wigderson. 2-source dispersers for sub-polynomial entropy and ramsey graphs beating the frankl-wilson construction. In *STOC*, pages 671–680, 2006.

[3] J. Bourgain. More on the sum-product phenomenon in prime fields and its applications. *International Journal of Number Theory*, 1:1–32, 2005.

[4] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Cryptography resilient to continual memory leakage. Cryptology ePrint Archive, Report 2010/278, 2010. `http://eprint.iacr.org/`.

[5] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity (extended abstract). In *FOCS'85*, pages 429–442. IEEE, 1985.

[6] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. Preliminary version in CRYPTO'98.

[7] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[8] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, May 1991. ACM.

[9] S. Dziembowski and K. Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.

[10] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS'08*, pages 293–302. IEEE Computer Society, 2008.

[11] S. Goldwasser and G. N. Rothblum. Securing computation against continuous leakage. In *Advances in Cryptology - CRYPTO'10*, volume 6223 of *Lecture Notes in Computer Science*, pages 59–79. Springer, 2010.

[12] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: cold-boot attacks on encryption keys. *Commun. ACM*, 52(5):91–98, 2009.

[13] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[14] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[15] A. Juma and Y. Vahlis. Protecting cryptographic keys against continual leakage. In *Advances in Cryptology - CRYPTO'10*, volume 6223 of *Lecture Notes in Computer Science*, pages 41–58. Springer, 2010.

[16] S. Micali and L. Reyzin. Physically observable cryptography. In *TCC'04*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[17] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology - CRYPTO'09*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.

[18] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[19] N. Nisan and D. Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[20] A. Rao. Extractors for a constant number of polynomially small min-entropy independent sources. In *STOC*, pages 497–506, 2006.

[21] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986.

[22] U. V. Vazirani. Strong communication complexity or generating quasirandom sequences form two communicating semi-random sources. *Combinatorica*, 7(4):375–392, 1987.