An extended abstract of this paper is published in the proceedings of the 16th IACR International Conference on Practice and Theory of Public-Key Cryptography—PKC 2013. This is the full version.

On the Connection between Leakage Tolerance and Adaptive Security*

Jesper Buus Nielsen[1], Daniele Venturi[†2], and Angela Zottarel[1]

[1]*Aarhus University*
[2]*Sapienza University of Rome*

March 12, 2015

Abstract

We revisit the context of leakage-tolerant interactive protocols as defined by Bitanski, Canetti and Halevi (TCC 2012). Our contributions can be summarized as follows:

- For the purpose of secure message transmission, any encryption protocol with message space $\mathcal{M}$ and secret key space $\mathcal{SK}$ tolerating poly-logarithmic leakage on the secret state of the receiver must satisfy $|\mathcal{SK}| \geq (1-\epsilon)|\mathcal{M}|$, for every $0 < \epsilon \leq 1$, and if $|\mathcal{SK}| = |\mathcal{M}|$, then the scheme must use a fresh key pair to encrypt each message.
- More generally, we show that any n party protocol tolerates leakage of $\approx poly(\log \kappa)$ bits from one party at the end of the protocol execution, *if and only if* the protocol has passive adaptive security against an adaptive corruption of one party at the end of the protocol execution. This shows that as soon as a little leakage is tolerated, one needs full adaptive security.

All our results can be based on the only assumption that collision-resistant function ensembles exist.

Keywords. simulation-based security, leakage tolerance, adaptive security, arguments of knowledge

*Partially supported by European Research Council Starting Grant 279447. Partially supported by DFF Starting Grant 10-081612. Partially supported by the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, and also from the CFEM research center (supported by the Danish Strategic Research Council).

†Work done while at Aarhus University.

Contents

1 Introduction

Would you trust your partner when you don't trust his secrets? Suppose that Alice has a confidential message m she wants to communicate to Bob, in a way that the content of m is protected from outsiders. In a world where public-key cryptography exists, Bob can sample a fresh public key pk and hands it to Alice via an authenticated channel (while keeping the corresponding secret key sk). Now Alice can use pk to encrypt m and send the resulting ciphertext c to Bob, who in turn can decrypt using sk and recover the message.

The problem sketched above, also known as the problem of *secure message transmission*, is one of the most basic questions in cryptography. For instance we know that when Bob's secret key is uniform and "well protected", any semantically secure encryption scheme would suffice for the purpose of secure message transmission. But what if (part of) Bob's secrets can leak to an outsider? Even when Bob's secret is not exposed, what if the randomness Alice used to encrypt can leak? Can Alice still trust the protocol above?

Leakage-resilient cryptography. In the last few years, questions of this kind gained momentum in the cryptographic community due to the spread of *side-channel attacks*. Starting from the early 90s', it has become clear that an adversary can potentially gain partial information on the secret state of uncorrupted players in a variety of ways, e.g. by measuring time [35], power [36] and electromagnetic emission [45]. This information, often called *leakage*, can be powerful knowledge in the hands of an adversary, putting security of the cryptographic primitive under attack on edge.

Indeed, cryptographic algorithms are typically analyzed in a black-box fashion where secrets are assumed to be completely oblivious to an adversary; in particular they offer no guarantees in the presence of side-channel attacks. To change the above state of affairs, researchers started to investigate the possibility of constructing schemes which preserve both their functionality and their security properties even in the presence of (an as large as possible class of) leakage. As a result, we now possess a rich list of leakage-resilient (a.k.a. leakage-tolerant) schemes, e.g., for pseudorandomness generation [21, 43], storage [15, 18], encryption [39, 16, 24], signatures [33, 23, 24, 11, 41, 14, 22] and general non-interactive circuits [31, 25, 20].

However, in order to have a scheme Π which maintains (in the presence of leakage) exactly the same security guarantees it has in a leak-free setting, some restriction on the leakage itself must be placed as to escape trivial attacks. Examples include putting a bound on the total information leaked, assuming that "only computation leaks information" [38], that different parts of the memory leak independently [15, 18, 19], that leakage occurs only in specific times or that the leakage is "hard to invert" [17]. Two general approaches have emerged:

- In the *game-based* approach, one augments a standard cryptographic game for Π by giving the adversary $\mathbb{A}$ access to an auxiliary interface from which she can input some function (within a set of admissible functions) and receive back the value of the function applied to the secret state of Π.

- In the *simulation-based* approach, one shows that Π (augmented with a leakage interface) achieves the same properties of an ideal execution where a simulator $\mathbb{S}$ interacts with a functionality $\mathbf{F}$ (also augmented with a leakage interface) and no communication between parties takes place. Hence, security is achieved if $\mathbb{A}$ can be simulated in the UC framework [12], i.e. for any $\mathbb{A}$ attacking Π there exists a simulator $\mathbb{S}$ such that no environment $\mathbb{Z}$ can tell whether it is interacting with $\mathbb{A}$ and Π or with $\mathbb{S}$ and $\mathbf{F}$.

Both approaches have advantages and disadvantages. Sometimes, game-based notions do not exactly capture the realistic security threats they wish to model and do not come in general with easy composition rules. Simulation-based notions are harder to achieve and often require the use of expensive tools.

The model of Bitanski, Canetti and Halevi. In this paper we focus on the second approach, building upon previous work of Bitanski *et al.* [6]. In their model, leakage queries from an adversary $\mathbb{A}$ are viewed as a form of partial corruption, where $\mathbb{A}$ does not receive the complete state of the chosen party but just some function $f_{\mathbb{A}}(\cdot)$ of it.

Note that without any help the simulator $\mathbb{S}$ would have a very hard life. Consider for instance the case of secure message transmission: Already a single bit of arbitrary leakage, say the first bit of the transmitted message, makes it impossible to achieve semantic security! The solution is to allow also the simulator to leak on the "ideal state" of the protocol, by specifying some function $f_{\mathbb{S}}(\cdot)$. Now, security means that a real world attacker leaking λ bits from the entire secret state of the implementation can be simulated given λ bits of leakage on the corresponding ideal state (i.e., on the message alone in case of secure message transmission).

The functionality is also able to react to leakage, in the sense that it can be asked to "give-up" on security when too much leakage occurred. This feature allows us to model relaxed security notions of protocols in the presence of leakage, and in particular to specify how the security degrades with the leakage.

1.1 Our Contribution

We revisit the context of leakage-tolerant interactive protocols. Our results give strong evidence that leakage tolerance in the simulation-based setting requires expensive tools already when a small amount of leakage needs to be tolerated. Our main contributions are outlined below:

1. For the concrete case of secure message transmission, we show that any encryption protocol Π tolerating a poly-logarithmic amount of leakage in the definition of Bitanski *et al.* [6] must satisfy $|\mathcal{SK}| \geq (1-\epsilon)|\mathcal{M}|$ for all $0 < \epsilon \leq 1$, where $\mathcal{M}$ is the message space and $\mathcal{SK}$ is the space of secret keys. In other words, the decryption key must be essentially as long as the message being encrypted. Furthermore, if the messages and the secret keys have the same length, then a fresh key must be used to encrypt every message.

2. We prove that Π is secure against one adaptive corruption of the receiver at the end of the protocol execution *if and only if* Π is secure against leakage of $\approx$ *poly*$(\log\kappa)$ bits from the receiver's internal state at the end of the protocol execution. More in general, we prove that any n-party protocol tolerates leakage of $\approx$ *poly*$(\log\kappa)$ bits from one party at the end of the protocol execution, *if and only if* the

protocol has passive security against an adaptive corruption. This shows that simulation-based leakage tolerance becomes identical to full adaptive security already for very little leakage, as long as at most one party can be corrupted.

3. We further discuss under which conditions it is possible to generalize our result from 2 to adaptive corruption of an *arbitrary* number of parties in a leakage-tolerant protocol.

All our results can be based on the solely assumption that collision-resistant function ensembles exist.

1.2 Our Techniques

At the heart of our results there is a novel technique exploiting succinct interactive arguments for $\mathbf{NP}$. These are argument systems where the total amount of communication is at most poly-logarithmic in the length of the witness and the instance being proven. Succinct interactive arguments (with a constant number of rounds) are known to exist given any collision-resistant function ensemble [34, 47].

Proof outline. We now sketch the proof of our main result. Since protocol Π is leakage-tolerant, there exists a simulator $\mathbb{S}$ producing a "convincing" view of the protocol for $\mathbb{A}$. In addition, $\mathbb{S}$ can handle leakage queries from $\mathbb{A}$.

We exhibit an environment $\mathbb{Z}$ for which the existence of a simulator yields our bound. The environment inputs a uniformly random $m \in \mathcal{M}$. Then, it lets the protocol terminate without making any leakage query or any corruption, i.e. it simply delivers all messages between Alice (the sender) and Bob (the receiver). As part of this, $\mathbb{Z}$ learns pk and the ciphertext c from observing the communication on the authenticated channel. After the protocol terminates, $\mathbb{Z}$ asks the receiver to prove the following $\mathbf{NP}$-statement via a succinct argument system: "There exists some sk that explains c as an encryption of m". Notice that the receiver can do this as it knows the secret key (i.e., a valid witness).

The crux of the strategy above is that $\mathbb{Z}$ can play the role of the verifier in the interactive argument by using the leakage queries on the state of the receiver to "extract" the massages of the prover.

Now, by completeness of the argument system, in the real world the proof will be accepting with overwhelming probability. On the other hand, leakage tolerance of Π implies that the simulator must cook-up an indistinguishable output in the ideal world. However, $\mathbb{S}$ has to choose c beforehand to simulate $\mathbb{A}$'s view, and later answer leakage queries consistently by "explaining" c as an encryption of m for decryption key sk. It follows from (computational) soundness of the proof system that this is only possible if for a large fraction of the messages in $\mathcal{M}$ there exists a secret key sk' which explains c consistently. From this, a simple counting argument shows that $|\mathcal{M}|$ must be negligibly close to $|\mathcal{SK}|$.

Extracting the state. Let $H(\cdot)$ be a collision-resistant hash function with range μ bits. When the argument system from above is an *argument of knowledge* (i.e., there exists a knowledge extractor which is able to extract a valid witness for a statement when given access to a successful prover with respect to that statement), we are able to show that Π is leakage-tolerant against $2poly(\log \kappa) + \mu + 1$ bits of leakage from the receiver's internal state, if and only if Π has semi-honest adaptive security. The second direction follows directly from the result of [6] that adaptive (semi-honest) security is sufficient to obtain leakage tolerance for a broad class of functionalities.

To prove the first direction, one has to construct a simulator $\mathbb{S}'$ which simulates first the communication (pk, c) of the protocol to adversary $\mathbb{A}'$, and then after being given m simulates the internal state of the receiver consistently. Roughly, we do this as follows. We start by considering an adversary $\mathbb{A}$ against leakage tolerance of Π; from the definition of leakage tolerance, we know there exists a simulator $\mathbb{S}$. Hence, we use $\mathbb{S}$ to construct $\mathbb{S}'$. The adversary starts by leaking the value h obtained by applying $H(\cdot)$ on the final state of the receiver; then $\mathbb{A}$ uses an argument of knowledge to ask for a proof that there is a consistent state inside the receiver which could be extracted (consistent also with the above value of h). Now, $\mathbb{A}$ uses an additional leakage query to "send" a distinguisher $\mathbb{Z}'$ (attacking adaptive security of Π) inside the receiver, have a look at the state and output its guess b. Finally, the adversary leaks a proof that the bit b was actually computed by $\mathbb{Z}'$ from the *same* state which could be extracted from the first argument of knowledge. Note that the latter can be achieved by using the same value of h in both arguments.

It follows that if we later use a simulator $\mathbb{S}'$ for this attack and extract from its first argument of knowledge some state, this state will have to look indistinguishable from a real state to any $\mathbb{Z}'$ (as long as finding collisions in $H(\cdot)$ is hard). Adaptive security follows.

Corruption of more parties. The above proof technique is quite general, and in fact it can be applied to any leakage-tolerant interactive n-party protocol, where at most one party gets corrupted.

The n-party case with arbitrary corruptions is more subtle as now a distinguisher for the adaptive security game should have access to the state of all parties when it makes its guess, and it is not clear how to simulate this given short leakages from each state. In particular, we cannot "send" a distinguisher $\mathbb{Z}'$ into each of the parties one by one, as sending $\mathbb{Z}'$ out of the parties again could require too much leakage. Indeed, in this case we do not know how to force the extracted internal states from the parties to be indistinguishable from the internal state in the real world.

We conjecture that for corruption of multiple parties equivalence does not hold in general in case of independent leakages from the state of each party; we leave the construction of a separating example as an open problem.[1]

1.3 Related Work

Simulation-based notions of leakage tolerance have been considered also for public-key encryption schemes by Halevi and Lin [29], and in the context of zero-knowledge protocols [27, 42, 1], coin tossing [10], and secure multi-party computation [9, 8, 7].

We mention a few other papers exploiting argument systems for negative results. The first one is the work on "seed-incompressible functions" of Halevi, Myers and Rackoff [30], who use CS proofs [37] to show that no pseudorandom function exists which remains secure after one leaks a "compressed" key. Another example is the work of [44] on parallel repetition of computationally sound proofs, and the work of Jain and Pietrzak [32] who show that (game-based) leakage resilience for natural primitives like signatures and encryption does not always amplify in case of parallel repetition. The first and the last results rely on random oracles, whereas the second one is based on universal arguments [2].

We stress that the techniques used in all the above works are substantially different than ours.

2 Preliminaries

2.1 Notation

We let $\mathbf{N}$ denote the naturals and $\mathbf{R}$ denote the reals. For $a, b \in \mathbf{R}$, we let $[a, b] = \{x \in \mathbf{R} \,;\ a \leq x \leq b\}$; for $a \in \mathbf{N}$ we let $[a] = \{1, 2, \ldots, a\}$. If x is a string, we denote its length by $|x|$; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents

[1] In case the simulator is allowed joint leakage on the state of the parties equivalence still holds.

the number of elements in $\mathcal{X}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow \mathcal{X}$. When $\mathbb{A}$ is an algorithm, we write $y \leftarrow \mathbb{A}(x)$ to denote a run of $\mathbb{A}$ on input x and output y; if $\mathbb{A}$ is randomized, then y is a random variable and $\mathbb{A}(x; r)$ denotes a run of $\mathbb{A}$ on input x and randomness r. An algorithm $\mathbb{A}$ is *probabilistic polynomial-time* (PPT) if $\mathbb{A}$ is allowed to use randomness as part of its logic (i.e., $\mathbb{A}$ is probabilistic) and for any input $x \in \{0,1\}^*$ the computation of $\mathbb{A}(x)$ terminates in at most $poly(|x|)$ steps.

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

For two ensembles $\mathcal{X} = \{X_\kappa\}_{\kappa \in \mathbf{N}}, \mathcal{Y} = \{Y_\kappa\}_{\kappa \in \mathbf{N}}$, we write $\mathcal{X} \approx \mathcal{Y}$, meaning that every probabilistic polynomial-time distinguisher has negligible advantage in distinguishing $\mathcal{X}$ and $\mathcal{Y}$.

2.2 Interactive Argument Systems

Our results are based on the existence of round-efficient interactive argument systems. The definition below is taken from [47].

Definition 1 (Round-efficient interactive argument system)**.** *An interactive protocol (P, V) is an interactive argument system for a language $\mathcal{L}$ if there is a relation $\mathcal{R}$ such that $\mathcal{L} = \mathcal{L}_{\mathcal{R}}$, and functions $\nu, s : \mathbf{N} \rightarrow [0, 1]$ such that $1 - \nu(\kappa) > s(\kappa) + 1/poly(\kappa)$ 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 $1 - \nu(|x|)$.*
- *(Computational soundness): If $x \notin \mathcal{L}$, then for every non-uniform probabilistic polynomial-time P^* and for all sufficiently long $x \notin \mathcal{L}$, the verifier V accepts in $(P^*, V)(x)$ with probability at most $s(|x|)$.*

The value $\nu(\cdot)$ is called the *completeness error* and the value $s(\cdot)$ is called the *soundness 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. The protocol is called *public-coin* when the verifier's moves consist merely of tossing coins and sending their outcomes to the prover. We write $\mathbf{AM}_{\nu,s}(\rho(\kappa), \lambda(\kappa))$ to denote public-coin interactive argument systems with completeness error ν, soundness 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.

We get an argument of knowledge whenever it is possible to extract a witness from any successful prover:

Definition 2 (Argument of knowledge)**.** *An interactive protocol (P, V) is an interactive argument of knowledge for a language $\mathcal{L}$ if it is an interactive argument system, where the computational soundness condition is replaced by the following:*

- *(Argument of knowledge): For all non-uniform probabilistic polynomial-time P^* such that V accepts in $(P^*, V)(x)$* with overwhelming probability*, there exists a non-uniform probabilistic polynomial-time extractor E_{P^*} outputting (x, w) such that $(x, w) \in \mathcal{R}$ with overwhelming probability.*

There are other forms of extractability, where from any prover succeeding to convince V with probability $p(\cdot)$, one can extract a witness with probability which is polynomially related to $p(\cdot)$ [3, 46]. Here we only need the weak notion above, where extraction is only guaranteed if the prover convinces the verifier with probability close to 1. The technical reason is that in the real world we will ask a party to "leak" an argument of knowledge of its internal state, which will succeed with overwhelming probability by completeness.

Instantiations. Kilian [34] constructs a 4-round public-coin succinct argument of knowledge for $\mathbf{NP}$ based on a probabilistically checkable proof (PCP) system for $\mathbf{NP}$ and a collision-resistant function ensemble. Gentry and Wichs [28] 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 [4] (as shows Micali [37] by applying the Fiat-Shamir transform [26] to Kilian's protocol) or under so-called "knowledge of exponent" assumptions [5].

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.

2.3 Leakage-Tolerant Secure Message Transmission

Syntax of public-key encryption. A public-key encryption (PKE) scheme is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows. The key generation algorithm Gen takes as input a security parameter κ and outputs $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$; we let $\mathcal{PK} \times \mathcal{SK}$ be the key space. The encryption algorithm takes as input a message $m \in \mathcal{M}$ and outputs a ciphertext $c \leftarrow \mathsf{Enc}(pk, m)$ in some ciphertext space $\mathcal{C}$. The decryption algorithm takes as input a ciphertext $c \in \mathcal{C}$ and a secret key $sk \in \mathcal{SK}$ and outputs $m \leftarrow \mathsf{Dec}(sk, c)$.

Since we aim to apply our result to arbitrary encryption schemes, we will assume that decryption is also randomized. We say that $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has negligible completeness error if it holds that $\mathsf{Dec}(sk, (\mathsf{Enc}(pk, m))$ returns m with overwhelming probability over the coin tosses of $(\mathsf{Enc}, \mathsf{Dec})$ and the choices of $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$ and $m \in \mathcal{M}$.

Leakage-tolerant PKE. We recall the simulation-based notion of leakage tolerance introduce by Bitansky *et al.* [6]. Informally, leakage queries from an adversary $\mathbb{A}$ are viewed as a form of partial corruptions, where $\mathbb{A}$ does not received the complete state of the chosen party but just some function of it. Security is then achieved if such an adversary can be simulated in the UC framework. Without loss of generality we will consider only dummy adversaries — adversaries which just carry out the commands of the environment. I.e., it is the environment which specifies all leakage queries. We will therefore completely drop the adversary in the notation for clarity.

Let Π be a protocol implementing an ideal functionality $\mathbf{F}$. Let $\mathbb{Z}$ be an environment trying to "break" security of Π. The environment specifies all inputs to the protocol, sees all messages sent, schedules all message deliveries, sees all outputs and is in addition allowed to make leakage queries during the run of the protocol. Such queries are modelled in the following way: When $\mathbb{Z}$ wants to leak from the state of player X, it sends a leakage request $(X, f_{\mathbb{Z}})$ upon which it receives $f_{\mathbb{Z}}(\sigma_X)$, where σ_X is the current secret state of X. The function $f_{\mathbb{Z}}$ can be any function within a set of admissible leakage functions $\mathcal{F}$, which is a parameter in the definition.

In the ideal world, a trusted party is running $\mathbf{F}$ and a simulator $\mathbb{S}$ is interacting with it. The simulator must then simulate the protocol to the environment $\mathbb{Z}$. All inputs specified by $\mathbb{Z}$ go directly to $\mathbf{F}$; the simulator only sees the input of corrupted parties. The simulator must then simulate the communication of

Functionality $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$

Running with parties R, S and adversary $\mathbb{S}$, the functionality $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ is parametrized by the security parameter κ, message space $\mathcal{M}$ and the set of all admissible leakage functions $\mathcal{F}$. Hence, $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ behaves as follows:

- Upon input $(\texttt{send}, S, R, m)$ send a message $(\texttt{send}, S, R, |m|)$ to $\mathbb{S}$. Once $\mathbb{S}$ allows to forward the message, send $(\texttt{sent}, S, m)$ to R.
- Upon input $(\texttt{leak}, X, f_{\mathbb{Z}})$ for $X \in \{S, R\}$ and $f_{\mathbb{Z}} \in \mathcal{F}$ send a message $(\texttt{leak}, X)$ to $\mathbb{S}$. Receive $(\texttt{leak}, X', f_{\mathbb{S}})$ from $\mathbb{S}$, check that $f_{\mathbb{S}} \in \mathcal{F}$, and that $|f_{\mathbb{Z}}(\cdot)| = |f_{\mathbb{S}}(\cdot)|$ and $X' = X$. Send $(\texttt{leak}, f_{\mathbb{S}}(m))$ to $\mathbb{S}$ and $(\texttt{leaked}, |f_{\mathbb{S}}(m)|)$ to X'.

Figure 1: Ideal functionality $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ for secure message transmission with leakage

the protocol to $\mathbb{Z}$. In addition, all leakage queries $(\texttt{leak}, X, f_{\mathbb{Z}})$ from $\mathbb{Z}$ goes to the simulator. When a query $(\texttt{leak}, X, f_{\mathbb{Z}})$ arrives, the simulator is allowed to make its own leakage query $(\texttt{leak}, X, f_{\mathbb{S}})$ to the ideal functionality, under the restriction that the length of the leakage requested by $\mathbb{S}$ does not exceed the length of the leakage requested by $\mathbb{Z}$.

We say that Π is a leakage-tolerant secure implementation of $\mathbf{F}$ if there exists a simulator $\mathbb{S}$ such that no environment can distinguish between the real life protocol Π and $\mathbb{S}$ interacting with the ideal functionality $\mathbf{F}$. More formally, consider the ideal functionality $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$, depicted in Figure 1. Denote with $\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}, \mathbb{S}, \mathbb{Z}}(\mathcal{F}, \kappa)$ the output of the environment $\mathbb{Z}$ when interacting with simulator $\mathbb{S}$ in the simulation.

Consider the following protocol Π between a sender S and a receiver R, supposed to realize $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ via a public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$ and key space $\mathcal{PK} \times \mathcal{SK}$, assuming authenticated channels:

1. S transmits to R its willing to forward a message $m \in \mathcal{M}$;
2. R samples $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$, where $pk \in \mathcal{PK}$ and $sk \in \mathcal{SK}$, and sends pk to S;
3. S computes $c = \mathsf{Enc}(pk, m; r_E)$ and forwards the result to R;
4. R outputs $m' = \mathsf{Dec}(sk, c; r_D)$.

Note that at the end of the execution of Π the state of S is $\sigma_S = (m, r_E)$ whereas the state of R is $\sigma_R = (sk, r_G, r_D, m')$. Denote with $\mathbf{REAL}_{\Pi, \mathbb{Z}}(\mathcal{F}, \kappa)$ the output of the environment $\mathbb{Z}$ after interacting with parties R, S in a real execution of Π.

Definition 3 (Leakage-tolerant PKE protocol)**.** *We say that Π is a leakage-tolerant public-key encryption protocol (w.r.t. a set of leakage functions $\mathcal{F}$) if Π securely implements $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$, i.e., there exists a probabilistic polynomial-time simulator $\mathbb{S}$ such that for any environment $\mathbb{Z}$ it holds that*

$$\{\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}, \mathbb{S}, \mathbb{Z}}(\mathcal{F}, \kappa)\}_{\kappa \in \mathbf{N}} \approx \{\mathbf{REAL}_{\Pi, \mathbb{Z}}(\mathcal{F}, \kappa)\}_{\kappa \in \mathbf{N}}.$$

When the total amount of leaked information is $\lambda = \sum_i |f_{\mathbb{Z}}^{(i)}(\cdot)|$, we say that Π tolerates λ bits of leakage.

3 Leakage-Tolerant PKE

In this section we present a result regarding the complexity of encryption schemes that are leakage-resilient according to Definition 3. Looking ahead, we will prove that it is not possible to achieve security in this setting without relying on an encryption scheme having similar properties to non-committing encryption [13].

3.1 Necessity of Long Keys

We show the following theorem.

Theorem 1 (Definition 3 requires long keys)**.** *Assume the existence of $\mathbf{AM}_{negl(\kappa),negl(\kappa)}(O(1), \lambda(\kappa))$ argument systems for $\mathbf{NP}$, where $\lambda(\kappa) = \lambda_P(\kappa) + \lambda_V(\kappa)$. Let Π be a leakage-tolerant public-key encryption protocol with key space $\mathcal{PK} \times \mathcal{SK}$ and message space $\mathcal{M}$. Then, whenever Π tolerates $\lambda'(\kappa) = \lambda_P(\kappa)$ bits of leakage it must be that $|\mathcal{SK}| \geq (1-\epsilon)|\mathcal{M}|$ for all $1 \geq \epsilon > 0$. In particular, if $\ell(\mathcal{SK})$ and $\ell(\mathcal{M})$ are resp. the bit length of the secret key and of the messages, we have $\ell(\mathcal{SK}) \geq \ell(\mathcal{M}) - 1$, i.e. to encrypt a message of length ℓ bits one needs a key of length at least $\ell - 1$ bits.*

Proof. Assume first that the decryption algorithm is deterministic and that the encryption scheme has perfect correctness, i.e., $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m; r_E)) = m$ for all r_E and for all m when $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$.

Since protocol Π is leakage-tolerant, we know that there exist a simulator $\mathbb{S}$ producing a "convincing" view of the protocol. Moreover, $\mathbb{S}$ can handle requests of the kind $(\texttt{leak}, X, f_{\mathbb{Z}})$, where X is either S or R and $f_{\mathbb{Z}}$ is a leakage function (chosen by the environment) to be applied to the internal state σ_X of X.

We construct an environment $\mathbb{Z}$ which uses λ_P bits of leakage on the receiver's state after the execution of Π, for which the existence of simulator $\mathbb{S}$ implies our bound. Consider the following relation:

$$\mathcal{R} := \{((pk, c, m), (sk, r_G)) : \ (pk, sk) = \mathsf{Gen}(1^\kappa; r_G) \wedge \mathsf{Dec}(sk, c) = m\} \ , \tag{1}$$

and let (P, V) be an $\mathbf{AM}_{negl(\kappa),negl(\kappa)}(O(1), \lambda(\kappa))$ argument system for $\mathcal{L} = \mathcal{L}(\mathcal{R})$. The main idea will be to let $\mathbb{Z}$ play the role of the verifier in the argument system, while running the prover with the help of the leakage queries on the state of the receiver. The environment $\mathbb{Z}$ works as follows:

1. Input a uniformly random $m \in \mathcal{M}$ to S.
2. Let the protocol terminate without any leakage queries or any corruptions, i.e., simply deliver all messages between S and R. As part of this $\mathbb{Z}$ learns pk and c from observing the authenticated channel between S and R.
3. After the protocol terminates, let R prove via leakage queries that $x = (pk, c, m) \in \mathcal{L}$. Notice that R can do this as it knows the witness $w = (sk, r_G)$. Details follow.

We now show how to generate an interactive argument for $\mathcal{L}$, by letting $\mathbb{Z}$ (holding the instance $x = (pk, c, m)$) play the role of the verifier and using the leakage queries on the receiver's state $w = (sk, r_G)$ to generate the interaction with the prover. Wlog. assume the verifier talks first, and denote with $\rho(\kappa) = poly(\kappa)$ the total number of rounds. (The case where the prover talks first can be derived similarly.)

We introduce some auxiliary notation. Let r_P (r_V) be a random string long enough to specify all random choices done by the prover (verifier), such that for fixed r_P (r_V), the prover (verifier) is deterministic. For all $i = 0, \dots, \rho/2 - 1$, denote with $y_{2i+1} = V(x, 2i+1, view_{2i}; r_V)$ the next message sent by the verifier, where the variable $view_j$ is defined as the entire view until round $j \in [\rho]$. Similarly, the next message computed by the prover is computed as $y_{2i} = P(x, w, 2i, view_{2i-1}; r_P)$ for all $i = 1, \dots, \rho/2$. Note that, with this notation, the complete view consists of $(y_1, y_2, \dots, y_\rho)$. At the end the verifier computes a judgement $J(x, view_\rho; r_V) \in \{0, 1\}$, where 1 indicates accept.

Therefore, it suffices to specify how $\mathbb{Z}$ (holding only (pk, c, m)) can generate the messages of the prover. It proceeds as follows:

1. $\mathbb{Z}$ samples uniformly random r_P and r_V.
2. $\mathbb{Z}$ computes $y_1 = V(x, 1, \perp; r_V)$ and then sets the leakage function $f_{\mathbb{Z}}^{(1)}$ to be the function $f_{\mathbb{Z}}^{(1)}(w) = P(x, w, 2, y_1; r_P)$. (This can be done by "hard-wiring" the values x and y_1 into the leakage function.)
3. In general, given $view_{2i} = (y_1, y_2, \dots, y_{2i})$, the adversary $\mathbb{Z}$ can compute y_{2i+1}, hard-wire this value into $f_{\mathbb{Z}}^{(i)}$ and get $y_{2i+2} = P(x, w, 2i+2, y_{2i+1}; r_P)$. This can be done for all $i \in [\rho]$, until the last message y_ρ of the argument system is obtained.
4. Then $\mathbb{Z}$ outputs $J(x, view_\rho; r_V)$ as its guess.

Note that the total amount of leaked information is the communication complexity of the prover in (P, V), i.e., λ_P bits. By completeness of the argument system, we know that $\mathbf{REAL}_{\Pi,\mathbb{Z}}(\mathcal{F}, \kappa) = 1$, except with negligible probability. From this we conclude that $\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}},\mathbb{S},\mathbb{Z}}(\mathcal{F}, \kappa) = 1$ except with negligible probability, by security of the protocol. We write out what this means. The simulation proceeds as follows:

1. First $\mathbb{Z}$ inputs a uniformly random $m \in \mathcal{M}$ to the ideal functionality on behalf of S. As a result $\mathbb{S}$ is given $(\texttt{send}, S, R, |m|)$.
2. Then $\mathbb{S}$ must simulate the communication of the protocol, which in particular means that it must output some pk and c to $\mathbb{Z}$.
3. After the simulation of the protocol terminates, the environment makes the leakage queries with which R proves that $x = (pk, c, m) \in \mathcal{L}$. The leakage queries are answered by $\mathbb{S}$. In more detail:
   (a) $\mathbb{Z}$ samples uniformly random r_P and r_V.
   (b) $\mathbb{Z}$ sets the leakage function $f_{\mathbb{Z}}^{(1)}$ to be the function $f_{\mathbb{Z}}^{(1)}(w) = P(x, w, 2, y_1; r_P)$. The function is sent to $\mathbb{S}$, who must choose some function $f_{\mathbb{S}}$ producing value y_2.
   (c) In general, given $view_{2i} = (y_1, y_2, \dots, y_{2i})$, the environment $\mathbb{Z}$ specify $f_{\mathbb{Z}}^{(i)}$ and sends the same $f_{\mathbb{Z}}^{(i)}$ as in the protocol to $\mathbb{S}$ which in turn chooses $f_{\mathbb{S}}^{(i)}$ defining some y_{2i+1}. This is done for all $i \in [\rho]$, until the last message y_ρ of the argument system is obtained.
   (d) Then $\mathbb{Z}$ outputs $J(x, view_\rho; r_V)$ as its guess.

Since $\mathbb{Z}$ is computing its own messages y_{2i+1} as the verifier of (P, V) would have done, and the messages y_{2i} are computed by $\mathbb{S}$ which is PPT, and $J(x, view_\rho; r_V) = 1$, it follows from soundness that $x \in \mathcal{L}$ except with negligible probability. This means that there exist (sk, r_G) such that $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$ and $m = \mathsf{Dec}(sk, c)$. In particular, there exist $sk \in \mathcal{SK}$ such that $m = \mathsf{Dec}(sk, c)$. Let $M_{pk,c} \subset \mathcal{M}$ denote the subset of $m' \in \mathcal{M}$ for which there exist $sk' \in \mathcal{SK}$ such that $m' = \mathsf{Dec}(sk', c)$. We have that $m \in M_{pk,c}$. Notice, that if it was the case that $m \notin M_{pk,c}$, then it would be the case that $(pk, c, m) \notin \mathcal{L}$ and hence $\mathbb{S}$ would not be able to answer the leakage queries such that $J(x, view_\rho; r_V) = 1$, except with negligible probability, by soundness. Hence, it follows from $\{\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}},\mathbb{S},\mathbb{Z}}(\mathcal{F},\kappa)\}_{\kappa\in\mathbf{N}} \approx \{\mathbf{REAL}_{\Pi,\mathbb{Z}}(\mathcal{F},\kappa)\}_{\kappa\in\mathbf{N}}$ that the probability that $m \in M_{pk,c}$ is overwhelming. This implies that $|M_{pk,c}|/|\mathcal{M}|$ is negligibly close to 1, in particular $|M_{pk,c}| \geq (1-\epsilon)|\mathcal{M}|$ for all $0 < \epsilon \leq 1$. Take two $m_0 \neq m_1 \in M_{pk,c}$. By definition there exist $sk_0, sk_1 \in \mathcal{SK}$ such that $m_0 = \mathsf{Dec}(sk_0, c)$ and $m_1 = \mathsf{Dec}(sk_1, c)$. From $m_0 \neq m_1$, we conclude that $sk_0 \neq sk_1$, so $|\mathcal{SK}| \geq |M_{pk,c}|$. From this we get the theorem.

To handle randomized decryption functions, we let the environment pick the randomness which should be used for decryption. I.e., $\mathbb{Z}$ hard-wires a random string r_D into the instance x and asks the receiver to prove that there exists r_G, sk such that $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$ and $\mathsf{Dec}(sk, c; r_D) = m$. In the real world, this will hold with overwhelming probability, and hence in the ideal world we can, along the lines above, conclude that for any two messages m_0 and m_1, there exists $sk_0, sk_1 \in \mathcal{SK}$ such that $m_0 = \mathsf{Dec}(sk_0, c; r_D)$ and $m_1 = \mathsf{Dec}(sk_1, c; r_D)$. This again allows to conclude that $sk_0 \neq sk_1$. Note that it is important that $\mathbb{Z}$ picks r_D. If it was considered part of the witness, we would only get that there exists $sk_0, sk_1 \in \mathcal{SK}$ and r_D^0, r_D^1 such that $m_0 = \mathsf{Dec}(sk_0, c; r_D^0)$ and $m_1 = \mathsf{Dec}(sk_1, c; r_D^1)$, from which we cannot conclude that $sk_0 \neq sk_1$, as $r_D^0 \neq r_D^1$ might be enough to give different decryptions for a fixed $sk_0 = sk_1$. □

Remark 1. *Assuming the existence of collision-resistant function ensembles (which implies an argument system for $\mathbf{AM}_{negl(\kappa),negl(\kappa)}(4, poly(\log\kappa))$), we get that Theorem 1 holds for any leakage-tolerant public-key encryption protocol tolerating poly-logarithmic leakage on the receiver's state.*

3.2 On Re-Using Keys

One could still hope that it is possible to use the same key to encrypt more than one message. Below, we prove that this hope is also vacuous.

Corollary 1 (Fresh key for every message)**.** *If Π is a leakage-tolerant public-key encryption protocol tolerating poly-logarithmic leakage and such that $2\ell(\mathcal{M}) - 1 > \ell(\mathcal{SK}) \geq \ell(\mathcal{M}) - 1$, then a fresh key must be used to encrypt every message.*

Proof. We prove this by contradiction to Theorem 1. Namely, assume $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has message space $\mathcal{M}$, key space $\mathcal{PK} \times \mathcal{SK}$ and uses a single pair $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$ to encrypt two messages m' and m'' sequentially. Denote with $c' \leftarrow \mathsf{Enc}(pk, m')$ and $c'' \leftarrow \mathsf{Enc}(pk, m'')$ the corresponding ciphertexts.

Now consider the following public-key encryption scheme $\overline{\Pi} = (\overline{\mathsf{Gen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$. The key generation algorithm $\overline{\mathsf{Gen}}$ simply runs $(pk, sk) \leftarrow \mathsf{Gen}(1^\kappa)$. The encryption algorithm takes as input a message $m \in \mathcal{M}^2$, writes it as $m = m'||m''$ and outputs

$$\overline{\mathsf{Enc}}(pk, m) = \mathsf{Enc}(pk, m')||\mathsf{Enc}(pk, m'') = c'||c'' = c.$$

The decryption algorithm $\overline{\mathsf{Dec}}$ parses c as $c'||c''$ and outputs $m \leftarrow \mathsf{Dec}(sk, c')||\mathsf{Dec}(sk, c'')$.

Since Π securely realizes $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ in the presence of λ bits of leakage, Theorem 1 implies $\ell(\mathcal{SK}) \geq \ell(\mathcal{M}) - 1$. On the other hand, the notion of leakage tolerance composes sequentially, so that $\overline{\Pi}$ securely realizes $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ (with the same leakage bound). However, $\overline{\Pi}$ has message space $\overline{\mathcal{M}} = \mathcal{M}^2$ and key space $\overline{\mathcal{SK}} = \mathcal{SK}$. Hence, Theorem 1 yields

$$\ell(\mathcal{SK}) = \ell(\overline{\mathcal{SK}}) \geq \ell(\overline{\mathcal{M}}) - 1 = 2\ell(\mathcal{M}) - 1,$$

a contradiction. □

Connection with Bitanski *et al.* The authors in [6] show that any non-committing encryption protocol [13] suffices to securely realize $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$. It is understood that every non-committing encryption protocol must satisfy the property that both the public and the secret key are as long as the total number of message bits ever encrypted [40].

4 Relation to Adaptive Security

In this section we investigate the relation between leakage tolerance and adaptive security for arbitrary functionalities. We obtain the following results:

- We show that being secure against a semi-honest adversary which is allowed to do one adaptive corruption after the execution of the protocol is equivalent to being secure against a little leakage from a single party after the execution of the protocol.
- We conjecture that equivalence does not hold in general for *independent* leakages from the state of more than one party (but still holds for the case of joint leakage).

4.1 Corruption of a Single Party

Assume that there exists an $\mathbf{AM}_{negl(\kappa),negl(\kappa)}(O(1), \lambda(\kappa))$ argument system, which is also an argument of knowledge. Also assume there exists a family of collision-resistant hash functions $\mathcal{H} = \{H_s\}_s$ with output length $\mu(\kappa)$.

We now prove that it holds for any leakage-tolerant PKE protocol Π, as in the above section, that Π is secure against one adaptive corruption of R after the protocol execution *if and only if* Π is secure against leakage of $\approx \lambda(\kappa) + \mu(\kappa)$ bits from R after the protocol execution. Note that the above statement is clearly true when λ is large, as this would mean that the adversary is essentially leaking the entire state. Interestingly, we prove that also for a small amount of leakage (how small depends on the communication complexity of the underlying argument of knowledge) simulation-based leakage tolerance becomes identical to adaptive security.

Assume that Π is secure against one adaptive corruption of R after the protocol execution. In that case Π is also secure against any leakage queries from R after the protocol execution. This follows from [5], as leakage is weaker than adaptive corruption. We therefore focus on the other direction.

Theorem 2 (Equivalence to adaptive security)**.** *Assume the existence of* $\mathbf{AM}_{negl(\kappa),negl(\kappa)}(O(1), \lambda(\kappa))$ *argument of knowledge systems for* $\mathbf{NP}$*, where* $\lambda(\kappa) = \lambda_P(\kappa)+\lambda_V(\kappa)$*. Let* $\mathcal{H}$ *be afamily of collision-resistant hash functions with range* μ *and* Π *be a leakage-tolerant public-key encryption protocol. If* Π *tolerates* $\lambda'(\kappa) = 2\lambda_P(\kappa) + \mu(\kappa) + 1$ *bits of leakage from* R *after the protocol execution, then* Π *is passive secure against an adaptive corruption of* R *after the protocol execution.*

Proof. For simplicity we prove the theorem in the case where decryption is deterministic. One can handle randomized decryption using the same technique as in the proof of Theorem 1.

Let $\mathbf{F}_{\mathsf{SMT}}$ be the ideal functionality for secure message transmission without leakage (featuring simulator $\mathbb{S}'$), and denote with $\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}},\mathbb{S}',\mathbb{Z}'}(\kappa)$ and $\mathbf{REAL}_{\Pi,\mathbb{Z}'}(\kappa)$ the real and ideal distributions in the adaptive security game. To prove that Π is secure against one adaptive corruption of R after the protocol execution, we have to construct a simulator $\mathbb{S}'$ such that for all environments $\mathbb{Z}'$ (corrupting R at the end of the protocol execution) and for all $\kappa \in \mathbf{N}$ it holds that $\mathbf{REAL}_{\Pi,\mathbb{Z}'}(\kappa) \approx \mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}},\mathbb{S}',\mathbb{Z}'}(\kappa)$.

Note that $\mathbb{S}'$ needs to simulate first the communication (pk, c) of the protocol, and then after being given m simulates the internal state (sk, r_G) of R. We will build $\mathbb{S}'$ by constructing an environment $\mathbb{Z}$ attacking Π in the leakage game. Then we will get a simulator $\mathbb{S}$ which can simulate the attack of $\mathbb{Z}$ in the ideal world, by the assumption that Π is secure. From $\mathbb{S}$ we will then construct $\mathbb{S}'$. For later use, $\mathbb{Z}$ will depend on an environment $\mathbb{Z}'$ for the adaptive security game. Specifically we will assume that $\mathbb{Z}'$ does a normal adaptive corruption of R after the execution of the protocol. The environment $\mathbb{Z}(\mathbb{Z}')$ runs as follows.

1. $\mathbb{Z}(\mathbb{Z}')$ runs an internal copy of $\mathbb{Z}'$.
2. Until the protocol Π is running $\mathbb{Z}$ simply runs $\mathbb{Z}'$, using the same inputs to Π and delivering messages in the same way. This is possible as the real world for leakage tolerance and adaptive security are identical as long as no leakage queries and no corruption queries are issued.
3. If $\mathbb{Z}'$ does not make an adaptive corruption of R after the execution of Π terminated, then $\mathbb{Z}$ just terminates with the same guess as $\mathbb{Z}'$.
4. If $\mathbb{Z}'$ makes an adaptive corruption of R, then $\mathbb{Z}$ proceeds as follows.
   (a) Ask R to leak $h = H_s(w)$, where $w = (sk, r_G)$ and s is a random seed for the hash family $\mathcal{H}$.
   (b) Ask R to leak an argument of knowledge of $w = (sk, r_G)$ such that $h = H_s(w)$ and $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$ and $\mathsf{Dec}(sk, c) = m$. (This can be done exactly in the same way as in the proof of Theorem 1, by letting $\mathbb{Z}(\mathbb{Z}')$ play the role of the verifier and simulating the interaction with the prover via leakage queries.)
   (c) Let σ be the current state of $\mathbb{Z}'$. We can without loss of generality assume that $\mathbb{Z}'$ is deterministic and that it terminates with its guess b after seeing the internal state (sk, r_G, m) of R; we write $b = \mathbb{Z}'(\sigma, sk, r_G, m)$. Now $\mathbb{Z}$ leaks $f(sk, r_G) = \mathbb{Z}'(\sigma, sk, r_G, m)$. Note that $\mathbb{Z}$ knows m as this was a value it input to Π itself, and that it knows σ as it is $\mathbb{Z}$ which is running $\mathbb{Z}'$ (so these values can be hard-wired into the leakage function).
   (d) Finally ask R to leak an argument of knowledge for $w = (sk, r_G)$ such that $h = H_s(w)$ and $b = \mathbb{Z}'(\sigma, sk, r_G, m)$.
   (e) Output b.

Note that the total amount of leakage is twice the communication complexity of the prover for the arguments of knowledge, plus μ bits of H_s's output and one additional bit for the output of $\mathbb{Z}'$, i.e., $\lambda' = 2\lambda_P + \mu + 1$. By leakage tolerance, there exists a simulator $\mathbb{S}$ for the above $\mathbb{Z}(\mathbb{Z}')$. Since $\mathbb{S}$ is required to work for *all* environments, it in particular works for $\mathbb{Z}(\mathbb{Z}')$ for all $\mathbb{Z}'$, from which we get

$$\{\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}},\mathbb{S},\mathbb{Z}(\mathbb{Z}')}(\mathcal{F}, \kappa)\}_{\kappa \in \mathbf{N}} \approx \{\mathbf{REAL}_{\Pi,\mathbb{Z}(\mathbb{Z}')}(\mathcal{F}, \kappa)\}_{\kappa \in \mathbf{N}}, \tag{2}$$

which we use later. Note, first, however, that by leakage resilience, it holds that in the view simulated by $\mathbb{S}$, the arguments of knowledge accept with probability negligibly close to 1, or we could easily construct a distinguisher between the real world and the simulation. Furthermore, the distributions of the bit b in the real world and in the simulation are computationally indistinguishable.

Consider now the following simulator $\mathbb{S}'$, interacting with $\mathbf{F}_{\mathsf{SMT}}$ in the adaptive security game.

1. Until the protocol Π is running, simulate using $\mathbb{S}$.

2. When $\mathbb{Z}'$ adaptively corrupted R, receive m from the ideal functionality.

3. Give the leakage function $H_s(\cdot)$ to $\mathbb{S}$ to make it generate a simulated value h. Note that $\mathbb{S}$ is a simulator for the ideal world in the definition of leakage tolerance, i.e., it might issue leakage queries $f_{\mathbb{S}}$ to the ideal functionality. Answer these with $f_{\mathbb{S}}(m)$ — the trick is that $\mathbb{S}'$ at this point knows m.

4. Similarly, make $\mathbb{S}$ give an argument of knowledge of $w = (sk, r_G)$ such that $h = H_s(w)$ and $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$ and $\mathsf{Dec}(sk, c) = m$.

5. By an above comment we know that this argument accepts except with negligible probability, so $\mathbb{S}'$ can extract from $P^* := \mathbb{S}$ a witness $w = (sk, r_G)$ such that $h = H_s(w)$ and $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$ and $\mathsf{Dec}(sk, c) = m$.

6. Output w.

It only remains to argue that the w output by $\mathbb{S}'$ has a distribution computationally indistinguishable from the internal state of R in the real world. Assume for the sake of contradiction that it is not. Then there exists an environment $\mathbb{Z}'$ which can distinguish. This means that $b = \mathbb{Z}'(w)$ has distinguishable distributions in the real world and the simulation (for the adaptive security game). Consider then the adversary $\mathbb{Z}(\mathbb{Z}')$ for the leakage resilience game.

Claim 1. $\{\mathbf{REAL}_{\Pi,\mathbb{Z}(\mathbb{Z}')}(\mathcal{F}, \kappa)\}_{\kappa \in \mathbf{N}} \equiv \{\mathbf{REAL}_{\Pi,\mathbb{Z}'}(\kappa)\}_{\kappa \in \mathbf{N}}$.

Proof of claim. In words, the output distribution of $\mathbb{Z}(\mathbb{Z}')$ in the real world of the leakage game and $\mathbb{Z}'$ in the real world of the adaptive security game are the same. This follows simply by construction of $\mathbb{Z}(\mathbb{Z}')$, which runs $\mathbb{Z}'$ on the internal state w of R. □

Claim 2. $\{\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}},\mathbb{S},\mathbb{Z}(\mathbb{Z}')}(\mathcal{F}, \kappa)\}_{\kappa \in \mathbf{N}} \approx \{\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}},\mathbb{S}',\mathbb{Z}'}(\kappa)\}_{\kappa \in \mathbf{N}}$.

Proof of claim. In words, the output distribution of $\mathbb{Z}(\mathbb{Z}')$ in the ideal world of the leakage game and $\mathbb{Z}'$ in the ideal world of the adaptive security game are computationally indistinguishable.

The output distribution of $\mathbb{Z}(\mathbb{Z}')$ in the ideal world of the leakage game is the value b simulated by $\mathbb{S}$. The output distribution of $\mathbb{Z}'$ in the ideal world of the adaptive security game is $\mathbb{Z}'$ applied to the value w extracted from $P^* := \mathbb{S}$. We need to prove that these two distributions are indistinguishable. To analyze the distribution of the b returned by $\mathbb{S}$ in the simulation of the leakage game, notice that since both the arguments of knowledge given by $\mathbb{S}$ are accepting, we can extract $w = (sk, r_G)$ and $w' = (sk', r'_G)$ such that $h = H_s(w)$ and $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$ and $\mathsf{Dec}(sk, c) = m$, and $h = H_s(w')$ and $b = \mathbb{Z}'(\sigma, sk', r'_G, m)$. From $H_s(\cdot)$ being collision resistant we can assume that $w = w'$, so we conclude that it holds from the w extracted from the first argument of knowledge generated by $\mathbb{S}$ that $w = (sk, r_G)$, $(pk, sk) = \mathsf{Gen}(1^\kappa; r_G)$, $\mathsf{Dec}(sk, c) = m$ and $b = \mathbb{Z}'(\sigma, sk, r_G, m)$. This means that unless collision resistance of $H_s(\cdot)$ is broken, the output distribution of $\mathbb{Z}(\mathbb{Z}')$ in the ideal world of the leakage game and $\mathbb{Z}'$ in the ideal world of the adaptive security game are the same. □

The two claims above together with the assumption that $\mathbb{Z}'$ can distinguish, imply that that $\mathbb{Z}(\mathbb{Z}')$ has distinguishable outputs in the real world and the ideal world for the leakage game, contradicting Eq. (2) above. From this we conclude that $\{\mathbf{REAL}_{\Pi,\mathbb{Z}'}(\kappa)\}_{\kappa \in \mathbf{N}} \approx \{\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}},\mathbb{S}'\mathbb{Z}'}(\kappa)\}_{\kappa \in \mathbf{N}}$ for all environments $\mathbb{Z}'$, which proves the theorem. □

4.2 Corruption of Many Parties

We note that the proof technique from the previous section can be easily generalized to show that an arbitrary n-party protocol Π is secure against one adaptive corruption after the protocol execution *if and only if* Π tolerates $poly(\log \kappa)$ bits of leakage from one of the parties after the protocol execution.

A natural question is whether the above proof technique can be generalized to work also for an arbitrary protocol and if we allow that many parties can be corrupted/leaked from after the protocol execution. We conjecture that the answer is, in general, negative. The main idea would be to let the environment ask each party to leak an argument of knowledge of an internal state consistent with its inputs and outputs. A simulator which can simulate such an argument could also "by extracting itself" have output the entire internal state. We cannot, however, perform the trick where we send the distinguisher $\mathbb{Z}'$ into the parties to leak $\mathbb{Z}'(w)$, as now a distinguisher for the adaptive security game should have access to $(w_1, \ldots, w_n)$, where w_i is the internal state of party i, and $(w_1, \ldots, w_n)$ is not sitting inside a single party, so $\mathbb{Z}'(w_1, \ldots, w_n)$ cannot *per se* be computed using short leakages $f_1(w_1), \ldots, f_n(w_n)$.[2] Hence, it seems hard to enforce the extracted internal state to be indistinguishable from the internal state in the real world.

References

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

[2] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM J. Comput.*, 38(5):1661–1694, 2008.

[3] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, pages 390–420, 1992.

[4] 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.

[5] 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.

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

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

[8] Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In *CRYPTO*, pages 316–334, 2013.

[9] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In *STOC*, pages 1235–1254, 2012.

[10] Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. In *DISC*, pages 181–196, 2011.

[2] It is not hard to see, though, that the equivalence hold in case the environment is allowed joint leakage on the state of the players.

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

[12] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[13] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.

[14] Özgür Dagdelen and Daniele Venturi. A second look at Fischlin's transformation. In *AFRICACRYPT*, pages 356–376, 2014.

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

[16] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.

[17] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

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

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

[20] Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In *TCC*, pages 230–247, 2012.

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

[22] 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.

[23] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.

[24] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In *INDOCRYPT*, pages 60–79, 2012.

[25] 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.

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

[27] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO*, pages 297–315, 2011.

[28] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.

[29] Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In *TCC*, pages 107–124, 2011.

[30] Shai Halevi, Steven Myers, and Charles Rackoff. On seed-incompressible functions. In *TCC*, pages 19–36, 2008.

[31] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.

[32] Abhishek Jain and Krzysztof Pietrzak. Parallel repetition for leakage resilience amplification revisited. In *TCC*, pages 58–69, 2011.

[33] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[34] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992.

[35] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pages 104–113, 1996.

[36] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

[37] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[38] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.

[39] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. *IACR Cryptology ePrint Archive*, 2009:105, 2009.

[40] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, pages 111–126, 2002.

[41] Jesper Buus Nielsen, Daniele Venturi, and Angela Zottarel. Leakage-resilient signatures with graceful degradation. In *Public Key Cryptography*, pages 362–379, 2014.

[42] Omkant Pandey. Achieving constant round leakage-resilient zero-knowledge. In *TCC*, pages 146–166, 2014.

[43] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.

[44] Krzysztof Pietrzak and Douglas Wikström. Parallel repetition of computationally sound protocols revisited. *J. Cryptology*, 25(1):116–135, 2012.

[45] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and countermeasures for smart cards. In *E-smart*, pages 200–210, 2001.

[46] Dominique Unruh. Quantum proofs of knowledge. In *EUROCRYPT*, pages 135–152, 2012.

[47] Hoeteck Wee. On round-efficient argument systems. In *ICALP*, pages 140–152, 2005.