Leakage-Resilient Zero Knowledge

Sanjam Garg, Abhishek Jain*, and Amit Sahai*

UCLA, {sanjamg,abhishek,sahai}@cs.ucla.edu

Abstract. In this paper, we initiate a study of zero knowledge proof systems in the presence of side-channel attacks. Specifically, we consider a setting where a cheating verifier is allowed to obtain arbitrary bounded leakage on the *entire state* (including the witness and the random coins) of the prover *during the entire protocol execution*. We formalize a meaningful definition of *leakage-resilient zero knowledge* (LR-ZK) proof system, that intuitively guarantees that *the protocol does not yield anything beyond the validity of the statement and the leakage obtained by the verifier*.

We give a construction of LR-ZK interactive proof system based on standard general assumptions. To the best of our knowledge, this is the first instance of a cryptographic *interactive protocol* where the adversary is allowed to perform leakage attacks during the protocol execution on the *entire state* of honest party (in contrast, prior work only considered leakage *prior* to the protocol execution, or very limited leakage *during* the protocol execution). Next, we give an LR-NIZK proof system based on standard number-theoretic assumptions.

Finally, we demonstrate the usefulness of our notions by giving two concrete applications:

- We initiate a new line of research to relax the assumption on the "tamper-proofness" of hardware tokens used in the design of various cryptographic protocols. In particular, we give a construction of a universally composable multiparty computation protocol in the *leaky token model* (where an adversary in possession of a token is allowed to obtain arbitrary bounded leakage on the *entire state* of the token) based on standard general assumptions.
- Next, we give simple, generic constructions of *fully* leakage-resilient signatures in the bounded leakage model as well as the continual leakage model. Unlike the recent constructions of such schemes, we also obtain security in the "noisy leakage" model.

1 Introduction

Zero knowledge proof systems, introduced in the seminal work of Goldwasser, Micali and Rackoff [31], have proven fundamental to cryptography. Very briefly, a zero knowledge proof system is an interactive proof between two parties – a

* Research supported in part from a DARPA/ONR PROCEED award, NSF grants 0916574 and 0830803, a Xerox Foundation Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant.

prover, and a verifier – with the remarkable property that the verifier does not learn anything beyond the validity of the statement being proved. Subsequent to their introduction, zero knowledge proofs have been studied in various adversarial settings such as concurrency attacks [23], malleability attacks [22], to list a few, with very successful results. Over the years, zero knowledge proofs (and its various strengthened notions) have turned to be extremely useful, finding numerous applications in the design of various cryptographic protocols.

We note that the standard definition of zero knowledge proofs, like most classical security notions, assumes that an adversary is given only black-box access to the honest party algorithms. Unfortunately, over the last two decades, it has become increasingly evident that such an assumption may be unrealistic when arguing security in the real world where the physical implementation (e.g. on a smart card or a hardware token) of an algorithm is under attack. Motivated by such a scenario, in this paper, we initiate a study of zero knowledge proof systems in the presence of *side-channel attacks* [46, 5, 61, 28, 37]. Specifically, we study zero knowledge proofs in the intriguing setting where a cheating verifier, in addition to receiving a proof of some statement, is able to obtain arbitrary bounded leakage on the *entire state* (*including the witness and the random coins*) of the prover *during the entire protocol execution*. We note that while there has been an extensive amount of research work on leakage-resilient cryptography in the past few years, to the best of our knowledge, almost all prior work has either been on leakage resilient *primitives* such as encryption and signature schemes [24, 2, 59, 21, 4, 56, 44, 18, 25, 3, 45, 10, 19, 20, 48, 51, 9, 47], or leakage-resilient (and tamper-resilient) *devices* [41, 40, 26, 1], while very limited effort has been dedicated towards constructing leakage-resilient *interactive protocols*. To the best of our knowledge, the recent works on correlation extractors [39], and leakage-resilient identification and authenticated key agreement protocols [4, 20, 19] come closest to being considered in the latter category. However, we stress that in all these works, either leakage attacks are allowed only *prior* to the protocol execution, or very limited leakage is allowed *during* the protocol execution; in contrast, we consider the setting where the adversary can obtain leakage on the *entire state* of the honest party during the protocol execution.

We find it imperative to stress that handling leakage attacks on interactive protocols can be particularly challenging. On the one hand, for the leakage attacks to be meaningful, we would want to allow leakage on the secret state of the protocol participants. However, the state of a party typically includes a secret value (witness and random coins of the prover in the case of zero knowledge proofs) and any leakage on that secret value might immediately violate a security property (e.g., the zero knowledge property) of the protocol. Then, coming back to setting of zero knowledge proofs, it is not immediately clear how to even define "leakage-resilient zero knowledge."

How to define Leakage-Resilient Zero Knowledge? One possibility is to pursue an assumption such as *only computation leaks information* [53] (i.e., assuming that there is no leakage in the absence of computation). While this is a

valuable and interesting approach, we note that this assumption is often problematic (e.g. cold-boot attacks [37]). In our work here, therefore, we do *not* make any such assumption. We seek a general definition maximizing the potential applicability of that definition to different application scenarios.

Another possibility is to allow a "leakage-free pre-processing phase" prior to the actual protocol execution, in an attempt to render the leakage attacks during the protocol useless. We note, however, that allowing pre-processing would limit the applicability of our notion. In particular, such a definition would be problematic for scenarios where the statement to be proven is generated "online" (thereby eliminating the possibility of pre-processing the witness "safely"). Furthermore, we give strong evidence that such an approach is unlikely to yield better guarantees than what we are able to achieve (see the full version for further discussion on this issue).

Indeed, our goal is to obtain a meaningful and appropriate definition of zero knowledge in the model where an adversarial verifier can obtain leakage on any content (state) of the prover machine at any time. We do not consider any "leakage-free" time-period; in particular, any pre-processing phase is subject to leakage as well. However, in such a setting, it is important to note that since the adversary could simply choose to leak on the *witness* (and no other prover state), the zero knowledge *simulator* must be able to obtain similar amount of leakage in order to perform correct simulation. We shall see that even with this limitation, our notion turns out to be both quite nontrivial to obtain and very useful in application scenarios.

Our Definition – Informally. To this end, we consider a definition of leakage-resilient zero knowledge that provides the intuitive guarantee that *the protocol does not yield anything beyond the validity of the statement and the leakage obtained by the adversary.* In other words, whatever an adversary "learns" from the protocol (with leakage) should be no more than what she can learn from only the leakage without running the protocol. To formalize the above intuition, as a first step, we consider a *leakage oracle* that gets as private input the witness of the honest prover; the zero knowledge simulator is then given access to such a leakage oracle. More concretely, we consider a parameter λ, and say that an interactive proof system is λ-*leakage-resilient zero knowledge* (LR-ZK) if for every cheating verifier, there exists a simulator with access to a leakage oracle (that gets the honest prover's witness as private input) that outputs a view of the verifier (indistinguishable from the real execution), with the following requirement. Let ℓ bits be an upper bound on the total amount of leakage obtained by the adversarial verifier. Then the simulator is allowed to obtain at most $\lambda \cdot \ell$ bits of leakage. (In the full version, we show that constructing an LR-ZK proof system with $\lambda < 1$ is in fact impossible.)

Applications of Our Definition. Now that we have a definition for LR-ZK proof system, one may question how meaningful it is. As we now discuss, the above definition indeed turns out to be very useful. Intuitively, our definition is appropriate for a scenario where a leakage-resilient primitive A is being used in conjunction with a zero knowledge proof system (where the proof system is used

to prove some statement about A), in the design of another cryptographic protocol B. The reason for this is that our definition of LR-ZK allows us to directly reduce the leakage-resilience property of B on the leakage-resilience property of A.

As an application of our LR-ZK interactive proof system, we first construct a universally composable (UC) multiparty computation protocol in the *leaky token model* (which is a relaxation of the model of Katz [43] in that a malicious token user is now allowed to leak arbitrary bounded information on the *entire state* of the token). Very briefly, we use *leakage-resilient hard relations* [20] and hardware tokens that implement the prover algorithm of our LR-ZK proof system where we prove the validity of an instance of the hard relation; then the leakage on the state of the token can be easily "reduced" to leakage on (the witness corresponding to) an instance of the hard relation.

Next, we are able to extend the notion of LR-ZK to the non-interactive setting in a natural way. Then, as an application of LR-NIZKs, we give generic constructions of *fully* leakage-resilient (FLR) signature schemes (where leakage is allowed on the *entire state* as opposed to only the secret key). Very briefly, we use leakage-resilient hard relations in conjunction with "simulation-extractable" LR-NIZKs (see below); we are then able to reduce the leakage-resilience property of the signature scheme to that of the hard relation. We now summarize our results.

1.1 Our Results

We first study the possibility of constructing leakage-resilient zero knowledge protocols and obtain the following results:

- We construct a $(1+\epsilon)$-leakage-resilient zero knowledge interactive proof system (where ϵ is any positive constant) based on standard general assumptions (specifically, the existence of a statistically hiding commitment scheme that is public-coin w.r.t. the receiver). To the best of our knowledge, this is the first instance of a cryptographic *interactive protocol* where an adversary is allowed to obtain arbitrary bounded leakage on the *entire state* of the honest parties *during* the protocol execution.
- Next, we consider the non-interactive setting and show that any NIZK proof system with *honest prover state reconstruction* property [36] is an LR-NIZK proof system for $\lambda = 1$. As a corollary, we obtain an LR-NIZK proof system from [36] based on the decisional linear assumption.

We supplement our above positive results by proving the impossibility of constructing an LR-ZK proof (or argument) system for $\lambda < 1$. Then, as applications of leakage-resilient zero knowledge, we obtain the following results:

- We initiate a new line of research to relax the assumption on the "tamper-proofness" of hardware tokens used in the design of various cryptographic protocols. In particular, assuming semi-honest oblivious transfer, we give a

construction of a universally composable (UC) multiparty computation protocol in the *leaky token model*, where the token user is allowed to obtain arbitrary bounded leakage on the *entire state* of the token. We stress that all prior works on designing cryptographic protocols using hardware tokens, including the work on UC secure computation [43, 14, 54, 15], made the implicit assumption that the tokens are completely leakage-resilient.

– Next, we extend the notion of leakage-resilient NIZKs to incorporate the property of *simulation-extractability* [63, 17] (also see [58] in the context of interactive proofs), in particular, the "true" variant [20]. We are then able to adapt the approach of Katz and Vaikuntanathan [44], and in particular, Dodis et al [20, 19] (who use a leakage-resilient hard relation in conjunction with a true simulation-extractable NIZK argument system to construct leakage-resilient signatures) to the setting of *full leakage*. As a result, we obtain simple, generic constructions of *fully* leakage-resilient signature schemes in the bounded leakage model as well as the continual leakage model. Similar to [20, 19], our signature scheme inherits the leakage-resilience properties (and the leakage bounds) of the hard relation used in its construction.[1] In contrast to the recent constructions of FLR signature schemes by [51, 9, 47] in the standard model[2], our scheme is also secure in the *noisy leakage model* [56]. We supplement our result by showing that a true simulation-extractable leakage-resilient NIZK argument system is implied by the UC-NIZK of Groth et al. [36], which can be based on the decisional linear assumption.

1.2 Our Techniques

We now briefly discuss the main techniques used to obtain our positive results on leakage-resilient zero knowledge proof systems. Recall that our goal is to realize a definition where a cheating verifier does not learn anything from the protocol beyond the validity of the statement and the leakage information obtained from the prover. Further, recall that in our definition, simulator is given access to a leakage oracle that gets the honest prover's witness as private input and accepts leakage queries on the witness string. (In contrast, the verifier is allowed to make leakage queries on the entire state, including the witness and the random coins used by the prover thus far in the protocol execution.) Then, during the simulation, on receiving a leakage query from the verifier, our simulator attempts to convert it into a "valid" query to the leakage oracle. Now, note that the simulator may be cheating in the protocol execution (which is typically the case since it does not possess a valid witness); then, since the verifier can arbitrarily leak on both the witness and the random coins (which completely determine the actions of the prover thus far), at every point in the protocol execution, the simulator

[1] As such, if we use the key pairs from the encryption scheme of Lewko et al [47] as a hard relation, then our signature scheme can tolerate leakage during the update process as well.

[2] Earlier, FLR signature schemes were constructed either only in the random oracle model [4, 20, 10], or were only "one-time" [44]

must find a way to "explain its actions so far". Note that this is reminiscent of *adaptive security* [7, 11, 13, 50] in the context of secure computation protocols. We stress, however, that adaptive security does not suffice to achieve the property of leakage-resilient zero knowledge in the interactive proofs setting, as we explain below.

Recall that the notion of adaptive security corresponds to the setting where an adversary is allowed to corrupt parties *during* the protocol execution (as opposed to static corruption, where the parties can only be corrupted *before* the protocol begins). Once a party is corrupted, the adversary learns the entire state (including the input and random coins) of that party. The adversary may choose to corrupt several parties (in the case of multi-party protocols) throughout the course of the protocol. The notion of adaptive security guarantees security for the remaining uncorrupted parties.

While adaptive corruption itself is not our focus, note that in our model, a cheating verifier may obtain leakage on the prover's state at *several points* during the protocol execution. Furthermore, the honest prover may not even be aware as to what was leaked. Our goal is to guarantee that the adversary does not learn anything beyond the leaked information. Then, in order to provide such a guarantee, note that our simulator must *continue to simulate the prover even after leakage happens*, in a way that is consistent with the leaked information even though it does not know the prover's witness or what information was leaked. In contrast, the simulator for adaptively secure protocols does *not* need to simulate a party once it is corrupted.[3] In summary, we wish to guarantee some security for the honest party even *after* leakage happens, while adaptive security does not provide any such guarantees. We stress that this difference is crucial, and explains why known techniques for achieving adaptive security do not suffice for our purposes. Nevertheless, as we explain below, adaptive security serves as a good starting point for our purpose.

Recall that the main challenge in the setting of adaptive security is that whenever an adversary chooses to corrupt a party, the simulator must be able to explain its random coins, in a way that is consistent with the party's input and the messages it generated so far in the protocol. The main technique for overcoming this challenge is to allow the simulator to *equivocate*. For our purposes, we will also make use of equivocation so that the leakage queries can be answered correctly by the simulator. However, since our simulator would need to simulate the prover even after leakage happens (without the knowledge of the prover's witness or the information that was leaked), we do not want this equivocation to interfere with the simulation of prover's messages. In other words we want to be able to simulate the prover's messages independent of what information is being leaked but still remain consistent with it. Our solution is to have two separate and independent ways of cheating at the simulator's disposal. It will use one way to cheat in the protocol messages and the second way is reserved for answering the leakage queries correctly. Furthermore, we would need to make

[3] Indeed, for this reason, known adaptively secure ZK protocols are not leakage-resilient.

sure that the simulator does not "step on its own toes" when using the two ways to cheat *simultaneously*.

We now briefly discuss the actual construction of our protocol in order to illustrate the above ideas. We recall two well-known ways of constructing constant-round zero knowledge protocols – the Feige-Shamir [27] approach of using equivocal commitments (also used in adaptive security), and the Goldreich-Kahan [29] approach of requiring the verifier to commit to its challenges in advance. Now, armed with the intuition that our simulator will need two separate ways of cheating, we use both the above techniques *together*. Our protocol roughly consists of two phases: in the first phase, the verifier commits to a challenge string using a standard challenge-response based extractable commitment scheme (in a manner similar to [62]); in the second phase, we execute the Blum-Hamiltonicity protocol instantiated with an equivocal commitment scheme. While leakage during the first phase can be handled easily by our simulator, handling leakage during the second phase makes use of the ideas discussed above.

Unfortunately, although the above construction seems to satisfy most of our requirements, it fails on the following account. Recall that our goal is to obtain a leakage-resilient zero knowledge protocol with nearly optimal precision (i.e., $\lambda = 1 + \epsilon$) with respect to the leakage queries of the simulator. Now note that in the above construction, the simulator would need to extract the verifier's challenge in the first phase by means of rewinding before proceeding to phase two of the protocol. Then, depending upon the verifier's behavior, the simulator may need to perform several rewinds in order to succeed in extraction. Now, note that a cheating verifier may be able to make a *different* leakage query during each rewind, thus forcing our simulator to make a new query as well to its leakage oracle. As a result, depending upon the number of such rewinds, the total leakage obtained by the simulator may potentially become a polynomial factor of the leakage obtained by the adversary in a real execution.

In order to obtain a precision in the leakage queries of the simulator, we borrow techniques from the work on *precise zero knowledge* pioneered by Micali and Pass [52]. We remark that in the context of precise ZK, (for fundamental reasons of modeling) it is typically not possible to obtain a precision of almost 1. In our case, however, we are able to achieve a precision of $\lambda = 1 + \epsilon$ (where ϵ is any positive constant) with respect to the leakage queries of the simulator.

Finally, we note that in the case of non-interactive zero knowledge, since the simulator does not need to simulate any "future messages" after the leakage, we are indeed able to show that an adaptively secure NIZK is also a leakage-resilient NIZK. Specifically, we show that any NIZK with *honest prover state reconstruction* property, as defined by Groth et al. [36] (in the context of adaptive security), is also a leakage-resilient NIZK with $\lambda = 1$.

Related Work. In a very recent and exciting concurrent work, Canetti et al. consider a model of leakage in the context of UC protocols [8].

2 Leakage-Resilient Zero Knowledge: Interactive Case

We consider the scenario where a malicious verifier can obtain arbitrary bounded leakage on the entire state (including the witness and the random coins) of the prover during the protocol execution. We wish to give a meaningful definition of zero knowledge interactive proofs in such a setting. To this end, we first modify the standard model for zero knowledge interactive proof system in order to incorporate leakage attacks and then proceed to give our definition.

We model the prover P and the verifier V as interactive turing machines that have the ability to flip coins during the protocol execution (such that the random coins used by a party in any round are determined only at the beginning of that round). In order to incorporate leakage attacks, we allow a malicious verifier V^* to make *adaptive* leakage queries on the state of the prover during the protocol execution. A leakage query to the prover consists of an efficiently computable function f_i (described as a circuit), to which the prover responds with $f_i(\texttt{state})$, where `state` is a variable that denotes the "current state" of the prover at any point during the protocol execution. The variable `state` is initialized to the witness of the prover. At the completion of each step of the protocol execution (that corresponds to the prover sending a protocol message to the verifier), the random coins used by the prover in that step are appended to `state`. That is, $\texttt{state} := \texttt{state} \| r_i$, where r_i denote the random coins used by the prover in that step. The verifier may make any arbitrary polynomial number of such leakage queries during the protocol execution. Unlike prior works, we do not require an a-priori bound on the total leakage obtained by the verifier in order to satisfy our definition (described below). Nevertheless, in order for our definition to be meaningful, we note that the total leakage obtained by the verifier must be smaller than the witness size.

We model the zero knowledge simulator $\mathcal{S}$ as a PPT machine that has access to a leakage oracle $L_w^{k,\lambda}(\cdot)$ that is parameterized by the honest prover's witness w, a leakage parameter λ (see below), and the security parameter k. A query to the oracle consists of an efficiently computable function $f(\cdot)$, to which the oracle answers with $f(w)$. In order to bound the total leakage available to the simulator, we consider a parameter λ and require that if the verifier obtains ℓ bits of total leakage in the real execution, then the total leakage obtained by the simulator (from the leakage oracle) must be bounded by $\lambda \cdot \ell$ bits. Finally, we require that the view output by the simulator be computationally indistinguishable from the verifier's view in the real execution. We formalize this in the definition below.

Definition 1 (Leakage-Resilient Zero Knowledge). *An interactive proof system $\langle P, V \rangle$ for a language $\mathcal{L}$ with a witness relation $\mathcal{R}$ is said to be* λ-leakage-resilient zero knowledge *if for every* PPT *machine V^* that makes any arbitrary polynomial number of leakage queries on P's state (in the manner as described above) with ℓ bits of total leakage, there exists a* PPT *algorithm $\mathcal{S}$ that obtains at most $\lambda \cdot \ell$ bits of total leakage from a leakage oracle $L_w^{k,\lambda}(\cdot)$ (as defined above) such that for every $(x, w) \in \mathcal{R}$, every $z \in \{0,1\}^*$, $\text{view}_{V^*}(x, z)$ and $\mathcal{S}^{L_w^{k,\lambda}(\cdot)}(x, z)$ are computationally indistinguishable.*

Some observations on the above definition are in order.

Leakage parameter λ. Note that when $\lambda = 0$, no leakage is available to the simulator (as is the case for the standard zero knowledge simulator). In this case, our definition guarantees the standard zero knowledge property. It is not difficult to see that it is impossible to realize such a definition. In fact, as we show in the full version, it is impossible to realize the above definition for any $\lambda < 1$, where ϵ is any constant less than 1. On the other hand, in Section 2.1, we give a positive result for $\lambda = 1 + \epsilon$, where ϵ is any positive constant. The meaningfulness of our positive result stems from the observation that when λ is close to 1, very roughly, our definition guarantees that *a malicious verifier does not learn anything from the protocol beyond the validity of the statement being proved and the leakage obtained from the prover.*

Leakage-oblivious simulation. Note that in our definition of leakage resilient zero-knowledge, (apart from the total output length) there is no restriction on the nature of leakage queries that the simulator may make to the leakage oracle. Then, since the simulator has indirect access to the honest prover's witness (via the leakage oracle), it may simply choose to leak on the witness (regardless of the leakage queries of the verifier) in order to help with the simulation of protocol messages instead of using the leakage oracle to *only* answer the leakage queries of the verifier. We stress that this issue should not affect any potential application of leakage resilient zero-knowledge that one may think of. Nonetheless, we think that this is an important issue since it relates to the meaningfulness of the definition. To this end, we note that this issue can easily handled by putting a restriction on how the simulator accesses the leakage oracle. Specifically, we can model the interaction between the simulation and the oracle such that the simulator is not allowed to look at the oracle's responses to its queries. The simulator is still allowed to look at the leakage queries of the verifier, and use them to create new queries for the oracle; however, the oracle's responses are sent directly to the verifier and the simulator does not get to seem them. We call such simulators *leakage-oblivious*. We note that the simulator that we construct for our protocol $\langle P, V \rangle$ (described in the next subsection) is leakage-oblivious.[4]

2.1 Our Protocol

We now proceed to give our construction of a leakage-resilient zero knowledge interactive proof system as per Definition 1. Very roughly speaking, our protocol can be seen as a combination of Feige-Shamir [27] and Goldreich-Kahan [29], in that we make use of equivocal commitments from the prover's side, as well as require the verifier to commit to all its challenges in advance. Note that while either of the above techniques would suffice for standard simulation, interestingly, we need to use them *together* to help the simulator handle leakage queries from a cheating verifier. We now describe our protocol in more detail.

[4] Indeed, since we cannot rule out of obfuscation of arbitrary functionalities, we do not know how to obtain a formal proof without making the simulator leakage-oblivious.

Let P and V denote the prover and verifier respectively. Our protocol $\langle P, V \rangle$ proceeds in three stages, described as follows. In *Stage 1*, V commits to its challenge and a large random string r' using a challenge-response based PRS [60] style preamble instantiated with a public-coin statistically hiding commitment scheme [57, 38, 16]. In *Stage 2*, P and V engage in coin-flipping (that was initiated in Stage 1 when V committed to r') to jointly compute a random string r. Finally, in *Stage 3*, P and V run k (where k denotes the security parameter) parallel repetitions of the 3-round Blum Hamiltonicity protocol, where P uses Naor's commitment scheme [55] to commit to the permuted graphs in the first round. Here, for each bit commitment i, P uses a different substring r_i (of appropriate length) of r as the first message of Naor's commitment scheme. Protocol $\langle P, V \rangle$ is described in Figure 1. Intuitively, the purpose of multiple challenge response slots in Stage 1 is to allow the simulator to extract the values committed by V^* with minimal use of the leakage oracle. With the knowledge of the extracted values, the simulator can force the output of the coin-flipping to a specific distribution of its choice. This, in turn, allows the simulator to convert Naor's commitment scheme into an *equivocal* commitment scheme during simulation.

Theorem 1. *If public-coin statistically hiding commitment schemes exist, then the protocol $\langle P, V \rangle$, parameterized by ϵ, is a $(1+\epsilon)$-leakage-resilient zero knowledge proof system.*

We note that statistically hiding commitment schemes imply one-way functions, which in turn suffice for Naor's statistically binding commitment scheme used in our construction. In the interest of space, we give a proof of Theorem 1 in the full version.

3 Leakage-Resilient NIZK

We now turn our attention to the setting of non-interactive zero knowledge proof systems. We consider the scenario where a malicious verifier can obtain arbitrary leakage on the witness and the random coins used by an honest prover to generate the proof string. To model leakage attacks, we allow the cheating verifier to make adaptive leakage queries on the honest prover's witness and the random coins used to generate the proof string. A leakage query to the prover consists of an efficiently computable function f, to which the prover replies with $f(w\|r)$, where w and r denote the prover's witness and random coins respectively. It is easy to see that in the non-interactive proofs setting, a cheating verifier who is allowed multiple leakage queries enjoys no additional power than one who is allowed only one leakage query. Therefore, for simplicity of exposition, from now on, we only consider cheating verifiers who make only one leakage query. We note that our definition given below can be easily adapted to incorporate multiple leakage queries.

We model the zero knowledge simulator $\mathcal{S}$ as a PPT machine that has access to a leakage oracle $L_w^k(\cdot)$ that is parameterized by the honest prover's witness w and the security parameter k. (Unlike the interactive proofs setting, here we do

Common Input: A k-vertex graph G.
Private Input to P: A Hamiltonian Cycle H in graph G.
Parameters: Security parameter 1^k, $n = \omega(\log(k))$, $t = 3k^4$, $\epsilon > 0$. Without loss of generality, we assume that $\frac{1}{\epsilon}$ is an integer.

Stage 1 (Commitment phase)

$V \rightleftarrows P$**:** Commit to a t-bit random string r' and $(\frac{n^2}{\epsilon})$-pairs of random shares $\left\{r'^0_{i,j}, r'^1_{i,j}\right\}_{i=1,j=1}^{i=\frac{n}{\epsilon},j=n}$ (such that $r'^0_{i,j} \oplus r'^1_{i,j} = r'$ for every $i \in [\frac{n}{\epsilon}], j \in [n]$) using a public-coin statistically hiding commitment scheme. Similarly commit to a k-bit random string ch and $(\frac{n^2}{\epsilon})$-pairs of random shares $\left\{ch^0_{i,j}, ch^1_{i,j}\right\}_{i=1,j=1}^{i=\frac{n}{\epsilon},j=n}$ (such that $ch^0_{i,j} \oplus ch^1_{i,j} = ch$ for every $i \in [\frac{n}{\epsilon}], j \in [n]$) using a public-coin statistically hiding commitment scheme.

Challenge-response slots: For every $i \in [\frac{n}{\epsilon}]$,

$P \rightarrow V$**:** Choose n-bit random strings $\alpha_i = \alpha_{i,1}, \ldots, \alpha_{i,n}$ and $\beta_i = \beta_{i,1}, \ldots, \beta_{i,n}$. Send α_i, β_i to V.

$V \rightarrow P$**:** For every $j \in [n]$, V^* decommits to $r'^{\alpha_{i,j}}_{i,j}$ and $ch^{\beta_{i,j}}_{i,j}$.

Stage 2 (Coin-flipping completion phase)

$P \rightarrow V$**:** Choose a t-bit random string r'' and send it to V.

$V \rightarrow P$**:** Decommit to r' and $r'^0_{i,j}$, $r'^1_{i,j}$ for every $i \in [\frac{n}{\epsilon}], j \in [n]$. Let $r = r' \oplus r''$.

Stage 3 (Blum Hamiltonicity protocol)

$P \rightarrow V$**:** Let $r = r_1, \ldots, r_{k^3}$, where $|r_i| = 3k$ for every $i \in [k^3]$. For every $i \in [k]$,

- Choose a random permutation π_i and prepare an isomorphic copy of G, denoted $G_i = \pi_i(G)$.
- For every $j \in [k^2]$, commit to bit b_j in the adjacency matrix of G_i using Naor's commitment scheme with $r_{i \times j}$ as the first message.

$V \rightarrow P$**:** Decommit to ch and $ch^0_{i,j}$, $ch^1_{i,j}$ for every $i \in [\frac{n}{\epsilon}], j \in [n]$.

$P \rightarrow V$**:** Let $ch = ch_1, \ldots, ch_k$. For each $i \in [k]$, if $ch_i = 0$, decommit to every edge in G_i and reveal the permutation π_i. Else, decommit to the edges in the Hamiltonian Cycle in G_i.

Fig. 1. Protocol $\langle P, V \rangle$

not consider the leakage parameter λ for simplicity of exposition.) The leakage oracle accepts queries of the form f (where $f(\cdot)$ is an efficiently computable function) and outputs $f(w)$. In order to bound the total leakage available to the simulator, we require that if the verifier obtains ℓ bits of total leakage from the honest prover, then the total leakage obtained by the simulator (from the leakage oracle) must be bounded by ℓ bits.

We now setup some notation. Let $\mathcal{R}$ be an efficiently computable relation that consists of pairs (x, w), where x is called the statement and w is the witness. Let $\mathcal{L}$ denote the language consisting of statements in $\mathcal{R}$. Recall that a non-interactive proof system for a language $\mathcal{L}$ consists of a setup algorithm K, a prover P and a verifier V. The setup algorithm K generates a common reference string σ. The prover P takes as input (σ, x, w) and checks whether $(x, w) \in \mathcal{R}$;

if so, it produces a proof string π, else it outputs `fail`. The verifier V takes as input (σ, x, π) and outputs 1 if the proof is valid, and 0 otherwise.

Definition 2 (LR-NIZK). *A non-interactive proof system (K, P, V) for a* PPT *relation $\mathcal{R}$ is said to be a* leakage-resilient NIZK *if there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ such that for all adversaries $\mathcal{A}$,*

$$\Pr[\sigma \leftarrow K(1^k) : \mathcal{A}^{PR(\sigma,\cdot,\cdot,\cdot)}(\sigma) = 1] \stackrel{c}{\equiv} \Pr[(\sigma, \tau) \leftarrow \mathcal{S}_1(1^k) : \mathcal{A}^{SR^{L^k_w(\cdot)}(\sigma,\tau,\cdot,\cdot,\cdot)}(\sigma) = 1],$$

where $PR(\sigma, x, w, f)$ computes $r \leftarrow \{0,1\}^{\ell_P(k)}$; $\pi \leftarrow P(\sigma, x, w; r)$; $y = f(w\|r)$ and returns (π, y), while $SR^{L^k_\lambda(\cdot)w}(\sigma, \tau, x, w, f)$ computes $r \leftarrow \{0,1\}^{\ell_S(k)}$; $\pi \leftarrow \mathcal{S}_2(\sigma, \tau, x; r)$; $f' \leftarrow \mathcal{S}_3(\sigma, \tau, x, r, f)$; $y \leftarrow L^k_w(f')$ and returns (π, y). Here, the leakage query f' made to $L^k_w(\cdot)$ is such that its output length is no more than the output length of f. Both the oracles PR and SR output `fail` *if $(x, w) \notin \mathcal{R}$.*

3.1 Our Result

We now claim that every NIZK proof system with the *honest prover state reconstruction property*[5] is in fact a leakage-resilient NIZK. An immediate corollary is that the Groth et al. [36] NIZK proof system is a leakage-resilient NIZK proof system. We refer the reader to the full version for a formal proof.

Theorem 2. *A NIZK proof system (K, P, V) for a relation $\mathcal{R}$ with honest prover state reconstruction is a leakage resilient NIZK for $\mathcal{R}$.*

4 Applications of Leakage-resilient Zero Knowledge

4.1 UC with Leaky Tokens

Starting with the work of Goldreich and Ostrovsky on software protection [30], tamper-proof hardware tokens have been used for a variety of cryptographic tasks such as achieving universal composability [43, 14, 54, 15], one-time-programs [32], unconditionally secure protocols [35, 34], compilers for leakage-resilient computation [42, 33], etc. To the best of our knowledge, all prior works using tamper-proof hardware tokens make the assumption that the tokens are completely leakage-resilient (i.e., a token does not leak any information to an adversary who is in possession of the token). Here, we start a new line of research to investigate whether it is possible to relax this assumption for various cryptographic tasks. In particular, we study the feasibility of doing universally composable secure computation using "leaky" tokens. We start with the tamper-proof hardware token model of Katz [43] and modify it appropriately to incorporate "bounded"

[5] Very briefly, this property (also known as *non-erasure zero knowledge*) requires that not only can we simulate an honest party making a proof, but also *how it constructed the proof* (i.e., create convincing randomness so that it looks like the simulated proof was constructed by an honest prover using this randomness). See [36] for a formal definition.

leakage. Then, by making use of leakage-resilient hard relations [20] and our leakage-resilient zero knowledge proof system, we give a construction for a universally composable multi-party computation protocol in the leaky token model.

The Leaky Token Model. We first briefly recall the hardware token model of Katz [43]. In the model of [43], it is assumed that a party (referred to as the *creator*) can take some software code and "seal" it inside a tamper-proof hardware token; the party can then give this token to another party (referred to as the *user*), who can then access the embedded software in a black-box manner. This setup is modeled by a "wrapper" functionality $\mathcal{G}_{wrap}$ that accepts two types of messages: the first type is used by a party P to "create" a hardware token (encapsulating an interactive protocol M) and to "send" this token to another party P'. On receiving the token, P' can interact with it in an arbitrary black-box manner. This is formalized by allowing P' to send messages of its choice to M via $\mathcal{G}_{wrap}$. Each time M is invoked, fresh random coins are chosen for M.

In order to incorporate leakage attacks, we consider a modified wrapper functionality $\mathcal{G}^{\ell}_{wrap}$ parametrized by a leakage-parameter ℓ that defines the "total" leakage available to a token user over all the executions of the token. More concretely, the new wrapper functionality $\mathcal{G}^{\ell}_{wrap}$ is defined in the same manner as $\mathcal{G}_{wrap}$, except that $\mathcal{G}^{\ell}_{wrap}$ accepts special `leak` queries (from the token user) that consist of a efficiently computable function $f_i : \{0,1\}^* \rightarrow \{0,1\}^{\ell_i}$ (described as a circuit), to which the functionality answers with $f(M, state)$, where M denotes the code of the interactive Turing machine encapsulated in the token and *state* denotes the current state of M consisting of all the protocol messages received from the user and the random coins used so far by M in the current protocol execution. The token user can make any arbitrary polynomial number of such leakage queries over multiple protocol executions with M; we only require that the functions f_i be efficiently computable, and the total number of bits leaked (over all executions) is $\sum_i \ell_i = \ell$. We stress that by allowing leakage on M, we allow the token user to obtain leakage on any secret values hardwired into M. In the interest of space, we defer formal description of $\mathcal{G}^{\ell}_{wrap}$ to the full version.

UC Security via UC-Puzzles. In order to obtain our positive result, we build on the recent work of Lin et al. [49] which puts forward a unified framework for designing UC secure protocols from known setup assumptions [12, 13, 43, 6]. Lin et al. observe that a general technique for constructing UC secure protocols is to have the simulator obtain a "trapdoor string" which is hard to compute for the adversary. This is formalized in the form of (two party) "UC-puzzle" protocols that enable the simulator to obtain such a trapdoor string (but prevent the adversary from doing so). Following the work of [49], the task of constructing UC secure protocols from any setup assumption reduces to the task of constructing a UC-puzzle in the hybrid model of the corresponding setup.[6] We obtain our

[6] Very briefly, this is because Lin et al. show that a UC-puzzle can be used in conjunction with a strongly non-malleable witness indistinguishable protocol in order to construct a "concurrent simulation-sound" zero knowledge protocol with "UC

positive result by following the same route. Specifically, we construct a "family of UC-puzzles" in the $\mathcal{G}_{wrap}^{\ell}$-hybrid model.

Our Protocol. Recall that in the hardware token model, each pair of parties in the system exchange hardware tokens with each other. Now consider a system with m parties $P_1, \ldots, P_m$. For each pair of parties (P_i, P_j), we will construct two different UC-puzzles in the $\mathcal{G}_{wrap}^{\ell}$ hybrid model, (a) one where P_i (resp., P_j) acts as the puzzle sender (resp., receiver) and (b) the other where the roles of P_i and P_j are reversed. This gives us a family of m^2 UC-puzzles.

We now describe the construction of a family of protocol and relation pairs $(\langle S_{ij}, R_{ij}\rangle, \mathcal{R}_{ij})$, where $i, j \in [m]$. Here the choice of notation is to highlight that party P_i (resp., P_j) plays the role of the sender (resp., receiver) in protocol $\langle S_{ij}, R_{ij}\rangle$. We will then prove that each pair $(\langle S_{ij}, R_{ij}\rangle, \mathcal{R}_{ij})$ is a UC-puzzle in the $\mathcal{G}_{wrap}^{\ell}$-hybrid model. In our construction, we will use a $(1+\epsilon)$-LR-ZK proof of knowledge system[7] as well as an ℓ'-leakage-resilient hard relation [20], where $\ell' = (1+\epsilon)\cdot\ell$.

Description of $\langle S_{ij}, R_{ij}\rangle$. The interactive Turing machine S_{ij}, when invoked with the inputs the identity of the sender P_i, the identity of the receiver P_j and the session id *sid*, proceeds as follows. It first checks whether this is the first time interacting with party P_j. If so, it first samples a pair (x, y) from an ℓ'-leakage resilient hard relation $\mathcal{R}_{\ell'}$ and then "creates" and "gives" P_j a token (by sending the "appropriate" message to $\mathcal{G}_{wrap}^{\ell}$), which encapsulates the interactive Turing machine M that gives a λ-LR-ZKPOK of the statement that there exists an x such that $(x, y) \in \mathcal{R}_{\ell'}$. To actually challenge P_j, S_{ij} simply sends y as the puzzle to the receiver.

The interactive Turing machine R_{ij}, on receiving y from S_{ij}, engages in an execution of our λ-LR-ZKPOK protocol with M (via $\mathcal{G}_{wrap}^{\ell}$) where M proves that there exists an x such that $(x, y) \in \mathcal{R}_{\ell'}$. An adversarial receiver R_{ij} may additionally send leakage queries $(\texttt{leak}, f)$ to $\mathcal{G}_{wrap}^{\ell}$, who responds with $f(M\|r)$ (where r denotes the random coins used by M "so far") as long as the total leakage (over all queries) is bounded by ℓ.

Description of $\mathcal{R}_{ij}$. The puzzle relation $\mathcal{R}_{ij}$ is simply $\{(x, y)|(x, y) \in \mathcal{R}_{\ell'}\}$.

This completes the description of the pair $(\langle S_{ij}, R_{ij}\rangle, \mathcal{R}_{ij})$. We refer the reader to the full version for a proof of our claim that $(\langle S_{ij}, R_{ij}\rangle, \mathcal{R}_{ij})$ is a UC-puzzle in the $\mathcal{G}_{wrap}^{\ell}$-hybrid model.

4.2 Fully Leakage-Resilient Signatures

In this section, we give generic constructions of fully leakage-resilient (FLR) signature schemes in the bounded leakage model as well as the continual leakage

simulation" property, which in turn is known to be sufficient to construct UC secure protocols (see e.g. [13]). We refer the reader to the full version for more details.

[7] We note here that the LR-ZK proof system discussed in Section 2.1 is not a *proof of knowledge*. However, it is easy to modify the construction to obtain a proof of knowledge system by using a *leakage-sound* zero knowledge proof system. We refer the reader to the full version for more details.

model. In order to obtain our results, we will adapt the approach of Katz and Vaikuntnathan [44], and in particular, Dodis et al. [20, 19] (who used leakage-resilient hard relations and tag-based *true simulation-extractable* (tSE) NIZK argument systems to construct "standard" leakage-resilient signature schemes) to the setting of *full leakage* (where the adversary can leak on the entire state, as opposed to only the secret key). Specifically, we first extend our notion of leakage-resilient NIZKs to incorporate the property of true simulation-extractability. Then, by using a hard relation that is leakage-resilient in the bounded (resp., continual) leakage model along with our *true simulation-extractable leakage-resilient* (tSE-LR) NIZK argument system, we obtain FLR signatures in the bounded (resp., continual) leakage model. Somewhat interestingly, unlike the recent constructions of FLR signature schemes [51, 9], our constructions are also secure in the noisy leakage model [56]. In interest of space, here we limit our discussion to the construction of an FLR signature scheme in the bounded leakage model. We refer the reader to the full version for further discussion on the continual leakage model and the noisy leakage model.

True Simulation-Extractable Leakage-Resilient NIZK. We first (informally) define tag-based tSE-LR-NIZK argument system and give a construction for the same. Let us first recall the notion of tSE-NIZK, as defined in [20]. Very roughly, a NIZK proof system is true simulation extractable if there exists a PPT extractor which (when given an extraction trapdoor to the CRS) extracts a witness w^* from any proof π^* produced by an adversary $\mathcal{A}$ (using a tag $\mathtt{tag}^*$), even if $\mathcal{A}$ has previously seen some simulated proofs for other true statements (with different tags). Our notion of tSE-LR-NIZK extends the notion of tSE-NIZK by allowing the adversary to obtain (in addition to simulated proofs) leakage on the witness and randomness used to generate the simulated proofs.

A tag based tSE-LR-NIZK argument system $(\mathcal{K}, \mathcal{P}, \mathcal{V})$ follows directly from the adaptively secure UC-NIZK of Groth et al. [36]. The complete construction and proof is given in the full version.

Fully Leakage-Resilient Signatures in the Bounded Leakage Model. We will follow the definition of FLR signature schemes due to Boyle et al [9]. Very roughly, we say that a signature scheme is fully leakage-resilient in the bounded-leakage model if it is existentially unforgeable against any PPT adversary that can obtain polynomially many signatures over messages of her choice, as well as bounded leakage information on the secret key and the randomness used by the signing algorithm and the key generation algorithm) throughout the lifetime of the system. Due to space constraints, we omit the formal definition of FLR signatures from this manuscript. We now proceed to describe our construction. The security proof is deferred to the full version.

Our Construction. Let $\mathcal{R}_\ell$ be an ℓ-leakage-resilient hard relation with a PPT sampling algorithm $\text{KGEN}(\cdot)$. Let (K, P, V) be a tag-based tSE-LR-NIZK argument system for a relation $\mathcal{R}$. The signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is described as follows.

- $\mathsf{KeyGen}(1^k, \ell)$: Sample $(x, y) \leftarrow \text{KGEN}(1^k)$, $\sigma \leftarrow K(1^k)$. Output $sk = x$ and $pk = (\sigma, y)$.

– $\mathsf{Sign}_{sk}(m)$: Output $\Phi = \pi$, where $\pi \leftarrow P(\sigma, y, m, x)$. (Here m is the tag in the argument.)
– $\mathsf{Verify}_{pk}(m, \Phi)$: Output $V(\sigma, y, m, \Phi)$.

Theorem 3. *If $\mathcal{R}_\ell$ is an ℓ-leakage-resilient hard relation and (K, P, V) is a tag-based true simulation-extractable leakage-resilient NIZK argument system, then (KeyGen, Sign, Verify) is an ℓ-fully-leakage-resilient signature scheme in the bounded-leakage model.*

5 Conclusions

In this paper, we give definitions and constructions of leakage-resilient zero knowledge proof systems, where an adversarial verifier can obtain arbitrary bounded leakage on the secret state of the prover. It is natural to consider the (opposite) scenario where a malicious prover can obtain arbitrary leakage on the random coins of the verifier during the protocol execution. The question that we may ask is whether it is possible to construct interactive proofs that remain *sound* in such a scenario. Going even further, we can consider the question of constructing an interactive proof system that *simultaneously* satisfies the notions of "leakage-soundness" and leakage-resilient zero knowledge. In the full version, we give positive results for both these settings.

A natural question following our work is whether we can extend our notions and results to the setting of secure two-party computation. We address this in an upcoming work.

References

1. Ajtai, M.: Secure computation with information leaking to an adversary. In: STOC (2011)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: TCC (2009)
3. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: EUROCRYPT (2010)
4. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: CRYPTO (2009)
5. Anderson, R., Kuhn, M.: Tamper resistance: a cautionary note. In: WOEC (1996)
6. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS (2004)
7. Beaver, D.: Adaptive zero knowledge and computational equivocation. In: STOC (1996)
8. Bitansky, N., Canetti, R., Halevi, S.: Leakage tolerant interactive protocols. Cryptology ePrint Archive, Report 2011/204 (2011)
9. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: EUROCRYPT (2011)
10. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In: FOCS (2010)

11. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC (1996)
12. Canetti, R., Fischlin, M.: Universally composable commitments. In: CRYPTO (2001)
13. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC (2002)
14. Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: EUROCRYPT (2008)
15. Damgård, I., Nielsen, J.B., Wichs, D.: Universally composable multiparty computation with partially isolated parties. In: TCC (2009)
16. Damgård, I., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. J. Cryptology (1997)
17. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: CRYPTO (2001)
18. Dodis, Y., Goldwasser, S., Kalai, Y.T., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: TCC (2010)
19. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS (2010)
20. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: ASIACRYPT (2010)
21. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: STOC (2009)
22. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. (2000)
23. Dwork, C., Naor, M., Sahai, A.: Concurrent zero knowledge. In: STOC (1998)
24. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS (2008)
25. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: TCC (2010)
26. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from leakage: the computationally-bounded and noisy cases. In: EUROCRYPT (2010)
27. Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: CRYPTO. pp. 526–545 (1989)
28. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: CHES (2001)
29. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptology (1996)
30. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM (1996)
31. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: STOC (1985)
32. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. In: CRYPTO (2008)
33. Goldwasser, S., Rothblum, G.N.: Securing computation against continuous leakage. In: CRYPTO (2010)
34. Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In: CRYPTO (2010)
35. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding cryptography on tamper-proof hardware tokens. In: TCC (2010)
36. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: EUROCRYPT (2006)

37. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: USENIX Security Symposium (2008)
38. Halevi, S., Micali, S.: Practical and provably-secure commitment schemes from collision-free hashing. In: CRYPTO (1996)
39. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Extracting correlations. In: FOCS (2009)
40. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits ii: Keeping secrets in tamperable circuits. In: EUROCRYPT (2006)
41. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: CRYPTO (2003)
42. Juma, A., Vahlis, Y.: Protecting cryptographic keys against continual leakage. In: CRYPTO (2010)
43. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: EUROCRYPT (2007)
44. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: ASIACRYPT (2009)
45. Kiltz, E., Pietrzak, K.: Leakage resilient elgamal encryption. In: ASIACRYPT (2010)
46. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: CRYPTO (1996)
47. Lewko, A., Lewko, M., Waters, B.: How to leak on key updates. In: STOC (2011)
48. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: TCC (2011)
49. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: STOC (2009)
50. Lindell, Y., Zarosim, H.: Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. In: TCC (2009)
51. Malkin, T., Teranishi, I., Vahlis, Y., Yung, M.: Signatures resilient to continual leakage on memory and computation. In: EUROCRYPT (2011)
52. Micali, S., Pass, R.: Local zero knowledge. In: STOC (2006)
53. Micali, S., Reyzin, L.: Physically observable cryptography. In: TCC (2004)
54. Moran, T., Segev, G.: David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: EUROCRYPT (2008)
55. Naor, M.: Bit commitment using pseudo-randomness (extended abstract). In: CRYPTO (1989)
56. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: CRYPTO (2009)
57. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: STOC (1989)
58. Pass, R., Rosen, A.: New and improved constructions of non-malleable cryptographic protocols. In: STOC (2005)
59. Pietrzak, K.: A leakage-resilient mode of operation. In: EUROCRYPT (2009)
60. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS (2002)
61. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: E-smart (2001)
62. Rosen, A.: A note on constant-round zero-knowledge proofs for NP. In: TCC (2004)
63. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS (1999)