Parallel Repetition for Leakage Resilience Amplification Revisited

Abhishek Jain[1] and Krzysztof Pietrzak[2]

[1] UCLA
abhishek@cs.ucla.edu
[2] CWI, Amsterdam
pietrzak@cwi.nl

Abstract. If a cryptographic primitive remains secure even if ℓ bits about the secret key are leaked to the adversary, one would expect that at least one of n independent instantiations of the scheme remains secure given $n \cdot \ell$ bits of leakage. This intuition has been proven true for schemes satisfying some special information-theoretic properties by Alwen et al. [Eurocrypt'10]. On the negative side, Lewko and Waters [FOCS'10] construct a CPA secure public-key encryption scheme for which this intuition fails.

The counterexample of Lewko and Waters leaves open the interesting possibility that for any scheme there exists a constant $c > 0$, such that n fold repetition remains secure against $c \cdot n \cdot \ell$ bits of leakage. Furthermore, their counterexample requires the n copies of the encryption scheme to share a common reference parameter, leaving open the possibility that the intuition is true for all schemes without common setup.

In this work we give a stronger counterexample ruling out these possibilities. We construct a signature scheme such that:

1. a single instantiation remains secure given $\ell = \log(k)$ bits of leakage where k is a security parameter.
2. any polynomial number of independent instantiations can be broken (in the strongest sense of key-recovery) given $\ell' = \text{poly}(k)$ bits of leakage. Note that ℓ' does not depend on the number of instances.

The computational assumption underlying our counterexample is that non-interactive computationally sound proofs exist. Moreover, under a stronger (non-standard) assumption about such proofs, our counterexample does not require a common reference parameter.

The underlying idea of our counterexample is rather generic and can be applied to other primitives like encryption schemes.

1 Introduction

In a cryptographic security definition one must precisely specify in which way an anticipated adversary can access the scheme. Classical security definitions usually give the adversary only black-box access, that is she can observe the input/output behavior of the scheme, but nothing else is leaked.

Y. Ishai (Ed.): TCC 2011, LNCS 6597, pp. 58–69, 2011.

Unfortunately, in the last two decades, it has become evident that such notions are often insufficient to guarantee security in the real world where the physical implementation (e.g. on a smart-card) of a scheme is under attack. Two important types on attacks which are not captured by classical security notions are side-channel attacks and malware.

Side-Channel Attacks. Cryptanalytic attacks where the adversary exploits information leakage from the physical implementation of a scheme are called side-channel attacks. Important examples of side-channels that have been exploited include measuring the running-time [24,29], electromagnetic radiation [32,17] or power consumption [25] of a cryptodevice. Cold-boot attacks [19] exploit the fact that memory retains its content for several seconds or even minutes even after being removed from a laptop, allowing the adversary to learn (a noisy version of) the content of the memory. In a probing attack [4], one measures the contents carried by some wires of the circuit.

Malware. Today most computers (and even simpler devices) are connected to the Internet, where they are constantly exposed to attacks by malicious softwares like viruses and Trojans. Even with anti-virus protection in place, it is quite inevitable that a computer is from time to time infected by malware. Such attacks are particularly devastating if cryptographic keys are stored on the computer, as the malware can send them out to the bad guys.

Notions for Key-Leakage. Traditional security notions only consider adversaries having black-box access to the primitive at hand, and thus do not capture side-channel or malware attacks at all. The common approach to protect against side-channel attacks (similarly for malware) is ad-hoc, in the sense that one has to devise a countermeasure against all the known attacks. A more recent approach is to model security against "general" side-channel (or malware) attacks at the definitional level.

Memory-Attacks. A basic such notion is called "security against memory-attacks". A cryptographic scheme is secure against memory attacks, if it remains secure even if a bounded amount of information about the secret key is given to the adversary. In this model, [1,28,7] construct public-key encryption schemes and [22,3] construct signature schemes, identification schemes and key exchange protocols.

Formally, memory attacks are modeled by giving the adversary – on top of the normal black-box access – the power to choose any efficient leakage function f with bounded range ℓ bits; she then gets $f(sk)$ where sk denotes the secret key of the scheme at hand. Of course ℓ must be significantly smaller than $|sk|$ since otherwise the adversary can just leak the entire key.

Although in a memory attack the adversary can learn arbitrary leakage, she is limited to learn a total of ℓ bits. This is not sufficient to protect against most side-channel attacks or malware, where we need stronger models. One way is to consider "continuous" leakage, which requires frequent key-updates, or making the key huge while preserving efficiency of the scheme.

Continuous Leakage Models. Typical side-channel attacks leak small amounts of information per invocation, but since a scheme can typically be invoked many times, no upper bound on the total leakage should be assumed. The notion of "leakage-resilient cryptography" [15,30,16,33,10,18,21] and the recently introduced model of "continuous memory attacks" [12,8] capture such attacks, allowing the adversary to learn a bounded amount of leakage (computed by adaptively chosen leakage functions) *with every invocation.* In the model of leakage-resilience one needs the additional assumption that during each invocation, only the part of the memory that is actually accessed leaks. Continuous memory attacks require (almost) leakage free update phases.

The Bounded Retrieval Model. The bounded-retrieval model (BRM) [13,11] propose a solution to the leakage of cryptographic keys from devices which can get infected by malware as explained above. The idea is to design schemes where the key can be made huge (2GB, say), and the scheme remains secure even if a large amount (1GB, say) of arbitrary information of the key is leaked. The key is protected, even if the computer is infected by malware which can perform arbitrary computation on the infected computer, but can only send out at most 1GB worth of data, either due to low bandwidth or because larger leakage can be detected. This notion is strictly stronger than security against memory attacks, as here one additionally requires that the efficiency of the scheme is basically independent of the size of the secret key. In the BRM model, symmetric authentication schemes [13,11,9], password authentication [11] and secret-sharing [14] were constructed. Recently the first *public-key* primitives in the BRM model were constructed by Alwen at al.[3,2].[1]

Security Amplification by Repetition. Given a scheme that withstands memory attacks with, say $\ell = |sk|/2$ leakage, we can construct a scheme that withstands 1GB of leakage by using a huge key $|sk|$ =2GB. This would *not* be considered a solution in the BRM model, as this model requires the efficiency of the scheme to depend only on some security parameter k, but not (or only logarithmically) on the key length (In particular, schemes in the BRM model cannot even read the entire key on a single invocation.)

The approach taken by Alwen et al. [3,2] is to use parallel repetition. They start with a scheme which can tolerate ℓ bits of leakage, and run this scheme n times in parallel. For a signature scheme this means to sample n secret keys $sk_1, \ldots, sk_n$, a signature for the parallel scheme would then consist of n signatures, using the respective keys.[2]

[1] The BRM predates the notion of memory-attacks, but public-key cryptography in the BRM came only after it was constructed in the model of memory attacks.

[2] This is still not really a scheme in the BRM model as efficiency of the scheme is linear in n. To actually get signatures, [3] only use a cleverly chosen subset of the keys for signing, and also must "compress" the n public keys in order to make their size independent of n.

The hope is that if the signature scheme is secure against ℓ bits of leakage, the parallel scheme will remain secure given $n \cdot \ell$ bits of leakage. Alwen et al. prove that for their particular scheme this is indeed the case, but their proof exploits information theoretic properties of the scheme, and cannot be generalized to work for any scheme.

The Counterexample of Lewko and Waters. Recently, Lewko and Waters [26] showed that in fact, n-fold parallel repetition does not, in general, amplify leakage resilience from ℓ to $n \cdot \ell$ bits. They construct a scheme (their main example is encryption, but it is outlined how to adapt the counterexample to signatures.) where a single instance is secure given ℓ bits of leakage, but can be broken given $c \cdot n \cdot \ell$ bits of leakage for some constant $c < 1$. Their attack also requires a common setup, in that all the instances of the basic scheme in the parallel system must be over the same group, and the group order must be secret.[3]

Our Results. The counterexample of Lewko and Waters leaves open the possibility that for every scheme, there exists a constant $c > 0$ such that n-fold parallel repetition (even with common setup) amplifies leakage-resilience from ℓ to $c \cdot n \cdot \ell$ bits.

Moreover the common setup seems crucial for their counterexample, and it leaves open the question whether n-fold parallel repetition without common setup amplifies leakage-resilience from ℓ to $n \cdot \ell$ bits for all schemes.

We give a new counterexample that answers both questions in the negative (albeit the 2nd only under a non-standard assumption, details are given below.) More concretely, from any secure signature scheme, we construct a new signature scheme such that:

1. a single instantiation of the scheme remains secure given $\ell = \log(k)$ bits of leakage where k is a security parameter.
2. n *independent* instances (where n can be any polynomial in k) of the scheme can be broken (in the strongest sense of key-recovery) given $\ell' = \text{poly}(k)$ bits of leakage.

Note that ℓ' – the leakage needed to break n instances – does not even depend on n.

Our techniques are quite general and we anticipate that they can be extended to construct counter-examples for other primitives as well. Besides the counterexample for signature schemes just mentioned (which also works for one-time signatures), we provide a similar counterexample for CCA-secure encryption schemes.

We note here that the results of [26] are applicable to even 1-bit (CPA-secure) encryption schemes and signature schemes with "random message unforgeability under no-message attack", while our techniques do not seem to lend themselves to such settings.

[3] The constant c depends on the underlying group, and can be made arbitrary small, but once the group is fixed, so is c.

The main assumption underlying our results is the existence of non-interactive CS proofs (see below). We note that in contrast, the counterexample of [26] is based on a specific-number theoretic, but falsifiable assumption.

Our Techniques. Our negative results make crucial use of computationally sound (CS) proofs as constructed by Micali [27] (using techniques from [23]). Specifically, our counterexample relies on the existence of *non-interactive* CS proofs for NP languages. The usefulness of non-interactive CS proofs lies in the fact that they are extremely short; in particular, their length is only polylogarithmic in the size of the statement and its witness.

Proof Idea. We construct our counterexample by starting with any signature scheme, and extending it as follows. To every secret key we add some random string w, and to the public-key we add the value $G(w)$ where $G(.)$ is a (sufficiently expanding) pseudorandom generator.

The signing algorithm is basically unchanged, except that it additionally checks if the message to be signed contains a CS proof for the fact that $G(w)$ is indeed the the range of $G(.)$. If this is the case, it does something stupid, namely outputting its entire secret key as part of the signature.

The security of the underlying signature scheme and the CS proof system implies that this altered scheme is secure against $\ell = \log(k)$ bit of leakage. On the other hand, a leakage function which has access to n secret keys, can output a short (i.e. of length which is independent of n) CS proof showing that the string $G(w_1), \ldots, G(w_n)$ is indeed the concatenation of n strings in the range of $G(.)$. This proof can then be used (making signing queries) to extract the entire secret key of all the n instances of the scheme.

About the common reference parameter. Non-interactive CS proofs have been shown to exist in the random oracle model (which in practice must be instantiated with an efficient hash-function.) Since a random oracle implies a common reference string, it seems that we haven't improved upon the counterexample of Lewko and Waters [26] as far as the usage of common setup is concerned. Recall that the common setup in [26] contains a number N whose factorisation must remain secret since otherwise the scheme can be trivially broken. In our case, the common setup contains a hash function $h(.)$ (replacing the random oracle). Although for every $h(.)$, there exist CS proofs for invalid statements, this may not necessarily be a problem for us, as all we need is that it is hard to come up with a proof for a *random* word not in the language (the language and exact distribution depends on the PRG we use in our construction.) It is conceivable that for some concrete choices of a PRG and hash function repalcing the random oracle (SHA256, say), it is hard to come up with such invalid proofs even given a polynomial amount of auxiliary information (containing e.g. many invalid proofs.) If we make such a (non-standard) assumption, our counterexample does not need any common setup.

Related Work. We are aware of at least two works where proof systems with short proofs are used to construct counterexamples. Closest to ours is the work of "seed-incompressible functions" of Halevi, Myers and Rackoff [20], who use CS proofs to show that no pseudorandom function exists which remains secure after one leaks a “compressed” key. Another example is the work on parallel repetition of computationally sound proofs [31] (based on [6]), which uses a different kind of proof systems, universal arguments [5], to show that parallel repetition of computationally sound protocols with eight (or more) rounds does in general not reduce the soundness error of the protocol.

2 Preliminaries

2.1 Leakage-Resilient Signatures

Our definition of leakage resilient signatures is essentially the standard notion of existentially unforgeability under adaptive chosen message attacks, except that we allow the adversary to specify an arbitrary function $f(\cdot)$ (whose output length is bounded by the leakage parameter), and obtain the value of f applied to the signing key. Let k be the security parameter, and (KeyGen, Sign, Verify) denote a signature scheme. Let ℓ denote the leakage parameter. In order to define leakage resilient signatures, we consider the following experiment.

1. Compute $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k, \ell)$ and give pk to the adversary.
2. Run the adversary $\mathcal{A}(1^k, pk, \ell)$. The adversary may make adaptive queries to the signing oracle $\textsc{Sign}_{sk}(\cdot)$ and the leakage oracle $\textsc{Leak}_{sk}(\cdot)$, defined as follows:
   - On receiving the i^{th} query m_i, $\textsc{Sign}_{sk}(m_i)$ computes $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$ and outputs σ_i.
   - On receiving an input f (where f is a polynomial-time computable function, described as a circuit), $\textsc{Leak}_{sk}(f)$ outputs $f(sk)$ to $\mathcal{A}$. The adversary is allowed only a single query to the leakage oracle. It can choose any function $f(\cdot)$ of the form $f : \{0,1\}^* \rightarrow \{0,1\}^{\ell}$.
3. At some point, $\mathcal{A}$ stops and outputs (m, σ).

We say that $\mathcal{A}$ *succeeds* if (a) $\mathsf{Verify}(pk, \sigma) = 1$, and (b) m was never queried to $\textsc{Sign}_{sk}(\cdot)$.

Definition 1. *A signature scheme* (KeyGen, Sign, Verify) *is ℓ-leakage resilient if all polynomial-time adversaries $\mathcal{A}$ can succeed with only negligible probability in the above experiment.*

2.2 CS Proofs

Our negative result makes crucial use of CS proofs as constructed by Micali [27] (using techniques from [23]). Below, we recall the definition of non-interactive CS proofs. The definition below is taken almost verbatim from [20].

Definition 2 (Non-interactive CS proofs). *A non-interactive CS-proof system for a language $L \in \mathcal{NP}$ (with relation R_L), consists of two deterministic polynomial-time machines, a prover P and a verifier V, operating as follows:*

- *On input $(1^k, x, w)$ such that $(x, w) \in R_L$, the prover computes a proof $\pi = P(x, w)$ such that $|\pi| \leq poly(k, \log(|x| + |w|))$.*
- *On input $(1^k, x, \pi)$, the verifier decides whether to accept or reject the proof π (i.e., $V(x, \pi) \in \{\texttt{accept}, \texttt{reject}\}$).*

The proof system satisfies the following conditions, where the probabilities are taken over the random coins of the prover and the verifier:

Perfect completeness: *For any $(x, w) \in R_L$ and for any k,*

$$\Pr[\pi \leftarrow P(x, w), V(x, \pi) = \texttt{accept}] = 1$$

Computational soundness: *For any polynomial time machine $\tilde{P}$ and any input $x \notin L$, it holds that*

$$\Pr[\pi \leftarrow \tilde{P}(x), V(x, \pi) = \texttt{accept}] \leq \mathrm{negl}(k)$$

3 A Leakage Resilient Signature Scheme

Let k be the security parameter. Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be any λ-leakage resilient signature scheme where λ is at least logarithmic in the security parameter. Let $G : \{0,1\}^k \rightarrow \{0,1\}^{2k}$ be a pseudo-random generator (PRG). Let L be an $\mathcal{NP}$ language such that a string $y \in L$ iff $y = y_1, y_2, \ldots$ (where $|y_i| = 2k$) and $\forall i$, $\exists w_i$ such that $G(w_i) = y_i$. Let $\langle P, V \rangle$ be a non-interactive CS-proof system for the language L, where both prover and verifier are PPT machines. We now describe a new ℓ-leakage resilient signature scheme (KeyGen, Sign, Verify), where $\ell = \log(k)$.

KeyGen$(1^k, \ell)$: Compute $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k, \lambda)$. Choose random $w \in \{0,1\}^k$ and compute $y = G(w)$. The public key is $PK = (pk, y)$ and the secret key is $SK = (sk, w)$.

Sign(SK, m): To sign message $m = m_1, m_2$ using secret key $SK = (sk, w)$, first parse m_1 as $m_1^1, m_1^2, \ldots$ such that $|m_1^i| = 2k$. If $\exists i$ such that $m_1^i = y$ $(= G(w))$, then run $V(m_1, m_2)$. If it returns $\texttt{accept}$, then output SK. Otherwise, output $\sigma \leftarrow \mathsf{Sign}(sk, m)$ as the signature.

Verify(PK, m, σ): Given a signature σ on message m with respect to the public key $PK = (pk, y)$, output 1 iff $\mathsf{Verify}(pk, m, \sigma) = 1$.

This completes the description of our signature scheme. We now state our main results.

Theorem 1. *The proposed signature scheme (*KeyGen*,* Sign*,* Verify*) is ℓ-leakage resilient, where $\ell = \log(k)$.*

Theorem 2. *There exists a fixed polynomial $q(\cdot)$ such that any n-fold repetition (where n can be any polynomial in k) of the signature scheme (*KeyGen*,* Sign*,* Verify*) can be broken by a key-recovery attack with $\ell' = q(k)$ bits of leakage.*

Remark. Note that in Theorem 2, $\ell' = q(k)$ only depends on the security parameter, but does not depend on the number of repetitions n.

We prove Theorem 1 in the next subsection. Theorem 2 is proven in Section 4.

3.1 Leakage Resilience of our Signature Scheme

We will prove theorem 1 by contradiction. Specifically, we will show that given an adversary $\mathcal{A}$ that forges signatures for (KeyGen, Sign, Verify) with non-negligible probability δ, we can construct an adversary $\mathcal{B}$ that forges signatures for $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ with probability $\delta' = \frac{\delta - \text{negl}(k)}{k}$.

Description of $\mathcal{B}$. Let C denote the challenger for the signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. At a high level, $\mathcal{B}$ works by internally running the adversary $\mathcal{A}$; $\mathcal{B}$ answers $\mathcal{A}$'s queries by using the responses (to its own queries) from C, and then outputs the signature forgery created by $\mathcal{A}$. We now give more details.

On receiving a public key pk from C, $\mathcal{B}$ chooses a random $w \in \{0,1\}^k$ and computes $y = G(w)$. It then sends $PK = (pk, y)$ as the public key to $\mathcal{A}$. Now, when $\mathcal{A}$ makes a signature query $m = m_1, m_2$, $\mathcal{B}$ first parses m_1 as $m_1^1, m_1^2, \ldots$ such that $|m_1^i| = 2k$. If $\exists i$ such that $m_1^i = y$, then $\mathcal{B}$ runs $V(m_1, m_2)$. If it returns `accept`, then $\mathcal{B}$ outputs the abort symbol $\perp$ and stops. Otherwise, it obtains a signature σ on message m from C and sends σ to $\mathcal{A}$. Further, when $\mathcal{A}$ makes a leakage query f, $\mathcal{B}$ simply guesses $y = f(sk, w)$ (where sk is the secret key corresponding to pk) and sends y to $\mathcal{A}$. Finally, when $\mathcal{A}$ creates a forgery (m^*, σ^*), $\mathcal{B}$ outputs (m^*, σ^*) and stops. This completes the description of $\mathcal{B}$.

Let us now analyze the interaction between $\mathcal{B}$ and $\mathcal{A}$. First note that since the leakage query function f has a bounded output length $\ell = \log(k)$, $\mathcal{B}$'s response to $\mathcal{A}$'s leakage query is correct with probability $\epsilon = \frac{1}{k}$. Now assuming that $\mathcal{B}$ outputs the abort symbol during its interaction with $\mathcal{A}$ with only negligible probability, we can establish that $\mathcal{B}$ outputs a valid forgery with probability $\delta' = \frac{\delta - \text{negl}(k)}{k}$ which proves our hypothesis. Therefore, in order to complete the proof, we only need to argue that $\mathcal{B}$ outputs the abort symbol with negligible probability.

Lemma 1. *$\mathcal{B}$ outputs the abort symbol $\perp$ with probability* $\text{negl}(k)$.

We prove lemma 1 by a simple hybrid experiment. Consider two hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$, described as follows.

$\mathcal{H}_0$**:** This is the real experiment between $\mathcal{B}$ and $\mathcal{A}$. Here $\mathcal{B}$ uses a pseudo-random generator G to compute y as part of the public key PK.

$\mathcal{H}_1$**:** Same as $\mathcal{H}_0$, except that $\mathcal{B}$ chooses $y \in \{0,1\}^{2q}$ uniformly at random.

Recall that $\mathcal{B}$ outputs the abort symbol only when $\mathcal{A}$ sends a signature query $m = m_1, m_2$ where $m_1 = m_1^1, m_1^2, \ldots$ ($|m_1^i| = 2q$) and $\exists i$ such that $m_1^i = y$ and $V(m_1, m_2)$ outputs `accept` (i.e., if m contains a non-interactive CS proof that "explains" y). We will denote such a query as a `bad` query. Let p_0 (resp., p_1) be the probability that $\mathcal{A}$ makes a `bad` query in hybrid $\mathcal{H}_0$ (resp., $\mathcal{H}_1$). Then, from the pseudo-randomness property of G, it follows that:

$$p_0 - p_1 \leq \mathrm{negl}(k) \tag{1}$$

Now, note that since y is chosen uniformly at random in hybrid $\mathcal{H}_1$, $\mathcal{A}$ can make a `bad` query in $\mathcal{H}_1$ only if it can create a false non-interactive CS proof. Then, from the soundness of the CS proof system $\langle P, V \rangle$, it follows that the probability that $\mathcal{A}$ makes a `bad` query in $\mathcal{H}_1$ is negligible, i.e., $p_1 = \mathrm{negl}(k)$. Combining this with equation 1, we establish that $p_0 = \mathrm{negl}(k)$. This completes the proof of lemma 1.

4 Attack on Parallel System

Recall from Definition 2 that the length of a non-interactive CS proof is $q'(k, \log(|x|+|w|))$ for some polynomial $q'(.,.)$, where x is the instance and w is the witness for x. Now, note that for any such pair (x, w), if $|x|$ and $|w|$ are polynomial in k, then for sufficiently large k, we have that $|x|+|w| \leq k^{\log(k)}$. Therefore, there exists a fixed polynomial $q(\cdot) \stackrel{def}{=} q'(., \log(k^{\log k}))$ such that for any non-interactive CS proof $\pi = P(x, w)$, we have that $|\pi| \leq q(k)$ for sufficiently large k.

Now consider a parallel system ($\overline{\text{KeyGen}}$, $\overline{\text{Sign}}$, $\overline{\text{Verify}}$) defined as an n-fold repetition of (KeyGen, Sign, Verify), where $n = p(k)$ for any polynomial $p(\cdot)$. $\overline{\text{KeyGen}}$ runs KeyGen n times to generate n key pairs $(PK_1, SK_1), \ldots, (PK_n, SK_n)$. To sign a message m, $\overline{\text{Sign}}$ computes $\sigma_i \leftarrow \text{Sign}(SK_i, m)$ for each $i \in [n]$ and outputs $\sigma = (\sigma_1, \ldots, \sigma_n)$ as the signature. Finally, on input a signature $\sigma = (\sigma_1, \ldots, \sigma_n)$, $\overline{\text{Verify}}$ outputs 1 iff $\forall i$, $\text{Verify}(PK_i, m, \sigma_i) = 1$.

We now describe an adversary $\mathcal{A}$ that can mount a key-recovery attack on ($\overline{\text{KeyGen}}$, $\overline{\text{Sign}}$, $\overline{\text{Verify}}$) given $q(k)$ bits of leakage. The adversary $\mathcal{A}$ receives $(PK_1, \ldots, PK_n)$ from the challenger of the signature scheme. Recall that $\forall i$, key pairs (PK_i, SK_i) are of the form: $PK_i = (pk_i, y_i)$, $SK_i = (sk_i, w_i)$. Let $y = y_1, \ldots, y_n$ and π be a non-interactive CS-proof to prove the membership of y in L. Then, $\mathcal{A}$ makes a leakage query with function f such that $f(SK_1, \ldots, SK_n) = \pi$. As discussed above, it holds that $|\pi| \leq q(k)$. $\mathcal{A}$ now queries the challenger for a signature on the message $m = m_1, m_2$ where $m_1 = y_1, \ldots, y_n$ and $m_2 = \pi$. At this point, $\mathcal{A}$ must receive $SK_1, \ldots, SK_n$ in response since π is a valid proof for the statement $y \in L$.

This completes the description of the attack on the parallel system.

5 Extending Our Techniques to Other Primitives

The techniques used in section 3 are rather general and not specific to signature schemes. In particular, they can be used to construct other leakage resilient crypto primitives that are insecure under parallel repetition. As an example, in this section, we briefly explain how to construct a leakage resilient CCA-secure public-key encryption scheme that is insecure under parallel repetition.

Let (KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$) be a CCA-secure public-key encryption scheme that can tolerate at least logarithmic amount of leakage (here the adversary has to choose the leakage function before getting the challange ciphertext). Let G be a length doubling PRG and $\langle P, V \rangle$ be a non-interactive CS-proof system for the language L, as described in section 3. We now describe a new ℓ-leakage resilient CCA-secure public-key encryption scheme (KeyGen, Encrypt, Decrypt), where ℓ is logarithmic in the security parameter.

KeyGen$(1^k, \ell)$: Compute $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$. Choose random $w \in \{0,1\}^k$ and compute $y = G(w)$. The public key is $PK = (pk, y)$ and the secret key is $SK = (sk, w)$.

Encrypt(PK, m): To encrypt a message m using public key $PK = (pk, y)$, simply compute $c \leftarrow \mathsf{Encrypt}(pk, m)$.

Decrypt(SK, c): Given a ciphertext $c = c_1, c_2$ and secret key $SK = (sk, w)$, first parse c_1 as $c_1^1, c_1^2, \ldots$ such that $|c_1^i| = 2k$. If $\exists i$ such that $c_1^i = G(w)$, then run $V(c_1, c_2)$. If it returns `accept`, then output SK. Otherwise, output $m \leftarrow \mathsf{Decrypt}(sk, c)$ as the decrypted message.

It is not difficult to see that the new scheme is ℓ-leakage resilient, where $\ell = \log(k)$. Essentially, we can leverage the soundness of CS proof system and the fact that G is a PRG (in the same manner as in the security proof of the signature scheme described in section 3) to reduce the security of (KeyGen, Encrypt, Decrypt) to that of the underlying scheme (KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$).

Now, consider an n-fold repetition of the above scheme. Let $PK_i = (pk_i, y_i)$ denote the public key of the i^{th} instance of the above scheme. A natural way to encrypt a message m in the parallel system is to split m into n shares and encrypt each share m_i with PK_i. Now, (as in the proof of Theorem 2) an adversary $\mathcal{A}$ can make a leakage query to obtain a short CS proof π that explains each y_i. Then, since $\mathcal{A}$ has access to a decryption oracle, it can now send a decryption query $c = c_1, c_2$ where $c_1 = y_1, \ldots, y_n$ and $c_2 = \pi$. $\mathcal{A}$ can therefore successfully recover the secret key of the parallel system.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)

4. Anderson, R., Kuhn, M.: Tamper resistance: a cautionary note. In: WOEC 1996: Proceedings of the 2nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce, pp. 1–11. USENIX Association, Berkeley (1996)
5. Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM J. Comput. 38(5), 1661–1694 (2008)
6. Bellare, M., Impagliazzo, R., Naor, M.: Does parallel repetition lower the error in computationally sound protocols? In: 38th Annual Symposium on Foundations of Computer Science, pp. 374–383. IEEE Computer Society Press, Los Alamitos (October 1997)
7. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
8. 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)
9. Cash, D., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 479–498. Springer, Heidelberg (2007)
10. Chow, S.S., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions. In: ACM CCS 2010: 17th Conference on Computer and Communications Security. ACM Press, New York (2010)
11. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)
12. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS (2010)
13. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
14. Dziembowski, S., Pietrzak, K.: Intrusion-resilient secret sharing. In: 48th Annual Symposium on Foundations of Computer Science, pp. 227–237. IEEE Computer Society Press, Los Alamitos (October 2007)
15. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th Annual Symposium on Foundations of Computer Science, pp. 293–302. IEEE Computer Society Press, Los Alamitos (October 2008)
16. Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010)
17. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
18. Goldwasser, S., Rothblum, G.N.: Securing computation against continuous leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010)
19. 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, pp. 45–60 (2008)
20. Halevi, S., Myers, S., Rackoff, C.: On seed-incompressible functions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 19–36. Springer, Heidelberg (2008)

21. Juma, A., Vahlis, Y.: Protecting cryptographic keys against continual leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
22. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
23. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proc. 24th STOC, pp. 723–732 (1992)
24. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
25. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
26. Lewko, A., Waters, B.: On the insecurity of parallel repetition for leakage resilience. In: FOCS (2010)
27. Micali, S.: A secure and efficient digital signature algorithm. Technical Memo MIT/LCS/TM-501b, Massachusetts Institute of Technology, Laboratory for Computer Science (April 1994)
28. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
29. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006)
30. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
31. Pietrzak, K., Wikström, D.: Parallel repetition of computationally sound protocols revisited. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 86–102. Springer, Heidelberg (2007)
32. Quisquater, J.J., Samyde, D.: ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
33. Yu, Y., Standaert, F.X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: ACM CCS 2010: 17th Conference on Computer and Communications Security. ACM Press, New York (2010)