Signatures Resilient to Continual Leakage on Memory and Computation

Tal Malkin[*] Isamu Teranishi[†] Yevgeniy Vahlis[*] Moti Yung[‡]

Abstract

Recent breakthrough results by Brakerski *et al* and Dodis *et al* have shown that signature schemes can be made secure even if the adversary *continually* obtains information leakage from the secret key of the scheme. However, the schemes currently do not allow leakage on the secret key and randomness *during signing*, except in the random oracle model. Further, the random oracle based schemes require updates to the secret key in order to maintain security, even when no leakage during computation is present.

We present the first signature scheme that is resilient to full continual leakage: memory leakage as well as leakage from processing during signing (both from the secret key and the randomness), in key generation, and in update. Our scheme can tolerate leakage of a $1 - o(1)$ fraction of the secret key between updates, and is proven secure in the standard model based on the symmetric external DDH (SXDH) assumption in bilinear groups. The time periods between updates are a function of the amount of leakage in the period (and nothing more).

Our construction makes new use of the Groth-Sahai proof systems, and in particular avoids composing proofs, which gives improved efficiency. In addition, we introduce a new tool: independent pre-image resistant hash functions, which may be of independent interest.

[*]Columbia University. {tal,evahlis}@cs.columbia.edu

[†]NEC Japan, and Columbia University. teranisi@ah.jp.nec.com

[‡]Google Inc., and Columbia University. moti@cs.columbia.edu

Contents

1 Introduction

Cryptographic schemes have traditionally been modeled under the assumption that the secret key is hidden completely within a device and attacks are of "black box" nature. However, cryptographic engineering has taught us that devices are not perfect and can leak information about the key, primarily via what is known as "side channels." These channels give the adversary some partial information by observing physical leakage correlated with the secret key, such as timing or radiation or power consumption associated with computations (either directly by probing the circuit or via antennas probing unshielded devices), as well as memory leakage that reveals information about the secret key (cf., [BB05, QS01, KJJ99, HSH+08].)

The threat of partial leakage of keys has not escaped cryptographers; perhaps the earliest works on the subject are Shamir's secret sharing [S79], and later Rivest's all or nothing transform [R97]. In the last few years a large body of work on leakage-resilient cryptography has emerged (cf., [ISW03, MR04, SMY09, AGV09, DKL09, P09, NS09, ADW09, KV09, FKPR10, DGK+10, FRR+10, BG10, DP10, JV10, GR10, DHLW10, BKKV10]), which provide different leakage models requiring that cryptographic schemes be secure even if partial information about the secret key is leaked to an adversary. In order to have an abstract model that is not directly dependent upon the hardware architecture itself, but rather can produce some general guidelines to systems and hardware designers regarding how much leakage can be tolerated overall (within a given setting), leakage was formally modeled as functions associated with algorithmic steps and states. These leakage functions $f_1(sk), \ldots, f_q(sk)$ of the secret key sk, are from some given family of functions, where the specific f_i is selected by the adversary arbitrarily. The leakage (i.e., the function result which contains only partial information about the key) is applied to the relevant state or computation and is given to the adversary (in addition to the black-box access it gets).

Wiki-Leakage: The variety of Leakage Models. There are many types of leakage sub-models based on various parameters of leakage, which we briefly review here. Adaptive leakage, where the function is chosen dynamically by the adversary, obviously gives it more power than non-adaptive. Such adversaries that rather than observing the device once execute side channel attacks repetitively, are stronger and require the schemes to be *continuous leakage resilient*. Repeating (continual) adaptive leakage requires the scheme to have an update procedure for the secret key. Otherwise, the adversary can eventually leak enough information about the key to break the scheme, even if the amount of leakage per time period is small. However, the public key should *not* be changed by the secret key update. For example, a signature verification key should remain valid throughout the lifetime of the scheme.

Next we note that leakage can be *processing leakage* (i.e., only computation leaks) so that physical computations on the device are the only source of leakage. Procedures such as key generation, key updates, and other kinds of processing (i.e., signing, decryption, etc.) which involve random values may be allowed different amounts of leakage, but are all related to some physics of computations (e.g., timing of operations). The influential works of Ishai, Sahai, and Wagner [ISW03] and Micali and Reyzin [MR04] enable us to construct schemes under the "any computation, and only computation, leak information," model, which has led to many recent achievements. In contrast, *memory leakage* [AGV09] (which, in some sense, can be traced to the original works of Shamir and Rivest mentioned above) are produced as a function of the memory state itself. This type of leakage is orthogonal to computational leakage: an adversary can get memory leakage by probing memories even if the memories are not currently used in any computation (e.g., the cold-boot attacks [HSH+08]). For example, the scheme of [DHLW10] is secure against memory attacks (even continual), but assumes that the signing process leaks no information. The most general model allows *full leakage* which includes leakage both from processing and memory.

The most demanding case for designing digital signature schemes seems to be the case of adaptive and continual full leakage that is available to the adversary from both computational and memory sources (without protection of sub-steps of computations). However, to date, there are no known schemes which achieve a digital signature scheme in this adversarial setting in the standard model. All known schemes with full (memory and processing) leakage either do not have a key update algorithm and thus are not continual (cf., [KV09]), have a key update algorithm but require some restrictions (e.g., [ADW09] which requires an additional leakage-free master key), or are based on the random oracle model (with a relaxation of the definition of a "time period") [BKKV10].

1.1 Our Contributions

We propose the first signature scheme satisfying all of the above requirements, whose security can be proven in the standard model and without further relaxation. Specifically, our scheme protects against (1) continual memory leakages combined with (2) all types of continual processing (computational) leakages, namely leakages from key generation, key updates, and signing.

Moreover, the amount of information that our scheme allows to leak in each time period is optimal, in the sense that our scheme remains secure even if $1 - o(1)$ fraction of the secret key of each time period is leaked. Here "time period" is the period between two consecutive updates of the secret key (the time period is a function of the accumulated leakage itself and not a relaxed notion which depends on other parameters). We stress that our scheme remains secure even when the leakage during signing is a function $f(sk, r)$ of both the secret key and the randomness. Further, the function f can be adaptively chosen by the adversary. Using standard techniques, our scheme also allows $O(\log \kappa)$ leakage in the key generation and in each of the key updates, where κ is a security parameter. (The secret key has $O(\kappa)$ bit length. The fraction of leakage is therefore $O(\log \kappa/\kappa)$.)

Comparison with Recent Schemes. Let us compare our scheme with the recent breakthrough results of Dodis, Haralambiev, Lopez-Alt, and Wichs [DHLW10] and Brakerski, Kalai, Katz, and Vaikuntanathan [BKKV10]. The signature scheme of [DHLW10], as noted above, has to assume that there is no leakage from the signing process. The work in [BKKV10] proposed two signature schemes. Their first scheme is secure on if there is no leakage from the signing process. The second scheme relies on random oracle to protect against leakage during signing, and further requires signatures to be treated as leakage. That is, even if there is no actual side-channel leakage during a certain time period, the signing key must be refreshed to preserve security. In contrast, our signature scheme is proved secure in the standard model and the key needs to be refreshed only if leakage occurs (i.e. signatures do not constitute leakage).

Other related work. In the recent years a very rich body of research on leakage resilient cryptography has been developed. Dziembowski and Pietrzak [DP08], and Pietrzak [P09] describe the first stream ciphers resilient to continual leakage in the only computation leaks model. Faust *et al* [FKPR10] construct signature schemes resilient to continual leakage in the only computation leaks model. Faust *et al* [FRR+10] give a general compiler using secure hardware that protects an arbitrary circuit against continual leakage that can be modeled as a shallow (AC^0) boolean circuit. Juma and Vahlis [JV10], and separately Goldwasser and Rothblum [GR10], give compilers that protect any algorithm against continual leakage (without complexity restrictions), using secure hardware. Recently, Dodis and Pietrzak [DP10] show how to build continual leakage resilient pseudorandom functions that are secure against non-adaptive leakage.

Discussion on processing leakage. Continual memory attacks are a very powerful leakage model that allows the adversary to continuously obtain leakage from the entire secret key. A natural question to ask is whether adding processing leakage into the mix adds anything to adversarial power. Indeed, the only additional information available to the adversary during processing leakage is ephemeral randomness that is used in the computation. In many cases, such as in the case of public key encryption or decryption (using the corresponding private key), leakage on ephemeral randomness does not provide any useful information to the adversary about the secret key. In fact, in public key encryption and decryption, the adversary can simulate the leakage from the randomness of these algorithms on her own. However, this is not the case for signature schemes.

In a signature scheme, a signature is computed using the secret key, and made public. Consequently, signatures can viewed as a very restricted type of leakage on the secret key. A signature scheme is considered secure if such "signature leakage" is useless to any efficient adversary. When the adversary is given leakage from the randomness of the signing process, she may be able obtain information that will allow her to extract useful information from the accompanying signature. For example, each signature may contain an encryption of the secret key under a random key that is generated during signing, and then forgotten. If the adversary is able to leak this random key, she trivially breaks the security of the scheme.

1.2 The Idea Behind Our Construction

We are motivated in our construction by the (non-continual) memory-attack resilient signature schemes of Alwen, Dodis, and Wichs [ADW09], and of Katz and Vaikuntanathan [KV09]. Both [ADW09] and [KV09] use the following high level approach, which is based on the Fiat-Shamir heuristic [FS86]: a signature of a message M relative to a public key pk is an extractable proof of knowledge (in the random oracle model) of a value sk for which $H(sk) = pk$. Here is H is a hash function.

The security proof of these signature schemes relies on the *second preimage resistance* of the hash function H, and the witness extractability of the proofs that are used. That is, they use the property that it is infeasible to find $sk^* \neq sk$ satisfying $H(sk^*) = H(sk)$.

To prove security, they construct a simulator that generates a secret key sk randomly and computes $pk = H(sk)$. The simulator then can answer leakage queries and signing queries using the secret key that it itself has generated. If an adversary can forge a message/signature pair, the simulator extracts the witness sk'. Now, if the fraction of leakage of sk is less than $1 - o(1)$, the exact key sk that is used by the simulator is information theoretically undetermined in the view of the adversary (specifically, there are at least two possible keys, given the leakage). Therefore, with probability at least $1/2$, the witness sk' is different from sk, which breaks the second pre-image resistance of H.

We start with this basic approach, and enhance it along three dimensions. Specifically, we:

1. Remove the requirement for a random oracle, and get a scheme secure in the standard model.
2. Add a key update procedure that refreshes the secret key, while keeping the public key fixed. This yields a signature scheme resilient against *continual* memory attacks [BKKV10].
3. Develop a proof method that allows leakage of randomness used in signing within a period (allowing optimal leakage).

1.2.1 Removing the Random Oracle

The simplest idea to remove the random oracle from the scheme of [ADW09, KV09] is to prove the knowledge of the secret key not by using Fiat-Shamir heuristic, but by using other known non-interactive proof systems in the standard model.

This initial attempt fails for the following reason: the argument of [ADW09, KV09] showing $sk^* \neq sk$ is purely information theoretic. Hence, if we want to use the argument of [ADW09, KV09], the proof systems should hide sk not in a computational sense, but in an information theoretic sense. But if the proof system hides sk information theoretically, the secret key sk^* used by an adversary is also hidden information theoretically (since no random oracle is available). Hence, it is impossible for the simulator to get the second pre-image sk^* from the adversary's forgery, and so we cannot rely on second pre-image resistance.

To overcome the above problem, we use the Waters' function

$$h(\mathcal{H}, M) = H_0 + \sum_k M_k H_k,$$

where M_k is the k-th bit of a message M and $\mathcal{H} = (H_0, \ldots, H_m)$ is a tuple of group elements[1]. The function is introduced in [W05] in order to construct an adaptively secure ID-based encryption scheme. The Waters' function has the property that a simulator can choose the parameters $H_0, \ldots, H_m$ in a special way that defines a subset $\mathcal{M}$ of the range of h. It can then be shown that with non-negligible probability, all signing queries M of the adversary satisfy $h(\mathcal{H}, M) \in \mathcal{M}$, and the forgery M^* satisfies $h(\mathcal{H}, M^*) \notin \mathcal{M}$.

We construct a Waters-like function h' such that $\mathcal{M}$ is a set of all non-DDH tuples in the range of h'. Consequently, we get that with non-negligible probability all signing queries of the adversary map to non-DDH tuples, and the forgery maps to a DDH tuple.

We then combine the above hash function h' with the Groth-Sahai [GS08] proof system. Groth-Sahai is a proof system which uses a common reference string (CRS). The proof system hides the witness information theoretically if the CRS is a non-DDH tuple and hides it only computationally, and the witness therefore is extractable, if the CRS is a DDH tuple.

Hence, by using Groth-Sahai proofs as signatures, and $h'(M)$ as the CRS of the Groth-Sahai proof, we get a proof system that hides the witness sk information theoretically when the simulator generates proofs as answers to signing queries, and allows the witness sk^* to be extracted from the proof generated by an adversary as a forged signature.

The scheme before adding key update. Our scheme therefore has the following basic structure. The private key consists of a vector of group elements $\vec{W}$, and the public key consists of group elements $\vec{A}$ and T such that $e(\vec{A}, \vec{W}) = T$. The public key also contains the description of a Waters-like hash function h'. To sign a message M, we first use h' to compute a CRS $h'(\mathcal{H}, M)$ for the Groth-Sahai proof system. Then, the signature is a proof, under the CRS $h'(\mathcal{H}, M)$, that $e(\vec{A}, \vec{W}) = T$.

Before proceeding to describe our key update procedure, we believe it is instructive to see why the above scheme, without update, is secure. Intuitively, this follows from the second pre-image resistance of the hash function $H_{\vec{A}}(\vec{X}) := e(\vec{A}, \vec{X})$. From the perfect witness indistinguishability of Groth-Sahai proofs we know that the adversary learns about the specific witness $\vec{W}$ only from leakage. However, since the amount of leakage is bounded, the actual witness $\vec{W}$ in the private key remains information theoretically undetermined. Finally, when the adversary submits a successful forgery, we use the indistinguishability of common reference strings technique discussed above to show that, with non-negligible probability, a witness $\vec{W}'$ can be extracted from the forgery. This witness is likely to be different from $\vec{W}$, which would give the simulator two inputs $\vec{W}$ and $\vec{W}'$ such that $H_{\vec{A}}(\vec{W}) = H_{\vec{A}}(\vec{W}')$. This in turn violates the second pre-image resistance of $H_{\vec{A}}$.

We remark that the above technique is somewhat reminiscent of the Feige-Lapidot-Shamir [FLS90] method for using witness indistinguishability to achieve witness hiding.

1.2.2 Adding Key Updates

The approach of Section 1.2.1 allows us to move from the random oracle to the standard model. However, the above scheme can still tolerate only a bounded amount of leakage. We now describe a method for choosing the private keys of the scheme that allows us to randomize the key without having to issue a new public key.

[1]Here we use *additive notation* to describe the function.

We start by observing that if our scheme relies for security on the second pre-image resistance of the hash function H, then no key update algorithm can exist. This is because otherwise we could use the key update algorithm itself to break second pre-image resistance as follows:

> If we can get new key $sk^{[i+1]}$ efficiently by updating $sk^{[i]}$, this means that one can get two keys $sk^{[i]}$ and $sk^{[i+1]}$ satisfying of $T = H(sk^{[i]})$ and $T = H(sk^{[i+1]})$, where T is the public key. The function H therefore cannot be second pre-image resistant.

Note that our model does not allow to update the public key for the convenience of verifiers. The above collision of the function H is therefore unavoidable.

(n, k)-independent pre-image resistant (IPIR) hash functions. We overcome the above problem by introducing the following new notion: (n, k)-*independent pre-image resistant hash function* H, where n is an integer and $k \leq n - 2$. This is a linear function H from an n-dimensional vector space $\mathbb{H}^n$ to a 1-dimensional vector space $\mathbb{T}$, over $\mathbb{Z}_p$. We require that, given certain trapdoor information about H, one can find a tuple $(\vec{Y}_1, \ldots, \vec{Y}_{k+1}, T)$ satisfying $T = H(Y_j)$ for all $j \in [k + 1]$. However, it must be infeasible to find $\vec{Y}_* \in \mathbb{H}^n$ satisfying $T = H(\vec{Y}_*)$ and $\vec{Y}_* \notin \mathsf{Aff}(\vec{Y}_1, \ldots, \vec{Y}_{k+1})$, where $\mathsf{Aff}(\vec{Y}_1, \ldots, \vec{Y}_{k+1})$ is the smallest affine space spanned by $\vec{Y}_1, \ldots, \vec{Y}_{k+1}$. We call the hardness property (n, k)-*independent preimage resistance*.

We note that although in this paper we give a construction of an (n, k)-IPIR hash function, we do not use it as a black box in our signature scheme. Indeed, the Groth-Sahai proofs that are generated use the parameters of the hash function, which are of a very specific form. However, the notion of IPIR seems useful in itself, and we hope that other applications will be found for it. Furthermore, abstracting the properties that we need from H allows us to present a modular and structured proof of security for our signature scheme.

Generating and updating keys. Using the linearity of H, we can generate any linear sum of $\vec{Y}_1, \ldots, \vec{Y}_{k+1}$ in polynomial time. We use this property to perform key update. And we use the IPIR (instead of the second pre-image resistance) when we show that no adversary can forge a signature.

In slightly greater detail, we construct the key generation algorithm and the update algorithm of our scheme as follows. In the key generation, by using the trapdoor of the hash function H, we generate $(\vec{Y}_1, \vec{Y}_2, T)$ satisfying $T = H(\vec{Y}_1) = H(\vec{Y}_2)$. We then compute $\vec{Q} \leftarrow \vec{Y}_1 - \vec{Y}_2$ and publish $\vec{Q}$ as a part of the public key. The secret key is $\vec{W}^{[0]} \leftarrow \vec{Y}_2$. Note that $H(\vec{Q}) = H(\vec{Y}_1) - H(\vec{Y}_2) = 0$ holds.

Key update then works as follows: in the $(i + 1)$-st round, we select $s^{[i]} \overset{\$}{\leftarrow} \mathbb{Z}_p$ randomly and compute $\vec{W}^{[i+1]} \leftarrow \vec{W}^{[i]} + s^{[i]}\vec{Q}$. Based on the linearity of H, and the equality $H(\vec{Q}) = 0$, one can easily show that $H(\vec{W}^{[i+1]}) = H(\vec{W}^{[i]}) = \cdots H(\vec{W}^{[0]}) = H(\vec{Y}_2) = T$ holds, and that $\vec{W}^{[i]}$ is an element of $\mathsf{Aff}(\vec{Y}_1, \vec{Y}_2)$. The latter holds because $\vec{W}^{[i]}$ are linear sums of $\vec{W}^{[0]} = \vec{Y}_2$ and $\vec{Q} = \vec{Y}_1 - \vec{Y}_2$. We then use an adaptation of the "leakage resilient subspace" technique from [BKKV10] to show that the affine subspace $\mathsf{Aff}(\vec{Y}_1, \vec{Y}_2)$ (or even $\mathsf{Aff}(\vec{Y}_1, \ldots, \vec{Y}_{k+1})$) is hidden from the adversary, even given continual leakage (assuming the refresh procedure above is periodically performed). Given the hiding property of the affine space, it follows that if the adversary forges a signature $\sigma^* = \mathsf{Prf}_{M^*}(\vec{W}^*)$ for some message M^*, she likely uses a witness $\vec{W}^* \notin \mathsf{Aff}(\vec{Y}_1, \vec{Y}_2)$. However, this violates the IPIR of H. The security of the scheme follows.

1.2.3 Security Against Leakage in Signing

The main challenge in achieving security under leakage from the signing process is that the signature and the secret key are correlated through the randomness that was used to produce the signature. When the adversary obtains leakage on the randomness, the signature may become much more valuable, and potentially allow the adversary to break the scheme (as we discussed in the introduction).

In the proof based signature schemes we described above, there is no guarantee that leakage on the randomness of the prover does not break the zero-knowledge or witness indistinguishability property of the proof system. We solve this problem through a combination of several tools: first, as described above, we rely on Groth-Sahai proofs, which have a dual mode – witness hiding and witness binding. When the proof is (perfectly) hiding, it is information theoretically independent from the witness, which is the secret key, if there is no leakage on the randomness of the prover.

We use the above fact to "invert" the order in which components of the signature are generated: first the GS proof σ in the signature is generated using some globally fixed witness $\mathcal{Y}$ (note that this is only done by the simulator in the analysis, and so there is no leakage on the witness $\mathcal{Y}$). Then, given an actual witness $\vec{W}$ for the proof, we "reverse engineer" the randomness R that would yield the same proof σ, and compute leakage on $(\vec{W}, R)$. We use an additional property of Groth-Sahai that for every pair of proof and witness $(\sigma, \vec{W})$ there exists a unique randomness $R_{\sigma,\vec{W}}$ that causes the prover to output σ given witness $\vec{W}$. Moreover, since the proof is perfectly witness hiding, for all witnesses $\vec{W}$, the distribution on the tuple $(\sigma, R_{\sigma,\vec{W}})$ are identical whether we first generate the proof using witness $\vec{V}$ and then determine the randomness, or choose the randomness uniformly, and compute the proof directly.

The above approach, however, does not work as is, since the process of finding R may not be efficiently computable! We therefore rely on an information theoretic leakage resilience property of random subspaces that was shown in [BKKV10] (in fact, we prove a slightly stronger version that suits our construction). We combine both techniques together, simultaneously using the randomness reverse engineering technique described above to handle leakage from signing, and information theoretic indistinguishability of random subspaces under leakage. Using these techniques together, we show that the adversary is unable to gain information about the subspace from which we generate private keys during update, even if leakage on the signing process is available. We then use independent pre-image resistance to conclude that our scheme is secure.

2 Preliminaries

Notations. Let $[n]$ denote $\{1, \dots, n\}$ and $[k..n]$ denote $\{k, \dots, n\}$. For two random variables X and Y, $\mathsf{dist}(X, Y)$ denote the statistical distance between X and Y.

Linear Algebra. Unless otherwise stated, vectors in this paper are column vectors. We represent a row vector as a transpose of a column vector in this paper. For natural numbers n and m, let $\mathbb{Z}_p^{n\times m}$ denote the set of $n \times m$ matrices over $\mathbb{Z}_p$. Let A^T denote the transposed of a matrix $A = (a_{i,j})_{i,j}$, that is, $A^T = (a_{j,i})_{i,j}$. For two vectors $\vec{u} = (u_1, \dots, u_n)$, $\vec{v} = (v_1, \dots, v_n) \in \mathbb{Z}_p^n$, we let $\langle \vec{u}, \vec{v} \rangle$ denote the inner product of them in $\mathbb{Z}_p$, that is, $\langle \vec{u}, \vec{v} \rangle = \sum_i u_i v_i \mod p$.

For a vector $\vec{v} = (v_1, \dots, v_n) \in \mathbb{Z}_p^n$ and an element W of the group $\mathbb{G}$ or $\mathbb{H}$, let $\vec{v}W$ denote the vector $(v_1 W, \dots, v_n W)$.

For a vector space $\mathcal{V}$ and vectors $\vec{Y}_1, \dots, \vec{Y}_{k+1} \in \mathcal{V}$, let $\mathsf{Span}(\vec{Y}_1, \dots, \vec{Y}_k)$ denote the smallest vector subspace of $\mathcal{V}^n$ which contains all of $(\vec{Y}_j)_{j\in[k]}$, that is,

$$\mathsf{Span}(\vec{Y}_1, \dots, \vec{Y}_k) = \{\vec{Y} \in \mathcal{V}^n \mid \exists s_1, \dots, s_k \in \mathbb{Z}_p : \vec{Y} = \sum_{j\in[k]} s_j \vec{Y}_j\}.$$

Similarly, let $\mathsf{Aff}(\vec{Y}_1, \dots, \vec{Y}_{k+1})$ is the smallest affine subspace of $\mathcal{V}^n$ which contains all of $(\vec{Y}_j)_{j\in[k+1]}$, that is,

$$\mathsf{Aff}(\vec{Y}_1, \dots, \vec{Y}_{k+1}) = \{\vec{Y} \in \mathcal{V}^n \mid \exists s_1, \dots, s_{k+1} \in \mathbb{Z}_p : \vec{Y} = \sum_{j\in[k+1]} s_j \vec{Y}_j, \ \sum_{j\in[k+1]} s_j = 1\}.$$

Note that the space $\mathsf{Aff}(\vec{Y}_1, \ldots, \vec{Y}_{k+1})$ becomes k dimensional, when $k+1$ vectors $\vec{Y}_1, \ldots, \vec{Y}_{k+1}$ are linear independent.

2.1 Signature Schemes Resilient Against Continual Leakage

A *signature scheme with key update* $\mathcal{SGN}$ consists of four algorithms Kg, Sig, Ver, and Update. The inputs and outputs of Kg, Sig, and Ver are the same as in standard signature schemes. Update takes as input a secret key and a public key and outputs a new element of the secret key space. $\mathcal{SGN} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Update})$ has to satisfy the following property:

- (**Correctness**) For any integer $n \geq 0$ and any message M, if we compute $(pk, sk_0) \leftarrow \mathsf{Gen}(1^\kappa)$, $sk_1 \leftarrow \mathsf{Update}_{pk}(sk_0)$, ..., $sk_n \leftarrow \mathsf{Update}_{pk}(sk_{n-1})$, and $\sigma \leftarrow \mathsf{Sig}(sk_n, M)$, $\mathsf{Ver}(pk, M, \sigma) = 1$ always holds.

We follow the definition of [BKKV10] of leakage resilient signatures.

Definition 1 ([BKKV10]). Let ρ_G, ρ_U, ρ_M, and ρ_S be elements of the real range $[0,1]$. We say that $\mathcal{SGN} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Update})$ is $(\rho_G, \rho_U, \rho_M, \rho_S)$- *EU-CMA-CML secure* (stand for existentially unforgeable under chosen message attack in the CML model) if no PPT adversary $\mathcal{A}$ succeeds in the game of Fig.1 with non-negligible probability. Here $\mathsf{Rnd}[\mathsf{Algo}]$ denote the set of randomnesses for algorithm Algo.

Setup $\mathcal{A}(1^\kappa)$ sends to the challenger a function f satisfying $|f(R)| \leq \rho_G|R|$ for all R. The challenger then selects $R \xleftarrow{\$} \mathsf{Rnd}[\mathsf{Gen}]$ computes $(pk, sk_0) \leftarrow \mathsf{Gen}(1^\kappa; R)$ sends $(pk, f(R))$ to $\mathcal{A}$, and initializes $i \leftarrow 0$ and $L_i \leftarrow |f(R)|$. Here i represents the number of updates and L_i denote the bit length of all leakages about the i-th secret key.

Queries $\mathcal{A}$ makes queries of the following three types polynomial number of times:

- Update queries (update, f) where f is a circuit satisfying $|f(sk, R)| \leq \rho_U(|sk| + |R|)$ for any (sk, R). If $L_i + |f(sk_i, R)| \leq \rho_M |sk_i|$ holds, the challenger chooses $R \xleftarrow{\$} \mathsf{Rnd}[\mathsf{Update}]$ randomly, computes $sk_{i+1} \leftarrow \mathsf{Update}_{pk}(sk_i)$ and sends $f(sk_i, R)$ back to $\mathcal{A}$ and resets $i \leftarrow i+1$ and $L_i \leftarrow |f(sk_i, R)|$. Otherwise, the challenger aborts.
- (Memory) leak queries (leak, f), where f is a circuit. If $L_i + |f(sk_i)| \leq \rho_M |sk_i|$ holds, the challenger sends $f(sk_i)$ to adversary and resets $L_i \leftarrow L_i + |f(sk_i)|$. Otherwise, the challenger aborts.
- Signing queries (sig, M, f) where f is a circuit with $|f(sk, R)| \leq \rho_S(|sk| + |R|)$ for any (sk, R). The challenger chooses $R \leftarrow \mathsf{Rnd}[\mathsf{Sig}]$ randomly, computes $\sigma \leftarrow \mathsf{Sig}(sk_i, M; R)$ and sends $(\sigma, f(sk_i, R))$ back to $\mathcal{A}$.

Challenge Assuming the challenger did not aborts, $\mathcal{A}$ outputs (M_*, σ_*). It succeeds if $\mathsf{Ver}(pk, M_*, \sigma_*) = 1$ holds and $\mathcal{A}$ never made query (sig, M_*).

Figure 1: Game of $(\rho_G, \rho_U, \rho_M, \rho_S)$-EU-CMA-CML secure

2.2 Bilinear Pairings

In our paper, we are working on a bilinear pairing, $\mathsf{e} : \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{T}$ with prime order p, where $\mathbb{G} \neq \mathbb{H}$ holds and there is no efficiently computable homomorphism between two groups $\mathbb{G}$ and $\mathbb{H}$ (Such a pairing is called Type III [GPS08]). We denote by gk bilinear map parameters of the form $(p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$. We will occasionally refer to gk as *group description*.

Our proofs rely on basic properties of linear algebra. We therefore find it convenient to use *additive notation* for pairings. For example, we write $\mathsf{e}((a+b)A, W) = a \cdot \mathsf{e}(A, W) + b \cdot \mathsf{e}(A, W)$. For two (column) vectors $\vec{A} = (A_1, \ldots, A_n)^{\mathsf{T}} \in \mathbb{G}$ and $\vec{W} = (W_1, \ldots, W_n)^{\mathsf{T}} \in \mathbb{H}$, we denote

$$\mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{W}) = \sum_{i \in [n]} \mathsf{e}(A_i, W_i).$$

Assumption 2 (SXDH assumption [GS08]). *We say that $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$ satisfies the Symmetric external Diffie-Hellman (SXDH) assumption, if the DDH assumption holds both over $\mathbb{G}$ and $\mathbb{H}$ (note that this is possible in type III pairings).*

2.3 Groth-Sahai Proofs

Groth and Sahai [GS08] proposed efficient non-interactive witness indistinguishable proof systems for settings where a bilinear pairing is available. Their system allows efficient proofs of various statements about the groups of the pairing, and the pairing relation.

In this work we prove statements of the form $\mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{W}) = T$, where an instance $(\vec{A}, T) \in \mathbb{G}^n \times \mathbb{T}$ is the input to the verifier, and $\vec{W}$ is the witness used by the prover.

Let $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$ be a group description and $crs = (\vec{G}, \vec{H}) \in \mathbb{H}^2$. The Groth-Sahai proof using crs as CRS (Common Reference String) is as follows.

- $\mathsf{Prf}(gk, crs, (T, \vec{A}), \vec{W})$: Parse gk and crs as $(p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$ and $(\vec{G}, \vec{H})$ respectively. Select $R \xleftarrow{\$} \mathbb{Z}_p^{n \times 2}$ randomly, and compute

$$(\vec{C}, \vec{D}) \leftarrow (R \cdot \vec{G}, \vec{W} + R \cdot \vec{H})$$
$$\vec{\Pi} \leftarrow R^{\mathsf{T}} \vec{A}$$

  and output $\sigma = (\vec{C}, \vec{D}, \vec{\Pi})$.

- $\mathsf{Vrf}(gk, crs, (\vec{A}, T), \sigma)$: Parse gk, crs, and σ as $(p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$, $(\vec{G}, \vec{H})$ and $(\vec{C}, \vec{D}, \vec{\Pi})$ respectively.

  Output 1 iff the following equality holds.

$$(\mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{C}), \mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{D})) \stackrel{?}{=} (\mathsf{e}(\vec{\Pi}^{\mathsf{T}}, \vec{G}), T + \mathsf{e}(\vec{\Pi}^{\mathsf{T}}, \vec{H}))$$

One can easily show that a correctly generated proof is always accepted by Vrf.

Groth and Sahai [GS08] gave the following two algorithms HideCRS and BindCRS to generate two different types of CRS: hiding and binding. When a hiding CRS is used for the proof, the witness is perfectly (information theoretically) hidden. When a binding CRS is used, BindCRS provides a trapdoor along with the CRS, and the trapdoor can be used to extract the witness from any proof. Finally, it is required that the two types of CRS are computationally indistinguishable.

For the above proof system, let $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$. The algorithms HideCRS and BindCRS are defined as follows:

- $\mathsf{HideCRS}(gk)$: Select $\vec{G}, \vec{H} \xleftarrow{\$} \mathbb{H}^2$ randomly and output $crs \leftarrow (\vec{G}, \vec{H})$.

- $\mathsf{BindCRS}(gk)$: Select $\vec{G} \xleftarrow{\$} \mathbb{H}^2$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$ randomly, compute $\vec{H} \leftarrow \alpha\vec{G}$ and output $crs \leftarrow (\vec{G}, \vec{H})$ and the trapdoor α.

Groth-Sahai show that in the above proof system, a hiding CRS and a binding CRS are indistinguishable under the SXDH assumption. Formally, the perfect witness indistinguishability, and the witness extractability properties are defined as follows [GS08]. Below, $\mathsf{Setup}(1^\kappa)$ be an algorithm which generates a group description $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$.

- **(Composable Perfect Witness Indistinguishability)** For all (possibly unbounded) adversaries $\mathcal{A}$

$$\Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\kappa), crs \leftarrow \mathsf{HideCRS}(gk), \\ (\vec{A}, T, \vec{W}_0, \vec{W}_1, st) \leftarrow \mathcal{A}(gk, crs), \sigma \leftarrow \mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W}_0), b \leftarrow \mathcal{A}(\sigma, st) \end{array} : b = 1\right]$$

$$= \Pr\left[\begin{array}{l} gk \leftarrow \mathsf{Setup}(1^\kappa), crs \leftarrow \mathsf{HideCRS}(gk), \\ (\vec{A}, T, \vec{W}_0, \vec{W}_1, st) \leftarrow \mathcal{A}(gk, crs), \sigma \leftarrow \mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W}_1), b \leftarrow \mathcal{A}(\sigma, st) \end{array} : b = 1\right]$$

  where we require $\mathsf{e}(\vec{A}, \vec{W}^{[0]}) = \mathsf{e}(\vec{A}, \vec{W}^{[1]}) = T$.

- **(Perfect Extractability)** For all possible output $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$ of $\mathsf{Setup}(1^\kappa)$, all possible output (crs, α) of $\mathsf{BindCRS}(gk)$, all $(\vec{A}, T) \in \mathbb{G}^n \times \mathbb{T}$ and all $\sigma = (\vec{C}, \vec{D}, \vec{\Pi}) \in \mathbb{H}^2 \times \mathbb{H}^2 \times \mathbb{G}^2$ satisfying $\mathsf{Vrf}(gk, crs, (\vec{A}, T), \sigma) = 1$, if we set

$$\vec{W}^* \leftarrow \vec{D} - \alpha\vec{C},$$

  the equality $\mathsf{e}(\vec{A}, \vec{W}^*) = T$ always holds.

3 Independent Preimage Resistant (IPIR) Hash Functions

We introduce a new notion: independent pre-image resistance (IPIR), that we use in the construction and analysis of our scheme. As we have already mentioned in the introduction, our construction does not use the IPIR hash function described below in a black box way. Nevertheless, we believe it to be instructive to define this notion separately, both to conceptually isolate the properties of the hash function that we use, and for potential future use.

Definition 3 (Independent Preimage Resistant Hash Function). Let n be a positive number, and let $\mathbb{H}$ and $\mathbb{T}$ be cyclic groups of order p. Let $\mathsf{Gen}, H, \mathsf{GenSim}, \mathsf{Check}$ be polynomial time algorithms whose inputs and outputs are as follows. Below, k is the parameter representing the dimension. Note that a k-dimensional *affine* (not linear) subspace contains $k+1$ vectors and GenSim of the below therefore outputs $k+1$ vectors $\mathcal{Y}_j$.

- $\mathsf{Gen}(1^\kappa)$ outputs some public information P.
- For any possible output P of Gen, H_P is a deterministic linear function from $\mathbb{H}^n$ to $\mathbb{T}$.
- GenSim takes an integer $k \in [n-2]$ as an input and outputs a public information P, a trapdoor td, $(T, \vec{Y}_1, \dots, \vec{Y}_{k+1}) \in \mathbb{T} \times (\mathbb{H}^n)^{k+1}$ satisfying $T = H_P(\vec{Y}_i)$ for all $i \in [k+1]$.
- Check takes P, $(T, \vec{Y}_1, \dots, \vec{Y}_{k+1})$, an element $\vec{Y}'$ of $\mathbb{H}^n$, and a trapdoor td as inputs and outputs 1 or 0.

For integers n and $k \in [n-2]$, we say that H is (n,k)-*independent pre-image resistant* with respect to $(\mathsf{Gen}, \mathsf{GenSim}, \mathsf{Check})$ if it satisfies the following properties.

- (**Correctness**) For all outputs $(P, td, (T, \vec{Y}_1, \ldots, \vec{Y}_{k+1}))$ of GenSim and all $\vec{Y}' \in \mathbb{H}^n$, $\mathsf{Check}(P, td, (T, \vec{Y}_1, \ldots, \vec{Y}_{k+1}), \vec{Y}') = 1$ holds iff $T = H_P(\vec{Y}')$ and $\vec{Y}' \in \mathsf{Aff}(\vec{Y}_1, \ldots, \vec{Y}_{k+1})$ holds.

  Moreover, for an output $(P, td, (T, \vec{Y}_1, \ldots, \vec{Y}_{k+1}))$ of GenSim, P have the same distribution as an output of $\mathsf{Gen}(1^\kappa)$ and $(\vec{Y}_1, \ldots, \vec{Y}_{k+1})$ uniformly distributed on $\{\vec{Y} \in \mathbb{H}^n \mid H_P(\vec{Y}) = T\}^{k+1}$.

- ((n,k)**-Independent Preimage Resistance**) For any polytime adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l} (P, td, (T, \vec{Y}_1, \ldots, \vec{Y}_{k+1})) \leftarrow \mathsf{GenSim}(1^\kappa), \\ \vec{Y}_* \leftarrow \mathcal{A}(P, T, \vec{Y}_1, \ldots, \vec{Y}_{k+1}) \end{array} : \begin{array}{l} T = H_P(\vec{Y}_*) \\ \vec{Y}_* \notin \mathsf{Aff}(\vec{Y}_1, \ldots, \vec{Y}_{k+1}) \end{array}\right]$$

  is negligible.

  Note that one can check whether $\vec{Y}_* \notin \mathsf{Aff}(\vec{Y}_1, \ldots, \vec{Y}_{k+1})$ holds or not in polytime by using Check and the a trapdoor td.

Construction of an independent pre-image resistant function. In our signature scheme we use the function $H_{\vec{A}}(\vec{Y}) = \mathsf{e}(\vec{A}, \vec{Y})$. The algorithms Gen, GenSim, and Check for the function H are as follows. Below, $\mathsf{Setup}(1^\kappa)$ is an algorithm which generates a group description $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$.

- $\mathsf{Gen}(1^\kappa)$: Compute $gk \leftarrow \mathsf{Setup}(1^\kappa)$ and generates $\vec{A} \xleftarrow{\$} \mathbb{G}^n$ randomly, and output $P \leftarrow (gk, \vec{A})$.

- $\mathsf{GenSim}(1^\kappa)$: Compute $gk \leftarrow \mathsf{Setup}(1^\kappa)$, and choose randomly $A \xleftarrow{\$} \mathbb{G}$, and $\vec{a} \xleftarrow{\$} \mathbb{Z}_p^n$. Compute $\vec{A} \leftarrow \vec{a}A$. Choose randomly $\vec{Y} \xleftarrow{\$} \mathbb{H}$, $t \xleftarrow{\$} \mathbb{Z}_p$, and $\vec{y}_j \in \mathbb{Z}_p^n$ satisfying $\langle \vec{a}, \vec{y}_j \rangle = t$ for $j \in [k+1]$. Choose $n-k$ linearly independent vectors $\vec{e}_1, \ldots, \vec{e}_{n-k}$ satisfying $\langle \vec{e}_i, \vec{y}_j \rangle = 0$ for all $i \in [n-k]$, $j \in [k]$. Output $P = (gk, \vec{A})$, $td \leftarrow (\vec{e}_i)_{i \in [n-k]}$, and $\vec{Y}_j \leftarrow \vec{y}_j \vec{Y}$ for $j \in [k]$, $T \leftarrow t\vec{Y}$.

- $\mathsf{Check}(P, td, (T, \vec{Y}_1, \ldots, \vec{Y}_{k+1}), \vec{Y}')$: Parse P and td as $(gk, \vec{A})$ and $(\vec{e}_i)_{i \in [n-k]}$ respectively. Output 1 iff $\mathsf{e}(\vec{A}, \vec{Y}') = T$ and $\langle \vec{e}_i, \vec{Y}' \rangle = 0$ holds for all $i \in [n-k]$.

Proposition 4. *For any n and $k \leq n-2$, the scheme* $(\mathsf{Setup}, H, \mathsf{GenSim}, \mathsf{Check})$ *is (n,k)-independent pre-image resistant under the SXDH assumption.*

Correctness is straightforward. We give the proof of independent pre-image resistance in Appendix A.1.

4 Proposed Scheme

Let $n \geq 3$ and m be integers. Let Setup be a polytime algorithm that generates a group description $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$, as discussed above, where $\mathsf{e} : \mathbb{G} \times \mathbb{H} \to \mathbb{T}$. For $\mathcal{H} = (\vec{H}_0, \vec{H}_1, \ldots, \vec{H}_m) \in (\mathbb{H}^2)^{m+1}$ and $M \in \{0,1\}^m$, we define a Water's hash function h as

$$h_{gk}(\mathcal{H}, M) = \vec{H}_0 + \sum_{k \in [m]} M_k \vec{H}_k,$$

where M_k is the k-th bit of M. Let Prf and Vrf be the proof algorithm and the verification algorithm of the Groth-Sahai proof system reviewed in Section 2.3. Our signature scheme $\mathcal{SGN} = (\mathsf{Kg}, \mathsf{Update}, \mathsf{Sig}, \mathsf{Ver})$ works as follows.

Key Generation $\mathsf{Kg}(1^\kappa)$**:** $gk \leftarrow (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e}) \leftarrow \mathsf{Setup}(1^\kappa)$, $\vec{G} \leftarrow \mathbb{H}^2, \mathcal{H} \leftarrow (\vec{H}_0, \vec{H}_1, \ldots, \vec{H}_m) \leftarrow (\mathbb{H}^2)^{m+1}$.

Randomly select $A \overset{\$}{\leftarrow} \mathbb{G}$, $Q \overset{\$}{\leftarrow} \mathbb{H}$, and $\vec{a}, \vec{q} \overset{\$}{\leftarrow} \mathbb{Z}_p^n$ satisfying $\langle \vec{a}, \vec{q} \rangle = 0$ and compute $\vec{A} \leftarrow \vec{a}A$, $\vec{Q} \leftarrow \vec{q}Q$. Select $\vec{W}^{[0]} \overset{\$}{\leftarrow} \mathbb{H}^n$ randomly, compute $T \leftarrow \mathsf{e}(\vec{A}, \vec{W}^{[0]})$, and outputs $pk \leftarrow (gk, \vec{G}, \mathcal{H}, \vec{A}, T, \vec{Q})$ and $sk^{[0]} \leftarrow \vec{W}^{[0]}$.

Key Update $\mathsf{Update}_{pk}(sk^{[i]})$**:** Parse pk and $sk^{[i]}$ as $(gk, \vec{G}, \mathcal{H}, \vec{A}, T, \vec{Q})$ and $\vec{W}^{[i]}$ respectively, select $s \overset{\$}{\leftarrow} \mathbb{Z}_p$ randomly, and output $sk^{[i+1]} \leftarrow \vec{W}^{[i+1]} \leftarrow \vec{W}^{[i]} + s\vec{Q}$.

Signing $\mathsf{Sig}(sk^{[i]}, M)$ **for** $M \in \{0,1\}^m$**:** Parse pk and $sk^{[i]}$ as $(gk, \vec{G}, \mathcal{H}, \vec{A}, T, \vec{Q})$ and $\vec{W}^{[i]}$. Compute $\vec{H}_M \leftarrow h_{gk}(\mathcal{H}, M)$, set $crs_M \leftarrow (\vec{G}, \vec{H}_M)$, and $\sigma \leftarrow \mathsf{Prf}(gk, crs_M, (\vec{A}, T), \vec{W}^{[i]})$ and output σ.

Verification $\mathsf{Ver}(pk, M, \sigma)$**:** Parse pk as $(gk, \vec{G}, \mathcal{H}, \vec{A}, T, \vec{Q})$, compute $\vec{H}_M \leftarrow h_{gk}(\mathcal{H}, M)$, and set $crs_M \leftarrow (\vec{G}, \vec{H}_M)$. If $\mathsf{Ver}(gk, crs_M, (\vec{A}, T), \sigma) = 1$, output 1. Otherwise, output 0.

Theorem 5. *For any constants $c > 0$ and any $\gamma = \Theta(1/\sqrt{\kappa})$, the proposed scheme $\mathcal{SIG}$ is $(\rho_G, \rho_U, \rho_M, \rho_S)$-EU-CMA-CML secure under the SXDH assumption. Here*

$$(\rho_G, \rho_U, \rho_M, \rho_S) = \left(\frac{c \cdot \log k}{n \log p}, \frac{c \cdot \log k}{n \log p}, 1 - \frac{2 + \gamma}{n}, 1 - \frac{2 + \gamma}{n} \right).$$

We can achieve the fraction $1 - o(1)$ of leakage in signing and in memory by setting $n = \kappa$.

4.1 Security Analysis

4.1.1 Overview

Our proof starts with a reduction that shows how to convert any adversary that obtains leakage on key generation and updates, to an adversary that does not require such leakage. This follows from the lemma of Brakerski *et al* [BKKV10]. (See Appendix A.2 for the proof.)

Lemma 6 (Informally given in [BKKV10])**.** *Let $\mathcal{SGN} = (\mathsf{Kg}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{Update})$ be a $(0, 0, \rho_M, \rho_S)$-EU-CMA-CML secure signature scheme. Then if $\rho_M = \omega(\log n)$, it is also $(c \log \kappa/m, c \log \kappa/m, \rho_M, \rho_S)$-EU-CMA-CML secure for any c. Here m is the maximum of the length of secret key.*

The proof of Theorem 5 then proceeds by a sequence of games (depicted in Fig.2):

1. In games 1-3 we follow a standard argument about the properties of Waters' hash. Specifically, we show that with non-negligible probability the common reference strings determined by the messages that the adversary submits in signing queries are hiding CRS (and therefore hide the witness perfectly), and the CRS of the forgery is binding (and therefore the witness can be extracted).

   This part of discussion of the our proof is essentially the same as that of [W05] (simplified by [BR09]).

2. In game 4, we change the way that secret keys are generated. Instead of being generated by the update algorithm, the secret key is now randomly chosen from an n-3 dimensional subspace $\mathcal{W}$ of the set $\mathcal{Y} = \{\vec{W} | \mathsf{e}(\vec{A}, \vec{W}) = T\}$ of valid secret keys (which is of dimension $n - 1$).

   Indistinguishability with game 3 follows from the DDH assumption on the group $\mathbb{H}$.

$$\overbrace{\text{Original——Game }}^{\text{leak. of key gen and updates becomes 0}}\overbrace{\text{0——Game 1——Game 2——Game }}^{\text{Waters hash}}\overbrace{\text{3——Game }}^{\vec{W}^{[i]}\text{ becomes a random element of }\mathcal{W}}\overbrace{\text{4——neg.}}^{\text{IPIR}}$$

$$\text{stat. indis.}\left\{\,\Big\|\right.$$

$$\text{Game}^{\dagger}\ 4\ \ (\vec{W}^{[i]} \xleftarrow{\$} \mathcal{Y})$$

Figure 2: Games in the reduction

3. Game† 4 is described only to prove a specific property of Game 4 that we need. In Game† 4 the keys are chosen randomly from the space $\mathcal{Y}$ of all valid secret keys.

   We rely on the perfect witness hiding of Groth-Sahai proof and a lemma from [BKKV10] to show that game 4 and Game† 4 are *statistically* indistinguishable.

   We then obtain the property of Game† 4 that the subspace $\mathcal{W}$ is information theoretically hidden, and this property transfers to Game 4 due to the indistinguishability of the two games.

4. Finally, in Game 4 we use the fact that $\mathcal{W}$ is information theoretically hidden from the adversary to argue that the witness $\vec{W}^*$ extracted from the forgery the an adversary will almost certainly be an element of $\mathcal{Y} \setminus \mathcal{W}$.

   This allows us to violate the independent pre-image resistance of the hash function $H_{\vec{A}}(\vec{Y}) = \mathsf{e}(\vec{A}, \vec{Y})$, because we can find a pre-image $\vec{W}^*$ of T under $H_{\vec{A}}$, and that pre-image is independent from the set of known vectors $\mathcal{W}$.

4.1.2 Proof

Additional Notation. Let $\mathcal{A}$ be an adversary that breaks the $(0, 0, \rho_M, \rho_S)$-EU-CMA-CML security of our scheme with advantage ε. Let τ be the number of steps of $\mathcal{A}$, and let q_U, q_S, and q_L be the number of update, signing, and leakage queries of $\mathcal{A}$ respectively. Let $pk = (gk, \vec{G}, \mathcal{H}, \vec{A}, T, \vec{Q})$ be a public key and $sk^{[i]} = \vec{W}^{[i]}$ be the i-th updated secret key. Let $M_1, \dots, M_{q_S}$ be the messages queried by $\mathcal{A}$ to the signing oracle, and let (M^*, σ^*) be a message/signature pair forged by $\mathcal{A}$. For a message M, let crs_M be $(\vec{G}, h_{gk}(M, \mathcal{H}))$. That is, crs_M is the CRS which is used to compute $\mathsf{Sig}(sk^{[i]}, M)$ and to verify it.

We denote by $\mathsf{Succ.Game}_i[\mathcal{A}]$ the probability of success of $\mathcal{A}$ in Game i.

Games pertaining to properties of the Waters' function.

Game 0. Game 0 is the experiment of $(0, 0, \rho_M, \rho_S)$-EU-CMA-CML security, played with the adversary $\mathcal{A}$. By definition, we have $\mathsf{Succ.Game}_0[\mathcal{A}] = \varepsilon$.

Game 1. Game 1 proceeds the same as Game 0, except that the public parameters of the hash function $\mathcal{H}$ are generated differently. Specifically, let $\theta = \lceil q_S/\varepsilon \rceil$ and $\mathcal{U} = [-m(\theta - 1)..0] \times [0..\theta - 1]^m$. Then, the public parameters of $\mathcal{H}$ are computed as:

$$\vec{U}, \vec{V} \xleftarrow{\$} \mathbb{H}^2; \quad \vec{u} \leftarrow (u_0, \dots, u_m) \xleftarrow{\$} \mathcal{U}; \quad \vec{v} \leftarrow (v_1, \dots, v_m) \xleftarrow{\$} \mathbb{Z}_p^{m+1}$$
$$\vec{H}_k \leftarrow u_k\vec{U} + v_k\vec{V} \text{ for each } k \in [0..m] \text{ and set } \mathcal{H} \leftarrow (\vec{H}_0, \dots, \vec{H}_m)$$

This change does not change the distribution of $\mathcal{H}$. Hence, we get $\mathsf{Succ.Game}_0[\mathcal{A}] = \mathsf{Succ.Game}_1[\mathcal{A}]$.

We define the functions $K(\vec{u}, M)$ and $L(\vec{v}, M)$ as follows (following [BR09]). Note that K is computed over $\mathbb{Z}$ but L is computed over $\mathbb{Z}_p$.

$$K(\vec{u}, M) := u_0 + \sum_{k\in[m]} u_k M_k \qquad L(\vec{v}, M) := v_0 + \sum_{k\in[m]} v_k M_k \mod p$$

Then, we can rewrite:

$$h_{gk}(M, \mathcal{H}) = K(\vec{u}, M)\vec{U} + L(\vec{u}, M)\vec{V}.$$

Game 2. Game 2 proceeds identically to Game 1, except that the adversary is considered to have won only if she forges a signature, *and* $K(\vec{u}, M_j) \neq 0$ holds for all j, and $K(\vec{u}, M^*) = 0$ holds. Following the proof in [W05] (simplified by [BR09]), we get

Lemma 7. $\mathsf{Succ.Game}_2[\mathcal{A}] = \Omega(\varepsilon^2/mq_S)$

For completeness we give the proof in Appendix A.3.

Game 3. Game 3 proceeds as Game 2, except that the vector $\vec{V}$ is generated differently. Specifically, the challenger chooses $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p$ randomly and sets $\vec{V} \leftarrow \alpha\vec{G}$. From the DDH assumption on $\mathbb{H}$, there exists a polytime adversary $\mathcal{F}$ such that

$$\mathsf{Succ.Game}_3[\mathcal{A}] \geq \mathsf{Succ.Game}_2[\mathcal{A}] + \mathsf{Adv.DDH}_{\mathbb{H}}[\mathcal{F}]. \tag{4.1}$$

If $\mathcal{A}$ wins this game, $K(\vec{u}, M_j) \neq 0$ for all j and $K(\mathcal{H}, M^*) = 0$ holds. The former ensures that $crs_{M_i} = (\vec{G}, h_{gk}(M_i, \mathcal{H}))$ is a hiding CRS because $h_{gk}(M_i, \mathcal{H}) = K(\vec{u}, M_i)\vec{U} + L(\vec{u}, M_i)\vec{V}$ holds and $(\vec{G}, \vec{U})$ is not DDH with overwhelming probability.

The latter ensures that $crs_{M^*} = (\vec{G}, h_{gk}(M^*, \mathcal{H})) = (\vec{G}, L(\vec{u}, M^*)\vec{V}) = (\vec{G}, L(\vec{u}, M^*)\alpha\vec{G})$ is a binding CRS, and the challenger of Game 3 can compute the extraction key $L(\vec{u}, M)\alpha$ of crs_{M^*}. It therefore can extract the witness $\vec{W}^*$ from the signature σ^* forged by $\mathcal{A}$. We will use these properties towards the end of our analysis.

Changing $\vec{W}^{[i]}$ to a Random Element of $\mathcal{W}$.

Game 4. Game 4 is the same game as Game 3, except that the challenger generates the part $(\vec{A}, T, \vec{Q})$ of the public key and the secret keys $(\vec{W}^{[i]})_{i\in[q_U]}$ differently. The challenger computes $(\vec{A}, T, \vec{Q})$ as follows. To compute the public key, randomly choose

$$\begin{gathered}\vec{A} \overset{\$}{\leftarrow} \mathbb{G}^n, \vec{Z}_1, \ldots, \vec{Z}_{n-3} \overset{\$}{\leftarrow} \mathbb{H}^n, \text{ and } \vec{Q} \overset{\$}{\leftarrow} \mathbb{H} \\ \text{satisfying } \mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Z}_k) = 0 \text{ for all } k \in [n-3] \text{ and } \mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Q}) = 0,\end{gathered}$$

Additionally, randomly choose $\vec{Y} \overset{\$}{\leftarrow} \mathbb{H}^n$, and set $T \leftarrow \mathsf{e}(\vec{A}, \vec{Y})$.

To compute the private key, let

$$\mathcal{W} = \vec{Y} + \mathsf{Span}(\vec{Z}_1, \ldots, \vec{Z}_{n-3}).$$

and randomly choose $\vec{W}^{[i]} \overset{\$}{\leftarrow} \mathcal{W}$ for $i \in [0..q_U]$.

Note that the challenger of Game 4 can compute $(\vec{A}, \vec{Z}_1, \ldots, \vec{Z}_{n-3}, \vec{Q})$ in polynomial time by first generating the discrete logarithms of these group elements from the appropriate distribution, and then computing the group elements themselves.

The following lemma is shown in Appendix A.4.

Lemma 8. *There exists a polytime adversary $\mathcal{F}_2$ for the DDH problem satisfying*

$$\mathsf{Succ.Game}_4[\mathcal{A}] \geq \mathsf{Succ.Game}_3[\mathcal{A}] + \frac{1}{n^2}\mathsf{Adv.DDH}_{\mathbb{H}}[\mathcal{F}_2].$$

$\mathcal{W}$ is statistically hidden from the view of $\mathcal{A}$.

Game† 4. Game† 4 is the same as Game 4, except that the updated key $\vec{W}^{[i]}$ of time period i is chosen randomly from $\mathcal{Y}$ instead of $\mathcal{W}$. Here,

$$\mathcal{Y} = \{\vec{W} \in \mathbb{H}^n \mid \mathsf{e}(\vec{A}, \vec{W}) = T\}$$

is the set of valid witnesses for the statement $\mathsf{e}(\vec{A}, \vec{Y}) = T$.

The following lemma shows that Game 4 and Game† 4 are *statistically indistinguishable*. We leave the proof of the lemma until Section 4.2.

Lemma 9. *Let* view_4 *and* $\mathsf{view}_4^\dagger$ *be the views of the adversary $\mathcal{A}$ in Game 4 and Game† 4 respectively, where the view includes randomness of $\mathcal{A}$, public key, and answers to all queries. Then,* $\mathsf{dist}(\mathsf{view}_4, \mathsf{view}_4^\dagger)$ *is negligible, under the assumption that $\mathcal{A}$ wins.*

Bounding the advantage of $\mathcal{A}$ in Game 4. We now show how to use an adversary for Game 4 to break the IPIR of the hash function $H_{\vec{A}}(\vec{Y}) = \mathsf{e}(\vec{A}, \vec{Y})$.

To this end, we construct an adversary $\mathcal{B}$ for the IPIR. $\mathcal{B}$ takes $(\vec{A}^*, T^*, \vec{Y}_1^*, \ldots, \vec{Y}_{n-1}^*)$ an instance of the $(n, n-2)$-IPIR problem for the H defined above.[2] $\mathcal{B}$ then embeds this instance in Game_4 by setting

$$(\vec{A}, T, \vec{Y}) \leftarrow (\vec{A}^*, T^*, \vec{Y}_{n-1}^*),$$

$$\vec{Z}_j \leftarrow \vec{Y}_j^* - \vec{Y}_{n-1}^* \ \text{ for } j \in [n-3]; \qquad \vec{Q} \leftarrow \vec{Y}_{n-2}^* - \vec{Y}_{n-1}^*.$$

Then, $\mathcal{W}$ becomes

$$\mathcal{W} = \vec{Y} + \mathsf{Span}(\vec{Z}_1, \ldots, \vec{Z}_{n-3}) = \mathsf{Aff}(\vec{Y}_1^*, \ldots, \vec{Y}_{n-2}^*).$$

Note that $\mathcal{B}$ can simulate the signing and leakage oracles since she knows the secret key $\vec{W}^{[i]} \overset{\$}{\leftarrow} \mathcal{W}$.

We know that $\mathcal{A}$ wins the game with non-negligible probability and outputs a forgery (M^*, σ^*) such that a witness $\vec{W}^*$ satisfying $\mathsf{e}(\vec{A}, \vec{W}^*) = T$ can be extracted from σ^*.

From Lemma 9, $\mathcal{W}$ is statistically hidden from $\mathcal{A}$. Moreover, no information about $(\vec{Y}_{n-1}^*, \vec{Y}_{n-2}^*)$ is given to $\mathcal{A}$, except for $\vec{Q} = \vec{Y}_{n-2}^* - \vec{Y}_{n-1}^*$. Hence, $\vec{W}^*$ is independent from all of $\vec{Y}_1^*, \ldots, \vec{Y}_{n-2}^*$ with overwhelming probability. However, in this case, by outputting $\vec{W}^*$, $\mathcal{B}$ violates the $(n, n-2)$-IPIR of $H_{\vec{A}}$. Theorem 5 therefore holds.

4.2 Proof of Lemma 9

In this section, we use the notation from Section 4.1.2, and all algorithms are computationally unbounded, unless otherwise stated.

[2]We stress that we are considering an affine subspace of $\mathbb{H}^n$ (rather than linear). Hence, there exist $n-1$ *independent vectors in the* $n-2$ *dimensional affine subspace*. The instance of $(n, n-2)$-independent pre-image resistance contains $n-1$ vectors $\vec{Y}_j^*$.

Leakage resilient random subspace game [BKKV10]. In order to show Lemma 9, we use the following adaptation of the information theoretic argument about leakage from random subspaces from [BKKV10]: let $b \in \{0,1\}$, n, ℓ, q be integers satisfying $n \geq \ell \geq 2$, p be a prime, $\mathcal{P}$ be a finite set, and $\mathcal{K}$ be a n-dimensional vector space over $\mathbb{Z}_p$. We will make use of the following game played with a computationally unbounded adversary $\mathcal{D}$:

1. Let $Z_1, \dots, Z_\ell \overset{\$}{\leftarrow} \mathcal{K}$. Let $\Gamma_i \overset{\$}{\leftarrow} \mathsf{Span}(Z_1, \dots, Z_\ell)$ if $b = 0$ and $\Gamma_i \overset{\$}{\leftarrow} \mathcal{K}$ if $b = 1$.

2. $\mathcal{D}$ adaptively submits (possibly not polynomial time computable) functions $F_i, \dots, F_q : \mathcal{K} \to \mathcal{P}$ as leakage queries, and is given $F_i(\Gamma_i)$ for each i. We stress that each F_i may depend on the previous answers $F_1(\Gamma_1), \dots, F_{i-1}(\Gamma_{i-1})$ from the challenger.

3. Finally, $\mathcal{D}$ outputs a bit b'.

Proposition 10 (Implicitly used in [BKKV10]). *Let $\mathcal{D}$ be an adversary for the above game. Then, for all $\delta \geq 0$, if $|\mathcal{P}| \leq p^{\ell-1}\delta^2/q^2$, then*

$$|\Pr[\mathcal{D} \textit{ outputs } 1|b=0] - \Pr[\mathcal{D} \textit{ outputs } 1|b=1]| \leq \delta.$$

The proof of Proposition 10 is given in Appendix A.5.

Overview. The simplest idea to prove Lemma 9 is sending a function $F(\vec{X}) = f(\vec{Y} + \vec{X}, R_M)$ as a query to the oracle of the game of Proposition 10 with $\ell = n - 3$ and $\mathcal{K} = \{\vec{W} | \mathsf{e}(\vec{A}, \vec{W}) = 0\}$. Here f is the leakage function of a signing query, $\vec{Y}$ is a vector used in Game 4 and Game† 4, and R_M is the randomness used in the signing.

Then, the answer $F(\Gamma_i) = f(\vec{Y} + \Gamma_i)$ from the oracle is equal to the leakge in Game 4 or Game† 4 because $\vec{W}^{[i]} \leftarrow \vec{Y} + \Gamma_i$ is a random element of $\mathcal{W} = \vec{Y} + \mathsf{Span}(Z_1, \dots, Z_{n-3})$ or $\mathcal{Y} = \vec{Y} + \mathcal{K}$ depending on whether $b = 0$ or not. Game 4 and Game† 4 therefore seem to be indistinguishable.

This initial approach fails because a simulator has to generate a Groth-Sahai proof σ_M for the witness $\vec{W}^{[i]} \leftarrow \vec{Y} + \Gamma_i$ also without knowing Γ_i itself.

One approach to overcome this problem is setting a function $F(\vec{X})$ to the concatenation of the Groth-Sahai proof and leakage $f(\vec{Y} + \vec{X}, R_M)$. However, this approach fails also because the bit length of leakage function $|F(\vec{X})|$ beccomes too large for showing EU-CMA-CML security. Note that this is the approach which Brakerski et.al. used to show weaker security of their scheme [BKKV10] than ours.

To overcome this problem, we exploit the perfect witness indistinguishability of the Groth-Sahai proof. This property enables us to generate σ_M using arbitral witness and makes the challenger to find suitable randomness R_M.

In greater detail, let σ_M be a Groth-Sahai proof generated by arbitral witness, and $F(\Gamma_i)$ be the function which computes $\vec{W}^{[i]} \leftarrow \vec{Y} + \Gamma_i$, "reverse engineers" the randomness R_M that would yield the same proof $\sigma_M = \mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W}^{[i]}; R_M)$, (where R_M always exists because of the perfect witness indistinguishability) and outputs $\lambda_M = f(\vec{W}^{[i]}, R_M)$. Then σ_M and λ_M are a Groth-Sahai proof and its leakage using witness $\vec{W}^{[i]}$ respectively. Note that F is not necessarily efficiently computable, because of the definition of game of Proposition 10. Because of the above discussion, we suceeds in overcomming the problem mentioned before.

Additional Notation. In this section, we use the same notations as those in Section 4.1.2.

For an element $(\vec{A}, T)$ of $\mathbb{G}^n \times \mathbb{H}$, a hiding CRS crs, and a witness $\vec{W} \in \mathcal{Y}$, let

$$\mathsf{Prf}[gk, crs, (\vec{A}, T)]$$

be the set of possible outputs of $\mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W})$. Note that the distribution induced on the set $\mathsf{Prf}[gk, crs, (\vec{A}, T)]$ by the prover algorithm of Groth-Sahai does not depend on $\vec{W}$. This is because the Groth-Sahai proof system is perfectly witness indistinguishable.

For $\sigma \in \mathsf{Prf}[gk, crs, (\vec{A}, T)]$ and a witness $\vec{W} \in \mathcal{Y}$, let

$$\mathsf{rnd}(gk, crs, (\vec{A}, T), \sigma, \vec{W})$$

denote the randomness R such that $\sigma = \mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W}; R)$. We will show that rnd is well defined in Appendix A.6. Note that rnd is not necessarily efficiently computable.

The actual proof. We now prove Lemma 9. Let $\mathcal{B}$ be a (possibly unbounded) adversary $\mathcal{B}$ for Game 4 and Game† 4, and suppose that $|\mathsf{Succ.Game}_4[\mathcal{B}] - \mathsf{Succ.Game}_4^\dagger[\mathcal{B}]| > \delta$.

To avoid cumbersome notation, and for simplicity of exposition, we explain how to handle the case when the adversary only makes a single signing query between every two updates (recall that a signing query includes leakage). The general case follows in a straightforward way.

Description of an adversary $\mathcal{D}$. By using $\mathcal{B}$, we construct an adversary $\mathcal{D}$ which has advantage δ in distinguishing the two distributions in Proposition 10 with $\ell = n - 3$ and $\mathcal{K} = \{\vec{W} | \mathsf{e}(\vec{A}, \vec{W}) = 0\}$.

$\mathcal{D}$ generates all the parameters of Game 4 or Game† 4 *except* $(Z_1, \ldots, Z_{n-3})$ and feeds the public key to $\mathcal{B}$. More specifically, $\mathcal{D}$ generates $\vec{Y}$ and $pk = (gk, \vec{G}, \mathcal{H}, \vec{A}, T, \vec{Q})$, where $T = \mathsf{e}(\vec{A}, \vec{Y})$, and gives pk to $\mathcal{B}$. Whenever the adversary $\mathcal{B}$ submits a signing query (M, f), $\mathcal{D}$ first computes crs_M, chooses randomness R_M for the signing algorithm, and generates a signature

$$\sigma_M \xleftarrow{\$} \mathsf{Prf}[gk, crs_M, (\vec{A}, T)].$$

Note that $\mathcal{D}$ generates signatures in the above way regardless of the key updates. $\mathcal{D}$ then constructs a function $\hat{f}$ as follows, sends it as a query to the challenger, obtains an answer λ_M, and sends σ_M and λ_M to $\mathcal{B}$ as a signature and its leakage respectively.

Description of $\hat{f}$, where $(\vec{Y}, gk, crs_M, (\vec{A}, T), \sigma_M)$ is hard-wired in $\hat{f}$.

1. Take input Γ_i.
2. Let $\vec{W}^{[i]} \leftarrow \vec{Y} + \Gamma_i$ and $R'_M \leftarrow \mathsf{rnd}(gk, crs_M, (\vec{A}, T), \sigma_M, \vec{W}^{[i]})$.
3. Output $\lambda_M = f(\vec{W}^{[i]}, R'_M)$.

$\mathcal{D}$ does nothing when $\mathcal{B}$ makes update queries. $\mathcal{D}$ finally outputs 1 iff $\mathcal{B}$ wins.

Correctness of the simulation. We now argue that when Γ_i is selected from $\mathsf{Span}(Z_1, \ldots, Z_{n-3})$ and $\mathcal{K} = \{\vec{W} | \mathsf{e}(\vec{A}, \vec{W}) = 0\}$, $\mathcal{D}$ simulates $\mathcal{B}$ perfectly in Game 4 and Game† 4 respectively.

The main issue seems to be how we generate signatures: in Game 4 and Game† 4, signatures are computed using witness $\vec{W}^{[i]}$ selected from $\mathcal{W}$ and $\mathcal{Y}$ respectively, while in the simulation of $\mathcal{B}$ by $\mathcal{D}$ they are selected randomly from $\mathsf{Prf}[gk, crs, (\vec{A}, T)]$. Note that selecting a proof randomly from $\mathsf{Prf}[gk, crs, (\vec{A}, T)]$ is equivalent to generating a proof given a witness. This is implied by the fact that for each proof and witness there exists a *unique* randomness of the prover that yields this proof.

From the perfect witness indistinguishability of the Groth-Sahai proof system we have that the tuples

$$(\mathsf{Prf}(gk, crs_M, (\vec{A}, T), \vec{V}; R_M), f(\vec{W}^{[i]}, R'_M)) \text{ and}$$
$$(\mathsf{Prf}(gk, crs_M, (\vec{A}, T), \vec{W}^{[i]}; R'_M), f(\vec{W}^{[i]}, R'_M))$$

are identically distributed when R_M is uniformly random. Here, $\vec{W}^{[i]} = \vec{Y} + \Gamma_i$.

In other words, the pair (σ_M, λ_M) produced by $\mathcal{D}$ is a valid (signature, leakage) pair when the secret key is $\vec{W}^{[i]} = \vec{Y} + \Gamma_i$.

If Γ_i is selected from $\mathsf{Span}(Z_1, \ldots, Z_{n-3})$ or $\mathcal{K}$, the secret key $\vec{W}^{[i]} = \vec{Y} + \Gamma_i$ is a random element of $\mathcal{W} = \vec{Y} + \mathsf{Span}(Z_1, \ldots, Z_{n-3})$ or $\mathcal{Y} = \vec{Y} + \mathcal{K}$ respectively. Therefore, the tuples (σ_M, λ_M) given by $\mathcal{D}$ to $\mathcal{B}$ are distributed according to the distribution on signatures and leakage in Game 4 and Game† 4, when Γ_i is selected either $\mathsf{Span}(Z_1, \ldots, Z_{n-3})$ or $\mathcal{K}$ respectively. This contradicts Proposition 10. Lemma 9 therefore holds.

5 Conclusion

In this work, we propose a signature scheme that protected against (1) continual memory leakage combined with (2) all types of continual processing (computational) leakage, namely leakage from key generation, key updates, and signing. Our scheme remains secure even when the leakage during signing is a function $f(sk, r)$ of both the secret key and the randomness.

The security of our scheme is proven in the standard model. Moreover, the amount of information that our scheme is allowed to leak during each time period is optimal, in the sense that our scheme remains secure even if $1 - o(1)$ fraction of the secret key of each time period is leaked.

Recently (in fact, one day after we wrote a preliminary version of this paper), we learned that in a concurrent and independent work, Boyle, Segev, and Wichs [BSW10] proposed a different signature scheme which also tolerates the same amount of leakage as our scheme.

References

[AFGKHO10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, Miyako Ohkubo: Structure-Preserving Signatures and Commitments to Group Elements. CRYPTO 2010: pp.209-236.

[AGV09] Adi Akavia, Shafi Goldwasser, Vinod Vaikuntanathan: Simultaneous Hardcore Bits and Cryptography against Memory Attacks. TCC 2009: 474-495

[ADW09] Joël Alwen, Yevgeniy Dodis, Daniel Wichs: Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. CRYPTO 2009: pp.36-54

[BSW10] Elett Boyle, Gil Segev, Daniel Wichs. Fully Leakage-Resilient Signatures. eprint archive. (2010/488), 2010.

[BB05] David Brumley and Dan Boneh. Remote timing attacks are practical.Computer Networks, 48(5): 2005 pp.701–716,

[BR09] Mihir Bellare, Thomas Ristenpart: Simulation: without the Artificial Abort: Simplified Proof and Improved Concrete Security for Waters' IBE Scheme. EUROCRYPT 2009, pp.407-424.

[BFO08] Alexandra Boldyreva, Serge Fehr, Adam O'Neill: On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles. CRYPTO 2008: pp.335-359

[BSS99] Ian F. Blake, Gadiel Seroussi, Nigel Paul Smart: Elliptic Curves in Cryptography, London Mathematical Society 265, Cambridge University Press, 1999.

[BB04] Dan Boneh, Xavier Boyen: Secure Identity Based Encryption Without Random Oracles. CRYPTO 2004: pp.443-459

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and Leakage Resilient Public-Key Encryption under Subgroup Indistinguishability - (or: Quadratic Residuosity Strikes Back). CRYPTO 2010 pp.1-20.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, Vinod Vaikuntanathan. Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage. FOKS 2010.

[DKL09] Yevgeniy Dodis and Yael Tauman Kalai and Shachar Lovett. On cryptography with auxiliary input. STOC 2009. pp.621-630

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana Lopez-Alt, Daniel Wichs. Cryptography Against Continuous Memory Attacks. FOCS 2010.

[DHLW10b] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana Lopez-Alt, Daniel Wichs. Efficient Public-Key Cryptography in the Presence of Key Leakage. Cryptology ePrint Archive, Report 2010/154.

[DGK+10] Yevgeniy Dodis and Shafi Goldwasser and Yael Tauman Kalai and Chris Peikert and Vinod Vaikuntanathan. Public-Key Encryption Schemes with Auxiliary Inputs. TCC 2010: pp.361-381

[DP08] Stefan Dziembowski, Krzysztof Pietrzak: Leakage-Resilient Cryptography. FOCS 2008: 293-302

[DS05] Yevgeniy Dodis, Adam Smith: Correcting errors without leaking partial information. STOC 2005: pp.654-663

[DP10] Yevgeniy Dodis, Krzysztof Pietrzak: Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. CRYPTO 2010: pp.21-40

[FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, Guy N. Rothblum: Leakage-Resilient Signatures. TCC 2010: pp.343-360

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, Vinod Vaikuntanathan: Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. EUROCRYPT 2010: 135-156

[FS86] Amos Fiat and Adi Shamir: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. CRYPTO 1986: pp. 186-194

[FLS90] Uriel Feige, Dror Lapidot, Adi Shamir: Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract) FOCS 1990: pp.308-317

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, Nigel P. Smart: Pairings for cryptographers. Discrete Applied Mathematics 156(16): pp.3113-3121, 2008

[GR10] Shafi Goldwasser, Guy N. Rothblum: Securing Computation against Continuous Leakage. CRYPTO 2010: pp.59-79

[GSW10] Essam Ghadafi, Nigel P. Smart, Bogdan Warinschi: Groth-Sahai Proofs Revisited. Public Key Cryptography 2010: pp.177-192

[GS08] Jens Groth, Amit Sahai: Efficient Non-interactive Proof Systems for Bilinear Groups. EUROCRYPT 2008: 415-432

[HSH+08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, Edward W. Felten: Lest We Remember: Cold Boot Attacks on Encryption Keys. USENIX Security Symposium 2008: pp.45-60

[JV10] Ali Juma, Yevgeniy Vahlis: Protecting Cryptographic Keys against Continual Leakage. CRYPTO 2010: pp.41-58

[ISW03] Yuval Ishai, Amit Sahai, David Wagner: Private Circuits: Securing Hardware against Probing Attacks. CRYPTO 2003: 463-481

[KJJ98] Paul Kocher and Joshua Jaffe and Benjamin Jun. Introduction to Differential Power Analysis and Related Attacks. 1998. http://www.cryptography.com/dpa/technical/

[KJJ99] Paul C. Kocher and Joshua Jaffe and Benjamin Jun. Differential Power Analysis. CRYPTO 1999: pp.388-397

[KV09] Jonathan Katz, Vinod Vaikuntanathan: Signature Schemes with Bounded Leakage Resilience. ASIACRYPT 2009: pp.703-720

[MR04] Silvio Micali, Leonid Reyzin: Physically Observable Cryptography (Extended Abstract). TCC 2004: 278-296

[NS09] Moni Naor and Gil Segev. Public-Key Cryptosystems Resilient to Key Leakage.CRYPTO 2009. pp18-35.

[P09] Krzysztof Pietrzak: A Leakage-Resilient Mode of Operation. EUROCRYPT 2009: pp.462-482

[QS01] Jean-Jacques Quisquater, David Samyde: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. E-SMART '01: Proceedings of the International Conference on Research in Smart Cards 2001: pp.200–210

[R97] Ronald L. Rivest: All-or-Nothing Encryption and the Package Transform. FSE 1997: pp.210-218

[S79] Adi Shamir: How to Share a Secret. Commun. ACM 22(11): 612-613 (1979)

[SMY09] François-Xavier Standaert and Tal Malkin and Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. EUROCRYPT 2009: pp.443-461

[W05] Brent Waters: Efficient Identity-Based Encryption Without Random Oracles. EUROCRYPT 2005: pp.114-127

A Proof of Lemmata and Propositions

A.1 Proof of Proposition 4

In order to prove Proposition 4, we review the double pairing problem and show that this problem is hard to solve under the DDH assumption.

Proposition 11 (Hardness of Double Pairing Problem on $\mathbb{H}$ under DDH on $\mathbb{G}$ [AFGKHO10])**.** For any polytime adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{l} gk \leftarrow (p, \mathbb{G}, \mathbb{H}, \mathbb{T}) \leftarrow \mathsf{Setup}(1^\kappa), \\ C, D \xleftarrow{\$} \mathbb{G}, (U, V) \leftarrow \mathcal{A}(gk, C, D) \end{array} : \mathsf{e}(C, U) + \mathsf{e}(D, V) = 0, (U, V) \neq 0\right]$$

is negligible under the DDH assumption on $\mathbb{G}$. The problem defined in the above probability is called the *double paring problem.*

Proof of Proposition 11. (idea) Let (C, D, C', D') be an instance of the DDH problem on $\mathbb{G}$. If one can find $(W_1, W_2) \in \mathbb{H}^2$ satisfying $\mathsf{e}(C, W_1) + \mathsf{e}(D, W_2) = 0$, one can know whether (C, D, C', D') is a DDH tuple or not, by checking whether $\mathsf{e}(C, U) + \mathsf{e}(D, V) = 0$ holds or not. □

Proof of Proposition 4. Let n and $k < n - 1$ be natural numbers. Let $\mathcal{A}$ be an adversary for the (n, k)-IPIR of the function $\vec{W} \mapsto \mathsf{e}(\vec{A}^\mathsf{T}, \vec{W})$. By using $\mathcal{A}$ as a subroutine, we construct an adversary $\mathcal{B}$ for the double pairing problem on $\mathbb{H}$.

Description of $\mathcal{B}$. $\mathcal{B}$ takes an instance (C, D) of the double pairing problem on $\mathbb{H}$, selects $\vec{c}, \vec{d} \xleftarrow{\$} \mathbb{Z}_p^n$ and $\vec{Y}_{n-1} \xleftarrow{\$} \mathbb{H}^n$ randomly, and computes

$$P \leftarrow \vec{A} \leftarrow \vec{c}C + \vec{d}D, \quad T \leftarrow \mathsf{e}(\vec{A}^\mathsf{T}, \vec{Y}_{n-1}).$$

$\mathcal{B}$ then takes a random basis $(\vec{Z}_j)_{j \in [n-2]}$ of the kernel of the linear function

$$\phi \ : \ \vec{X} \in \mathbb{H}^n \ \ \mapsto \ \ (\vec{c}^\mathsf{T}\vec{X}, \vec{d}^\mathsf{T}\vec{X}) \in \mathbb{T}^2$$

as follows.

- Select $\vec{z}_1, \dots, \vec{z}_{n-2} \in \mathbb{Z}_p^n$ satisfying
  - $\langle \vec{c}, \vec{z}_j \rangle = \langle \vec{d}, \vec{z}_j \rangle = 0$,
  - $\vec{z}_1, \dots, \vec{z}_{n-2}$ are linearly independent with each other.
- Select $W \xleftarrow{\$} \mathbb{H}^n$ randomly.
- Compute $\vec{Z}_j \leftarrow \vec{z}_j W$ for each $j \in [n-2]$.

$\mathcal{B}$ then computes

$$\vec{Y}_j \leftarrow \vec{Y}_{n-1} + \vec{Z}_j \quad \text{for each } j \in [n-2],$$

feeds $(P, T, \vec{Y}_1, \dots, \vec{Y}_{k+1})$ to $\mathcal{A}$, gets an output $\vec{Y}'$ of $\mathcal{A}$, computes

$$\vec{Z}' \leftarrow \vec{Y}' - \vec{Y}_{n-1}, \quad (U, V) \leftarrow (\vec{c}^\mathsf{T}\vec{Z}', \vec{d}^\mathsf{T}\vec{Z}')$$

and outputs (U, V).

Since $\vec{Z}_j$ is in the kernel of ϕ, it satisfies $\mathsf{e}(\vec{A}, \vec{Z}_j) = \mathsf{e}(\vec{c}C + \vec{d}D, \vec{Z}_j) = 0$. Hence, $\mathsf{e}(\vec{A}, \vec{Y}_j) = \mathsf{e}(\vec{A}, \vec{Y}_{n-1} + \vec{Z}_j) = \mathsf{e}(\vec{A}, \vec{Y}_{n-1}) = T$ holds. This means that $\mathcal{B}$ simulates the view of $\mathcal{A}$ correctly.

Success Probability of $\mathcal{B}$. We show that the output (U, V) of $\mathcal{B}$ is an answer to the instance (C, D) of the double pairing problem with non-negligible probability. Specifically, we show

$$\mathsf{e}(C, U) + \mathsf{e}(D, V) = 0$$

$$(C, D) \neq 0.$$

From the definition of the game of the IPIR, the output $\vec{Y}'$ of $\mathcal{A}$ satisfies the following properties with non-negligible probability:

$$\mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Y}') = T \tag{A.1}$$

$$\vec{Y}' \text{ is linearly independent from } \vec{Y}_1, \ldots, \vec{Y}_{k+1} \tag{A.2}$$

From equation (A.1) and $\mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Y}_{n-1}) = T$, it follows that $\mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Z}') = 0$. Hence, it follows that

$$\mathsf{e}(C, U) + \mathsf{e}(D, V) = \mathsf{e}(\vec{c}^{\mathsf{T}} C, \vec{Z}') + \mathsf{e}(\vec{d}^{\mathsf{T}} D, \vec{Z}') = \mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Z}') = 0.$$

We next show that $(C, D) \neq 0$. Since $\mathcal{B}$ did not feed $(\vec{Y}_{k+2}, \ldots, \vec{Y}_{n-1})$ to $\mathcal{A}$, the distribution of this tuple is independent from the view of $\mathcal{A}$. From (A.2), this means that $\vec{Y}'$ is linearly independent from all of $\vec{Y}_1, \ldots, \vec{Y}_{n-1}$. Therefore, the value $\vec{Z}' = \vec{Y}' - \vec{Y}_{n-1}$ is linearly independent from all of $\vec{Y}_j - \vec{Y}_{n-1} = \vec{Z}_j$ for $j \in [n-2]$.

Recall that the tuple $(\vec{Z}_j)_{j \in [n-2]}$ is the basis of $\ker \phi$. Since $\vec{Z}'$ is linearly independent from all of $\vec{Z}_j$ for $j \in [n-2]$, $\vec{Z}'$ is not in the kernel of ϕ. This means that $(U, V) = (\vec{c}^{\mathsf{T}} \vec{Z}', \vec{d}^{\mathsf{T}} \vec{Z}') = \phi(\vec{Z}')$ is not 0.

From the above discussion, $\mathcal{B}$ succeeds with non-negligible probability. □

A.2 Proof of Lemma 6

We prove Lemma 6 by using the Chernoff bound.

Lemma 12 (Chernoff Bound)**.** *Let $0 \leq p \leq 1$, and let $X_1, \ldots, X_m$ be independent 0-1 random variables, so that $\Pr[X_i = 1] = p$ for each i. Then, for all $\varepsilon > 0$, we have*

$$\Pr\left[\left|\frac{\sum_{i=1}^{m} X_i}{m} - p\right| \leq \varepsilon\right] \geq 1 - 2 \cdot e^{-2m\varepsilon^2}$$

Proof of Lemma 6. Suppose that there is an adversary $\mathcal{A}$ that has advantage ε_A in breaking $(0, 0, \rho_M, \rho_S)$-EU-CMA-CML security of the signature scheme $\mathcal{SGN}$. To simplify the notation, we recognize the key generation as "the 0-th key update".

By using $\mathcal{A}$ as a subroutine, we construct an adversary $\mathcal{B}$ which can break $(\rho_G, \rho_U, \rho_M, \rho_S)$-EU-CMA-CML security of $\mathcal{SGN}$, where

$$\rho_G = \rho_U = \frac{c \cdot \log k}{n \log p}.$$

Idea Behind the Construction of $\mathcal{B}$. Before giving the details, we give the idea behind the construction of $\mathcal{B}$.

The main problem is the simulation of answering the leakages in (the 0-th or later) key updates: $\mathcal{B}$ has to answer to $\mathcal{A}$ leakage $f(sk^{[i]}, r)$ in updates, although $\mathcal{B}$ herself cannot get these leakage from the challenger.

We overcome this problem by defining a function F_{st} which "guesses" the leakage α. $\mathcal{B}$ then sends F_{st} to the challenger as a (memory) leakage query, and passes the answer $F_{st}(sk^{[i]})$ to $\mathcal{A}$ as the leakage of updates.

The description of $F_{st}(sk^{[i]})$ is as follows. Let $\bar{\mathcal{A}}$ be the copy of the description of $\mathcal{A}$ and st be the current state of $\mathcal{A}$. For any $\alpha \in \{0,1\}^{c \log \kappa}$, $F_{st}(sk^{[i]})$ executes $\bar{\mathcal{A}}(st)$ as a subroutine, after giving α to $\bar{\mathcal{A}}$ as the leakage of update. Above, the descrption of $\bar{\mathcal{A}}$ and the state st are hard-wired in $F_{st}(sk^{[i]})$ by $\mathcal{B}$. Note that $F_{st}(sk^{[i]})$ can answer to oracle queries because it knows $sk^{[i]}$ as an input.

By doing the above execution a lot of times, $F_{st}(sk^{[i]})$ guesses α which maximizes the success probability of $\mathcal{A}$. $F_{st}(sk^{[i]})$ finally outputs this α. Note that $F_{st}(sk^{[i]})$ can do above procedures in polynomial time because the amount of update leakages are $O(\log n)$ in our case and therefore only polynomial number of candidate α of leakage are there.

Detailed Description of F_{st}. We first give the detailed description F_{st} for the i-th update for $i \geq 1$. Bellow, the copy $\bar{\mathcal{A}}$ of the description of $\mathcal{A}$ and the current state st of $\mathcal{A}$ are hard-wired in F_{st} by $\mathcal{B}$:

1. Take sk as an input.

2. For every candidate of leakage value $\alpha \in \{0,1\}^{c \log \kappa}$, repeat the following procedure κ times, each time rewinding $\bar{\mathcal{A}}$ to state st:

   (a) Give α to $\bar{\mathcal{A}}(st)$ as a leakage in the key update.

   (b) Simulate $\bar{\mathcal{A}}$ to completion, answering all queries correctly, and randomly updating the secret key.

   Note that since F_{st} has sk as input, it can sign, compute leakage, and randomize sk inside its computation and can therefore answer all queries asked by $\bar{\mathcal{A}}$.

3. Compute the fraction of times $\bar{\delta}_\alpha$ that the above procedure concluded with A producing a successfully forged signature.

4. Output α which maximize $\bar{\delta}_\alpha$. (If there are two or more such α, output any one of them.)

The function F_{st} for the key generation, which we will write F'_{st} due to differentiate the function for other updates, is the same as the above one, except that F'_{st} inputs to $\bar{\mathcal{A}}$ not only α but also the public key pk. (pk is hard-wired in it by $\mathcal{B}$.)

Detailed Description of $\mathcal{B}$. The description of $\mathcal{B}$ is as follows.

1. Take pk as an input.

2. Execute $\mathcal{A}$ until it sends a leakage function of the key generation as a query.

3. Copy the current state st of $\mathcal{A}$, make query F'_{st}, and pass the answer to $\mathcal{A}$.

4. Execute $\mathcal{A}$ until it halts, answering queries as follows.

   (a) If $\mathcal{A}$ makes a signing or a leakage query, pass it to the challenger, get an answer, and pass the answer to $\mathcal{A}$.

   (b) If $\mathcal{A}$ makes an update query with leakage function f, make update query with no leakage function, copy the current state st of $\mathcal{A}$, sends F_{st} to the challenger as a memory leakage query, and pass the answer to $\mathcal{A}$.

5. If $\mathcal{A}$ succeeds in outputting a forgery, output this forgery. Otherwise, output $\perp$.

Success Probability of $\mathcal{B}$. To simplify the analysis, we make the challenger compute all $sk^{[i]}$ for $i \in [0..q_U]$ at the beginning of game. This change enables us to avoid cumbersome discussion about the dependency of random valiables. Note that this change clearly do not effect the success probability of $\mathcal{B}$.

Let

$$\eta \leftarrow \frac{\varepsilon_A}{2(q_u+1)}.$$

Let st_{i-1} be a state of $\mathcal{A}$ just after making the i-th update query in the game with $\mathcal{B}$ and before getting the answer to the query.

In order to clarify the number i of times of updates, we let $\bar{\delta}_\alpha^{(i)}$ denote $\bar{\delta}_\alpha$.

The function $F_{st_{i-1}}$ computes the estimate $\bar{\delta}_\alpha^{(i)}$ of the probability $\delta_\alpha^{(i)}$ that $\bar{A}$ will win when she gets α as a leakage of the i-th key update.

The Chernoff bound implies that

$$\Pr\left[\left|\bar{\delta}_\alpha^{(i)} - \delta_\alpha^{(i)}\right| < \eta\right] \leq 1 - 2 \cdot e^{-2\kappa\eta^2} = 1 - \mathrm{neg}(\kappa). \tag{A.3}$$

We estimate $\max_\alpha \delta_\alpha^{(i)}$ and $\bar{\delta}_\alpha^{(i)}$ recurively as follows. Suppose that

$$\max_\alpha \delta_\alpha^{(i)} \geq \varepsilon_A - i\eta \tag{A.4}$$

holds. Recall that $F_{st_{i-1}}$ outputs α which maximizes $\bar{\delta}_\alpha^{(i)}$. From equation (A.3), $F_{st_{i-1}}$ outputs α satisfying

$$\bar{\delta}_\alpha^{(i)} \geq \varepsilon_A - (i+1)\eta \tag{A.5}$$

with overwhelming probability. Since $\mathcal{A}$ gets the output α of $F_{st_{i-1}}$ as the leakage of the i-th update, equation (A.5) implies that the adverasary $\mathcal{A}$ wins with probability $\varepsilon_A - (i+1)\eta$, even after getting α as the leakage of the i-th update.

This means that, if $\mathcal{A}$ gets the "correct" leakage in the $(i+1)$-th update, $\mathcal{A}$ wins with probability $\varepsilon_A - (i+1)\eta$ even after getting the leakage of the $(i+1)$-th update. In other word, there exists an answer β to the $(i+1)$-th update query satisfying $\delta_\beta^{(i+1)} \geq \varepsilon_A - (i+1)\eta$. In particular,

$$\max_\alpha \delta_\alpha^{(i+1)} \geq \varepsilon_A - (i+1)\eta \tag{A.6}$$

holds. The equation (A.6) enables us to estimate $\bar{\delta}_\alpha^{(i+1)}$, by the same way that the equation (A.4) enables us to estimate $\bar{\delta}_\alpha^{(i)}$.

By using the above estimation $q_U + 1$ times, we can get the estimate

$$\bar{\delta}_\alpha^{(q_U)} \geq \varepsilon_A - (q_U+1)\eta = \varepsilon_A/2.$$

From the definition of $\delta_\alpha^{(i)}$ and $\mathcal{B}$ it follows that

$$\bar{\delta}_\alpha^{(q_U)} = \Pr[\mathcal{B} \text{ wins}].$$

Hence, $\mathcal{B}$ wins with probability $\varepsilon_A - (q_U+1)\eta = \varepsilon_A/2$, which is non-negligible. This contradicts the assumption. $\square$

A.3 Proof of Lemma 7

This part of proof is essentially same as the simplified security proof [BR09] of the Waters' ID-based encryption [W05]. Our proof however is even more simpler than [BR09], because our usage of Waters' hash function enables us to skip the following part of discussion of [BR09]: In [BR09], Bellare and Ristenpart showed that that the advantage of an adversary does not depend on when the challenger checks equalities $K(\vec{u}, M_j) \neq 0$. In our case, however, it is clear that the advantage does not depend on it.

Proof of Lemma 7. Let $\mathcal{A}$ be an adversary of Game_2. Let q_S be the number of signing queries of $\mathcal{A}$, M_j be the j-th signing query of $\mathcal{A}$, and (M_*, σ_*) be a forgery of $\mathcal{A}$. To simplify the notation we write $\vec{M} = (M_*, M_1, \ldots, M_{q_S})$.

Let GD be the event that, at the end of Game_2, both $K(\vec{u}, M_j) \neq 0$ for all j and $K(\vec{u}, M_*) = 0$ holds. Let BD be the event that GD does not occur.

For any vector $\vec{M}' = (M'_*, M'_1, \ldots, M'_{q_S})$, let

$$\gamma(\vec{M}') = \Pr[K(\vec{u}, M'_*) = 0, K(\vec{u}, M_1) \neq 0, \ldots, K(\vec{u}, M'_{q_S}) \neq 0].$$

Here the probability is taken over the choice of $\vec{u} \overset{\$}{\leftarrow} \mathcal{U}$. This is probability of GD under the particular sequence of the signing queries $M'_1, \ldots, M'_{q_S}$ and the output message M'_*.

We use the following two lemmata for proving Lemma 7.

Lemma 13 ([BR09])**.** *Let m, θ, and q_S be parameters which is used to define Waters' function. Then for any $\vec{M}' = (M'_*, M'_1, \ldots, M'_{q_S}) \in (\{0,1\}^m)^{q_S+1}$, it follows that*

$$\frac{1}{m(\theta-1)+1}\left(1 - \frac{q_S}{\theta}\right) \leq \gamma(\vec{M}') \leq \frac{1}{m(\theta-1)+1}.$$

□

Lemma 14 ([BR09])**.** *Let $\vec{M}' = (M'_*, M'_1, \ldots, M'_{q_S})$ be a tuple of messages. Let $\mathsf{Q}(\vec{M}')$ be event that $\mathcal{A}$ in Game_2 results in making queries $M'_1, \ldots, M'_{q_S}$ and outputting M'_*. Then, it follows that*

- $\Pr[\mathsf{GD} \wedge \mathsf{Q}(\vec{M}')] = \gamma(\vec{M}') \Pr[\mathsf{Q}(\vec{M}')]$. □

The above two lemmata are proved in [BR09] based only on the computations of probabilities, using the game as a blackbox. These lemmata therefore holds even for our games.

Let F be the event that the forgery of $\mathcal{A}$ is accepted by the verification algorithm. Because the distribution of the value $\vec{u}$ is independent from that of the view of $\mathcal{A}$, the success probability of forging does not depend whether GD holds or not. Hence, it follows that

$$\Pr[\mathsf{F} \mid \mathsf{GD}] = \Pr[\mathsf{F}]$$

That is,

$$\Pr[\mathsf{F} \wedge \mathsf{GD}] = \Pr[\mathsf{F}] \Pr[\mathsf{GD}].$$

From the above arguments, it follows that

$$\begin{aligned}
\mathsf{Succ.Game}_2[\mathcal{A}] &= \Pr[\mathsf{F} \wedge \mathsf{GD}] \\
&= \Pr[\mathsf{F}] \cdot \Pr[\mathsf{GD}] \\
&= \mathsf{Succ.Game}_1[\mathcal{A}] \cdot \sum_{\vec{M}'} \Pr[\mathsf{GD} \wedge \mathsf{Q}(\vec{M}')] \\
&= \varepsilon \cdot \sum_{\vec{M}'} \left(\gamma(\vec{M}') \Pr[\mathsf{Q}(\vec{M}')] \right) \\
&\geq \varepsilon \cdot \min_{\vec{M}'} \{\gamma(\vec{M}')\} \cdot \sum_{\vec{M}'} \cdot \Pr[\mathsf{Q}(\vec{M}')] \\
&\geq \varepsilon \cdot \frac{1}{m(\theta - 1) + 1} \left(1 - \frac{q_S}{\theta}\right) \cdot 1.
\end{aligned}$$

Since we set $\theta = \lceil q_S/\varepsilon \rceil$, the light hand side of the above inequality is not smaller than

$$\text{Right Hand Side} \geq \varepsilon \cdot \frac{1}{m((q_S/\varepsilon) - 1) + 1}(1 - \varepsilon) \geq \frac{\varepsilon^2(1 - \varepsilon)}{m(q_S - \varepsilon)} = \Omega(\varepsilon^2/mq_S).$$

□

A.4 Proof of Lemma 8

Lemma 15. *Let n be a natural number and $\mathcal{B}$ be an adversary. Define game $\mathsf{Game}[\mathcal{B}]$ as follows: The challenger takes*

$$\vec{Z}'_{n-2} \xleftarrow{\$} \mathbb{H}^{n-1}, \quad \nu_1, \ldots, \nu_{n-3} \xleftarrow{\$} \mathbb{Z}_p$$

randomly, sets

$$(Z'_1, \ldots, Z'_{n-3}) \leftarrow (\nu_1 \vec{Z}'_{n-2}, \ldots, \nu_{n-3} \vec{Z}'_{n-2}).$$

and gives $(Z'_1, \ldots, Z'_{n-3}, Z'_{n-2})$ or a randomly selected element of $(\mathbb{H}^{n-1})^{n-2}$ to $\mathcal{N}$. $\mathcal{B}$ has to guess whether the input is random one or not. Then, for any adversary $\mathcal{B}$, there exists an adversary $\mathcal{F}$ satisfying

$$\mathsf{Adv.Game}[\mathcal{B}] \geq \frac{1}{n^2} \mathsf{Adv.DDH}_{\mathbb{H}}[\mathcal{F}].$$

We omit the proof of Lemma 15 because it is well-known.

Proof of Lemma 8. By using an adversary $\mathcal{A}$ of Game_3, we construct an adversary $\mathcal{B}$ for the game of Lemma 15.

Descrption of Simulator. $\mathcal{B}$ takes a group description $gk = (p, \mathbb{G}, \mathbb{H}, \mathbb{T}, \mathsf{e})$ and an instance $(\vec{Z}'_1, \ldots, \vec{Z}'_{n-2}) \in (\mathbb{H}^{n-1})^{n-2}$ of the game of Lemma 15 as an input. She then generates matrix $Z = (\vec{Z}_1, \ldots, \vec{Z}_{n-3}) \in \mathbb{H}^{n \times (n-3)}$ and vectors $\vec{A} \in \mathbb{G}^n$ and $\vec{Q} \in \mathbb{H}^n$ satisfying

$$\mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Z}_1) = \cdots = \mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Z}_{n-2}) = \mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Q}) = 0,$$

$$\exists X_1, \ldots, X_{n-2} \in \mathbb{H}^n \ : \ \vec{Z}_\ell = \vec{Z}'_\ell || X_\ell, \ \ \vec{Q} = \vec{Z}'_{n-2} || X_{n-2}.$$

Specifically, it generates $Z = (\vec{Z}_1, \ldots, \vec{Z}_{n-3}) \in \mathbb{H}^{n \times (n-3)}$, $\vec{A}$, and $\vec{Q}$ as follows.

- Take $A_n \xleftarrow{\$} \mathbb{G}$ and $a_1, \ldots, a_{n-1} \xleftarrow{\$} \mathbb{Z}_p$ randomly and set
$$\vec{A} \leftarrow (a_1 A_n, \ldots, a_{n-1} A_n, A_n)^{\mathsf{T}}.$$
- For each $\ell \in [n-2]$, parse $\vec{Z}'_\ell$ as $(Z_{\ell,1}, \ldots, Z_{\ell,n-1})$ and set
$$\vec{Z}_\ell \leftarrow (Z_{\ell,1}, \ldots, Z_{\ell,n-1}, -\sum_{k \in [n-1]} a_k Z_{\ell,k})^{\mathsf{T}}.$$
- Set
$$Z \leftarrow (\vec{Z}_1, \ldots, \vec{Z}_{n-3}); \quad Q \leftarrow \vec{Z}_{n-2}.$$

$\mathcal{B}$ then randomly takes
$$\vec{Y}' \xleftarrow{\$} \mathbb{H}^n; \qquad sk^{[i]} \leftarrow \vec{W}^{[i]} \xleftarrow{\$} \vec{Y}' + \mathsf{Span}(Z_1, \ldots, Z_{n-3}) \text{ for each } i \in [0..q_U].$$
and computes
$$T \leftarrow \mathsf{e}(\vec{A}^{\mathsf{T}}, \vec{Y}').$$

$\mathcal{B}$ generates the other parts $(\vec{G}, \mathcal{H})$ of the public key than $(gk, \vec{A}, T)$, by using the same way as the challenger of Game_4 does. It then sets $pk \leftarrow (gk, \vec{G}, \mathcal{H}, \vec{A}, T, \vec{Q})$. and feeds pk to $\mathcal{A}$.

If $\mathcal{A}$ makes query (update, f), $\mathcal{B}$ will use $\vec{W}^{[i+1]}$ as the secret key. It resets $i \leftarrow i+1$. $\mathcal{B}$ simulates signing queries and leakage queries by using $\vec{W}^{[i]}$ as the current secret key. $\mathcal{B}$ finally outputs 1 or 0, depending on whether $\mathcal{A}$ wins the game or not.

Correctness of Simulation. If the instance $I = (\vec{Z}'_1, \ldots, \vec{Z}'_{n-2})$ is randomly selected elements of $(\mathbb{H}^{n-1})^{n-2}$, the view of $\mathcal{A}$ simulated by $\mathcal{B}$ is clearly the same as that in Game_5. In contrast, if the instance I satisfies $\vec{Z}'_\ell = \nu_\ell \vec{Z}'_{n-2}$ for some $(\nu_\ell)_{\ell \in [n-3]}$, one can easily show that
$$\vec{Z}_\ell = \mu_\ell \vec{Z}_{n-2} = \mu_\ell Q$$
holds for any $\ell \in [n-3]$.

Then, for $i \geq 1$,
$$\vec{W}^{[i]} \in \vec{Y}' + \mathsf{Span}(Z_1, \ldots, Z_{n-3}) = \vec{Y}' + \mathsf{Span}(\vec{Q})$$
holds. Hence, there exists $u^{[i]} \in \mathbb{Z}_p$ satisfying
$$\vec{W}^{[i]} = \vec{Y}' + u^{[i]} \vec{Q}.$$
Let
$$\vec{Y} \leftarrow \vec{Y}' + \bar{u}^{[0]} \vec{Q}, \quad \hat{s}^{[i]} \leftarrow u^{[i]} - u^{[i-1]} \bmod p$$
Then, it follows that
$$\vec{W}^{[0]} = \vec{Y}, \quad \vec{W}^{[i+1]} = \vec{W}^{[i]} + s^{[i]} \vec{Q}$$
That is, the distribution of $\vec{W}^{[i+1]}$ generated by $\mathcal{B}$ is the same as that of $\vec{W}^{[i+1]}$ in Game_4. Hence, the view of $\mathcal{A}$ simulated by $\mathcal{B}$ is clearly the same as that in Game_4. From the above discussion, the lemma holds. $\square$

A.5 Proof of Proposition 10

The following proposition was explicitly given by [BKKV10] and was proven by using "generalized crooked leftover hash lemma" [DS05, BFO08]. We will prove Proposition 10 based on it.

Proposition 16 ([BKKV10])**.** *Let n, ℓ be natural numbers satisfying $n \geq \ell \geq 2$, p be a prime, $\mathcal{P}$ be a finite set, $\mathcal{K}$ be a n-dimensional vector space over $\mathbb{Z}_p$, and $F : \mathcal{K} \to \mathcal{P}$ be a function. Then for randomly taken $Z \leftarrow (\vec{Z}_1, \dots, \vec{Z}_\ell) \overset{\$}{\leftarrow} \mathcal{K}^\ell$, $\vec{V} \overset{\$}{\leftarrow} \mathsf{Span}(\vec{Z}_1, \dots, \vec{Z}_\ell)$, and $\vec{U} \overset{\$}{\leftarrow} \mathcal{K}$,*

$$\mathsf{dist}((Z, F(\vec{V})), (Z, F(\vec{U}))) \leq \delta$$

holds as long as

$$|\mathcal{P}| \leq p^{\ell-1}\delta^2.$$

Proof of Proposition 10. We prove Proposition 10 using a hybrid argument. Specifically, let Hyb_i be the same as the game of Proposition 10 except that, regardless of the value of b, the challenger takes Γ_j from $\mathcal{K}$ for $1 \leq j \leq i-1$, and from $\mathsf{Span}(Z_1, \dots, Z_{n-3})$ for $i \leq j \leq q$. In order to differenciate the notation, we let $\vec{U}_j$ and $\vec{V}_j$ denote randomly selected element of $\mathcal{K}$ or $\mathsf{Span}(Z_1, \dots, Z_{n-3})$.

Let $\mathcal{D}$ be a distinguisher for the game Game of Proposition 10. We next show that for all $1 \leq i \leq q-1$, the outputs of a distinguisher $\mathcal{D}$ in Hyb_i and Hyb_{i+1} are closely distributed.

Let δ be the gap between the probability that $\mathcal{D}$ output 1 in Game if $b = 0$, and the probability that it outputs 1 in Game if $b = 1$. Let us denote by δ_i the gap between the probabilities of $\mathcal{D}$ outputting 1 in Hyb_i and Hyb_{i+1}. We know that there exists an $1 \leq i \leq q-1$ such that $\delta_i \geq \delta/q$.

For convenience of notation, let us remove $\mathcal{D}$ from the argument, and instead encode its adaptivity logic into the leakage functions. Then, the tuple of the answers in Game if $b = 1$ and $b = 0$ respectively can be written as

$$\begin{aligned}&(F'_1(\vec{V}_1), F'_2(\lambda_1, \vec{V}_2), \dots, F'_q(\lambda_1, \dots, \lambda_{q-1}, \vec{V}_q)) \text{ and}\\ &(F'_1(\vec{U}_1), F'_2(\lambda_1, \vec{U}_2), \dots, F'_q(\lambda_1, \dots, \lambda_{q-1}, \vec{U}_q))\end{aligned}$$

where λ_i is the output of F'_i (note that the above definition is recursive).

From the hyblid argument, we therefore get the distributions

$$\begin{aligned}D_i &= (F'_1(\vec{U}_1), \dots, F'_{i-1}(\lambda_1, \dots, \lambda_{i-2}, \vec{U}_{i-1}), F'_i(\lambda_1, \dots, \lambda_{i-1}, \vec{V}_i), \dots, F'_q(\lambda_1, \dots, \lambda_{q-1}, \vec{V}_{q-1})) \text{ and}\\ D_{i+1} &= (F'_1(\vec{U}_1), \dots, F'_i(\lambda_1, \dots, \lambda_{i-2}, \vec{U}_i), F'_{i+1}(\lambda_1, \dots, \lambda_i, \vec{V}_{i+1}), \dots, F'_q(\lambda_1, \dots, \lambda_{q-1}, \vec{V}_{q-1}))\end{aligned}$$

satisfying $\mathsf{dist}(D_i, D_{i+1}) \geq \delta/q$.

Let $Z = (\vec{Z}_1, \dots, \vec{Z}_\ell)$ be a randomly selected element of $\mathcal{K}^\ell$ and Γ_i be an element selected from either $\mathsf{Span}(Z_1, \dots, Z_\ell)$ or $\mathcal{K} = \{\vec{W} | \mathsf{e}(\vec{A}, \vec{W}) = 0\}$. We define the distrbution $D'(\Gamma_i)$ as follows:

- Choose $\vec{U}_1, \dots, \vec{U}_{i-1} \overset{\$}{\leftarrow} \mathcal{K}$.
- For $j = 1, \dots, i-1$, recursively compute the values $\lambda_j \leftarrow F'_{j-1}(\lambda_1, \dots, \lambda_{j-1}, \vec{U}_{j-1})$.
- Let $F(\cdot) = F_i(\lambda_1, \dots, \lambda_{i-1}, \cdot)$ be the function that we will use to distinguish the distributions of Proposition 16, and λ_i be $F(\Gamma_i)$.
- For $j = i+1, \dots, q$, recursively compute the values $\lambda_j \leftarrow F'_{j-1}(\lambda_1, \dots, \lambda_{j-1}, \vec{V}_{j-1})$.
- Let $D'(\Gamma_i)$ be $(\lambda_1, \dots, \lambda_q)$.

The distribution $D'(\Gamma_i)$ is the same as D_i or D_{i+1} depending on wether Γ_i is selected from $\mathsf{Span}(Z_1, \dots, Z_\ell)$ and $\mathcal{K} = \{\vec{W} | \mathsf{e}(\vec{A}, \vec{W}) = 0\}$. From Proposition 16, this means that $\mathsf{dist}(D_i, D_{i+1})$ should be less than δ_i. This contradicts $\mathsf{dist}(D_i, D_{i+1}) \geq \delta_i$. Proposition 10 therefore holds. □

A.6 Proof of Well-Definedness of rnd

Lemma 17. *For all $(\vec{A}, T) \in \mathbb{G}^n \times \mathbb{H}$, all hiding CRS crs, all witnesses $\vec{W} \in \mathcal{Y}$, and all possible output σ of $\mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W})$, there exists one and only one element R of the set of randomness satisfying $\sigma = \mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W}; R)$.*

Proof of Lemma 17. We take $(\vec{A}, T)$, σ, and $\vec{W}$ as in the statement of the proposition. Because of the perfect witness indistinguishability, there exists at least one random tape R satisfying $\sigma = \mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W}; R)$.

Parse σ as $(\vec{C}, \vec{D}, \Pi)$. From the definition of Prf, $(\vec{C}, \vec{D}) = (R \cdot \vec{G}, \vec{W} + R \cdot \vec{H})$ holds. When we write down the above equality by each components, we can consider it as the a system of linear equations for the coefficients of R. This system has $2n$ equations for $2n$ variables. Hence, it has only one solution at most. This means that there exists one and only one R satisfying $\sigma = \mathsf{Prf}(gk, crs, (\vec{A}, T), \vec{W}; R)$. $\square$