Optimal Security Reductions for Unique Signatures: Bypassing Impossibilities with A Counterexample

Fuchun Guo[1], Rongmao Chen[2], Willy Susilo[1], Jianchang Lai[1],
Guomin Yang[1], and Yi Mu[1]

[1] Institute of Cybersecurity and Cryptology
School of Computing and Information Technology
University of Wollongong, Wollongong, NSW 2522, Australia
{fuchun,wsusilo,jl967,gyang,ymu}@uow.edu.au
[2] College of Computer, National University of Defense Technology, Changsha, China
chromao@nudt.edu.cn

Abstract. Optimal security reductions for unique signatures (Coron, Eurocrypt 2002) and their generalization, i.e., efficiently re-randomizable signatures (Hofheinz *et al.*, PKC 2012 & Bader *et al.*, Eurocrypt 2016) have been well studied in the literature. Particularly, it has been shown that under a non-interactive hard assumption, any security reduction (with or without random oracles) for a unique signature scheme or an efficiently re-randomizable signature scheme must loose a factor of at least q_s in the security model of existential unforgeability against chosen-message attacks (EU-CMA), where q_s denotes the number of signature queries. Note that the number q_s can be as large as 2^{30} in practice. All unique signature schemes and efficiently re-randomizable signature schemes are concluded to be accompanied with loose reductions from these impossibility results.

Somewhat surprisingly, in contrast to previous impossibility results (Coron, Eurocrypt 2002; Hofheinz *et al.*, PKC 2012; Bader *et al.*, Eurocrypt 2016), in this work we show that without changing the assumption type and security model, it is not always the case that any security reduction must loose a factor of at least q_s. As a counterexample, we propose a unique signature scheme with a tight reduction in the EU-CMA security model under the Computational Diffie-Hellman (CDH) assumption. Precisely, in the random oracle model, we can program a security reduction with a loss factor of at most $nq^{1/n}$, where n can be any integer independent of the security parameter for the scheme construction and q is the number of hash queries to random oracles. The loss factor in our reduction can be very small. Considering $n = 25$ and $q = 2^{50}$ as an example, the loss factor is of at most $nq^{1/n} = 100$ and therefore our security reduction is tight.

Notice that the previous impossibility results are derived from proofs via a so-called meta-reduction technique. We stress that instead of indicating any flaw in their meta-reduction proofs, our counterexample merely demonstrates that their given meta-reduction proofs fail to capture all security reductions. More precisely, we adopt a reduction called *query-based reduction*, where the reduction uses a hash query from the adversary to solve an underlying hard problem. We show that the meta-reduction proofs break down in our query-based reduction. The query-based reduction is not a new notion and it has been adopted for encryption proofs, but this work is the first seminal approach for applying query-based reduction in digital signatures.

The given counterexample in this work is of an independent interest as it implies a generic way of constructing a digital signature scheme (including unique signatures) with a tight reduction in the random oracle model from a digital signature scheme with a loose reduction. Although our proposed methodology is somewhat impractical due to the inefficiency of signature length, it introduces a completely new approach for tight proofs that is different from traditional approaches using a random salt.

Keywords. Unique Signatures; Tight Reduction; Impossibility; Counterexample

1 Introduction

Security reduction is the fundamental method for proving provable security in cryptosystems. In a security reduction, if there exists an efficient adversary who can break a newly proposed scheme, we prove that a generally believed-to-be-hard mathematical problem can be efficiently solvable with the help of this adversary. This, however, contradicts with the hardness assumption and hence we conclude that the proposed scheme is secure. Suppose the adversary can break a digital signature scheme in t polynomial time with a non-negligible probability ϵ. Generally speaking, a security reduction will solve an underlying hard problem in $t+T$ time with probability $\frac{\epsilon}{L}$. Here T refers to the time cost of reduction while L refers to the loss factor (or security loss), which means the success probability of reduction is $\frac{1}{L}$ only. When the loss factor is linear in the number of signature queries denoted by q_s from the adversary, the security reduction is said to be a loose reduction because the number of signature queries can be as larger as 2^{30}. When the loss factor L is constant and small, the security reduction is said to be a tight reduction. In concrete security [4, 17], a tight reduction is essential for constructing an efficient scheme without increasing the security parameter to compensate the loss factor.

A significant number of digital signature schemes have been proposed in the literature. Many of them, such as [17, 11, 9, 5, 19, 14, 2, 6], are provably secure with tight reductions. In this work, a security reduction for a signature scheme refers to a reduction from a non-interactive computational hard problem under the security model of existential unforgeability against chosen-message attacks (EU-CMA) [12]. In a security reduction, we say a signature is simulatable if the signature can be simulated by the simulator without knowing the corresponding secret (signing) key and a signature is reducible if the signature can be reduced to solve a hard problem. The difficulty of achieving a tight reduction in the literature is due to the fact that we need to program the reduction with a high success probability without knowing which messages whose signatures should be programmed as simulatable and which messages whose signatures should be programmed as reducible. To overcome this difficulty, all known signature schemes with tight reductions to date use a random salt (number) in the signature generation such that a signature of a given message can be either simulatable or reducible depending on the choice of the random salt. The simulator relies on this functionality to obtain a high success probability of reduction, especially without abortion during signature queries.

Unique signatures are one special type of digital signatures which do not use any random salt in the signature generation. The signature for each message therefore is unique. As a consequence, we cannot switch the functionality (simulatable or reducible) of each unique signature in the security reduction. Therefore, it seems any security reduction for a unique signature scheme cannot be tight. In fact, optimal security reductions for unique signatures or their generalization have been studied in [10, 15, 3]. They showed that it is impossible to program a better security reduction with a loss factor of less than q_s for a unique signature scheme or an efficiently re-randomizable signature scheme, where q_s is the number of signature queries. To be precise, in Eurocrypt 2002, Coron [10] described a meta-reduction proof to prove the impossibility of tight reductions for certain digital signatures including unique signatures. Coron claimed the success probability of any security reduction for a unique signature scheme cannot substantially exceed $\epsilon_{\mathcal{F}}/(eq_s)$. That is, the loss factor is of at least $e \cdot q_s$. Here e is the base of the natural logarithm and $\epsilon_{\mathcal{F}}$ is the adversary's advantage (probability) of forging a valid signature in the EU-CMA security model. Ten years later, in Eurocrypt 2012, Kakvi and Kiltz [16] found and fixed a subtle technical flaw in the Coron's impossibility result. They showed the optimal security reduction for the RSA-FDH (Full Domain Hash) signature scheme cannot be guaranteed, if public keys are not certified. In PKC 2012, Hofheinz, Jager and Knapp [15] extended the meta-reduction proof for a slightly more general signature notion, namely efficiently re-randomizable signatures. They proved and claimed that any black-box security reduction for a signature scheme with efficiently re-randomizable signatures must have a reduction loss of at least $\Omega(q_s)$; otherwise, the underlying hardness assumption is false. In Eurocrypt 2016, Bader, Jager, Li and Schäge [3] improved the impossibility result of [10, 16, 15] with simpler proofs and bound, where the minimum loss factor is q_s. Particularly, they introduced the notion of efficiently re-randomizable relations to overcome the subtle issue discovered by Kakvi and Kiltz [16] and further generalized the notion of efficiently re-randomizable signatures [15].

In short, the aforementioned impossibility results (Coron, Eurocrypt 2002; Hofheinz *et al.*, PKC 2012; Bader *et al.*, Eurocrypt 2016) have shown that *any security reduction* for a unique signature scheme or an efficiently re-randomizable signature scheme must loose a factor of q_s, which is the number of signature queries. That is, in all corresponding security reductions, the success probability of reduction is of at most $\frac{1}{q_s}$.

Our Contributions. In contrast to previous impossibility results, in this work, we show that *not all security reductions* for unique signature schemes or efficiently re-randomizable signature schemes must loose a factor of q_s. As a counterexample, we construct a special unique signature scheme with a tight reduction under the Computational Diffie-Hellman (CDH) assumption in the EU-CMA security model. In our security proof with random oracles, we program the security reduction to solve the CDH problem with a loss factor of at most $nq^{\frac{1}{n}}$. Here, n is an integer decided by the scheme designer independent of the security parameter for the scheme construction and q is the number of hash queries to the random oracle. Although our loss factor is associated with q but it can be very small. Taking the example of $n = 25, q = 2^{50}$ and $q_s = 2^{30}$, the loss factor in our security reduction for the given counterexample is of $nq^{\frac{1}{n}} = 100$ only, which is significantly reduced compared to $q_s = 2^{30}$. Hence, we claim that our security reduction is a tight reduction from a non-interactive computational hard problem in the standard EU-CMA security model. We note that our unique signature scheme is the first real[3] unique signature scheme with a tight reduction.

We stress that our counterexample does not indicate any flaw in the given proofs [10, 15, 3]. Instead, we found their proofs fail to capture all security reductions. More precisely, in the traditional security reduction for digital signatures, the reduction uses a forged signature from the adversary that cannot be efficiently computed by the simulator to solve a computational hard problem. We call this reduction *forgery-based reduction*. The tight security reduction in our given example is completely different and we name it as *query-based reduction*, where the reduction uses one of hash queries from the adversary to solve a computational hard problem. We analyze the reason why there is a gap in the proofs of impossibilities when using the query-based reduction in Section 1.4. Notice that the query-based reduction is not a completely new notion because it has been used in enabling provable security for encryption under computational hard problems in the indistinguishability security model, such as the hashed ElGamal encryption scheme [1]. However, this is the first time of applying query-based reduction in proving security for digital signatures.

The given counterexample is of an independent interest in terms of exploring generic approaches for signature scheme constructions with tight reductions. It implies a methodology of constructing a digital signature scheme including unique signatures with a tight reduction in the random oracle model from a digital signature scheme with a loose reduction. Our proposed methodology has to increase the signature length by n times and hence it is somewhat impractical even taking the loss factor into account. However, we stress that our idea is theoretically interesting as it, for the first time, introduces how to obtain a tight proof for digital signatures without the use of a random salt in the signature generation.

Remark 1 *The authors in [10, 15, 3] claimed that the impossibility of tight reductions holds for* any *security reduction. From what they have proved (i.e., given theorems), we found that their "security reduction" refers to the following two types.*

- *Any security reduction without the use of random oracles for a signature scheme.*
- *Any security reduction with random oracles for a hash-and-sign signature scheme [10], such as a full domain hash signature scheme.*

We note that our query-based reduction with random oracles is proposed to prove a unique signature scheme, whose structure is different from the hash-and-sign signature defined in [10]. Our signature scheme can be seen as a hash-and-sign structure repeated for n times. Therefore, our given example is not *included in the above two types of reductions.*

[3] Kakvi and Kiltz in [16] proposed a conceptual level FDH (Full Domain Hash) unique signature scheme with a tight reduction. The proposed RSA-FDH scheme is a unique signature, while the reduction uses a fake but indistinguishable public key in the simulation such that the simulated RSA-FDH signatures are no longer unique. We note that both real signature scheme and simulated signature scheme in our counterexample are unique signatures.

1.1 Overview of Our Counterexample and Security Reduction

In this section, we propose a simplified counterexample with $n = 2$ and show how to program the reduction with a loss factor of at most $2q^{\frac{1}{2}}$ under the CDH assumption in the EU-CMA security model. Here, q is the number of hash queries to the random oracle made by the adversary[4]. The given scheme and reduction can be naturally extended to a full scheme with a loss factor of $nq^{\frac{1}{n}}$, where each signature in the full signature scheme contains $n + 1$ group elements. We admit that the proposed scheme is less efficient compared to other unique signature schemes such as [7] especially with the growth of n. We emphasize that the proposed scheme is merely for demonstrating that there indeed exists a unique signature scheme with a tight reduction.

Our unique signature scheme is constructed over a pairing group $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ with a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where p is the group order of $\mathbb{G}, \mathbb{G}_T$ and g is the generator of $\mathbb{G}$. The simplified scheme is described as follows.

SysGen: The system generation algorithm takes as input a security parameter λ, it chooses a pairing group $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ and a cryptographic hash function $H : \{0, 1\}^* \to \mathbb{G}$ that will be viewed as a random oracle in the security proof. The system parameters are composed of $\mathbb{PG}$ and H.

KeyGen: The key generation algorithm randomly chooses $\alpha \in \mathbb{Z}_p$ and computes $h = g^\alpha$. The algorithm returns a public/secret key pair $(pk, sk) = (h, \alpha)$.

Sign: The signing algorithm takes as input the system parameters, a message $m \in \{0, 1\}^*$ and the secret key sk. It returns the signature $\varSigma_m$ on m as

$$\varSigma_m = (\sigma_1, \sigma_2, \sigma_3) = \Big(H(m)^\alpha,\ H(m||\varSigma_m^1)^\alpha,\ H(m||\varSigma_m^2)^\alpha\Big),$$

where $\varSigma_m^i = (\sigma_1, \sigma_2, \cdots, \sigma_i)$. The final signature $\varSigma_m$ is equivalent to $\varSigma_m^3$.

Verify: The verification algorithm takes as input the system parameters, a signed message $(m, \varSigma_m)$ and the public key pk. It accepts the signature if and only if

$$\begin{aligned} e(\sigma_1, g) &= e(H(m), h), \quad \text{and} \\ e(\sigma_2, g) &= e(H(m||\varSigma_m^1), h), \quad \text{and} \\ e(\sigma_3, g) &= e(H(m||\varSigma_m^2), h). \end{aligned}$$

The structure of our unique signature is similar to the *blockchain* [8, 18]. Each component σ_i in our signature scheme is a *block signature* of a given message m to be signed together with all previous block signatures. The block signature is generated using the BLS signature [7]. In this simplified scheme, each signature $\varSigma_m$ is composed of three block signatures $(\sigma_1, \sigma_2, \sigma_3)$. The full unique signature scheme will be composed

[4] For easy understanding, we assume $q^{\frac{1}{2}}$ and $q^{\frac{1}{n}}$ are both integers for the query number q and the given number n in discussions.

of $n+1$ block signatures. We note that the block signature does not have to use the BLS signature and can be instantiated using a different unique signature from other schemes such as [20][5]. Our proposed signature scheme can also be seen as a variant of the BLS signature scheme towards a tight reduction.

For the security proof of the simplified unique signature scheme, we will program the security reduction in the random oracle model, where H is set as the random oracle. The reduction is an interaction between an adversary who can break the proposed scheme and a simulator who simulates both the real scheme and the random oracle for the adversary. Different from the forgery-based security reduction, the constructed simulator in our reduction will use one of hash queries generated by the adversary instead of a forged signature from the adversary to solve the CDH problem. Therefore, the random oracle plays the most important role in this type of reduction. The reduction is much more complex compared to that for the BLS signature scheme especially in the simulation of the random oracle. Before showing how to program the reduction, we give three preliminaries to help understanding the core of this security reduction.

In this paper, we say an adversary makes a hash query $H(x)$ to the random oracle meaning that the adversary sends x to the random oracle in order to know what $H(x)$ is. Without querying $H(x)$ to the random oracle, $H(x)$ is random and unknown to the adversary, such that the adversary has no advantage in computing $H(x)^\alpha$ even it knows α. In our simplified scheme, due to the "chain" structure, the adversary who does not know the signature Σ_m of a message m must first query $H(m)$, then query $H(m||\Sigma_m^1)$, and finally query $H(m||\Sigma_m^2)$ to the random oracle. This sequence is essential because Σ_m^1 contains the block signature $H(m)^\alpha$ requiring to know $H(m)$ first, and Σ_m^2 contains the block signature $H(m||\Sigma_m^1)^\alpha$ requiring to know $H(m||\Sigma_m^1)$ first. Therefore, each signature generation is required to make three hash queries to the random oracle and these three hash queries must be made sequentially. This is the first preliminary about sequential hash queries on the same message without having its signature.

The adversary could launch hash queries associated with the same message m in the following four cases **before querying its signature or without its signature query**.

- In the first case, the adversary never queries $H(m)$, $H(m||\Sigma_m^1)$, $H(m||\Sigma_m^2)$ to the random oracle.
- In the second case, the adversary queries $H(m)$ only to the random oracle. That is, the adversary never queries $H(m||\Sigma_m^1)$, $H(m||\Sigma_m^2)$ to the random oracle.
- In the third case, the adversary queries $H(m)$ and $H(m||\Sigma_m^1)$ to the random oracle. That is, the adversary never queries $H(m||\Sigma_m^2)$ to the random oracle.
- In the fourth case, the adversary queries all $H(m)$, $H(m||\Sigma_m^1)$, $H(m||\Sigma_m^2)$ to the random oracle. Hash queries associated with the message m^* to be forged by the adversary falls into this case. Otherwise, at least one hash value is random and unknown to the adversary, such that it cannot forge a valid signature of m^* with a non-negligible advantage.

[5] We emphasize that the corresponding security reduction is different when the adopted block signature is changed. In particular, how to simulate a block signature and how to solve a hard problem are dependent on the block signature. However, the reduction method with a tight reduction is the same.

Table 1. Classification of hash queries from the adversary into four cases. Each column refers to the input structure of each query, which will be used in the introduction of the third preliminary.

	Type 0	**Type 1**	**Type 2**
Case 1	–	–	–
Case 2	$H(m)$	–	–
Case 3	$H(m)$	$H(m\|\|\Sigma_m^1)$	–
Case 4	$H(m)$	$H(m\|\|\Sigma_m^1)$	$H(m\|\|\Sigma_m^2)$

These four cases are also summarized in Table 1. We stress that as an example, if the adversary first queries $H(m)$ to the random oracle, then queries the signature Σ_m of m to the simulator, and thereafter queries $H(m||\Sigma_m^1), H(m||\Sigma_m^2)$ to the random oracle for signature verification, we have the message m belongs to the second case not the fourth case. For a message m belonging to the third case, we note that the adversary must compute Σ_m^1 first before the hash query $H(m||\Sigma_m^1)$. For a message m belonging to the fourth case, the adversary must compute Σ_m^1 and Σ_m^2 first before the hash queries $H(m||\Sigma_m^1), H(m||\Sigma_m^2)$. To forge a valid signature, the adversary must compute $\Sigma_{m^*}^1$ and $\Sigma_{m^*}^2$ for hash queries $H(m^*), H(m^*||\Sigma_{m^*}^1), H(m^*||\Sigma_{m^*}^2)$ in order to forge the signature Σ_{m^*} of m^*. Therefore, there must exist one message belonging to the fourth case. Notice that the adversary could make hash queries associated with other messages belonging to the third case or the fourth case. In our reduction, the simulator shall use these special hash queries for reduction and these queries require the adversary to compute block signatures first. When one of block signatures is reducible, the simulator can then use the corresponding block signature to solve a computational hard problem. This is the second preliminary explaining how our query-based reduction works.

In the query-based reduction, we need to identify who (either the adversary or the simulator) **first time** submits a query to the random oracle. We note that the above four cases are only those queries first time generated and submitted by the adversary to the random oracle. Besides these hash queries, the simulator could add some other hash queries to the random oracle. Considering the above example where the adversary first queries $H(m)$ to the random oracle and then queries the signature Σ_m to the simulator. The simulator has to program hash queries $H(m||\Sigma_m^1)$ and $H(m||\Sigma_m^2)$ before the signature simulation on m, where these two queries could be made by the adversary again for the signature verification purpose. These two queries therefore are first time made by the simulator and the simulator should be able to compute Σ_m^1 and Σ_m^2. Notice that the simulator cannot solve the underlying hard problem using one hash query first time submitted to the random oracle by itself. Otherwise, the simulator can solve the hard problem without the adversary. It is therefore important to identify who first time makes a query to the random oracle and we focus on those queries first time generated by the adversary. This is the supplementary of the second preliminary.

The query input to the random oracle can be any arbitrary string adaptively chosen by the adversary. In this work, we assume the concatenation notation "||" will not appear within messages[6], such that it is easy for the simulator to distinguish the input structure of each query denoted by x. Furthermore, the simulator can run the pairing computation to verify the correctness of each block signature. Therefore, for each query x first time generated by the adversary, we define

- x is a query belonging to **Type 0** if x is equal to m without the notation "$||$".
- x is a query belonging to **Type 1** if x is equal to $m||\Sigma_m^1$ with one block signature.
- x is a query belonging to **Type 2** if x is equal to $m||\Sigma_m^2$ with two block signatures.
- x is a query belonging to **Type D** if x is different from the above three types. For example, x is equal to a message concatenated with invalid block signatures or more than two block signatures.

We define three message sets $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2$ during the random oracle simulation as follows to include all messages in those queries generated by the adversary and belonging to **Type 0**, **Type 1** or **Type 2**. We are not interested in queries belonging to **Type D** because they are not used for signature generations. Before the adversary starts making hash queries to the random oracle, these three message sets are initialized with empty sets. These three message sets are managed by the simulator and it adds messages into them when a query from the adversary satisfies the following conditions.

- If x is a query belonging to **Type 0**, the message m in x will be added into the message set $\mathcal{M}_0$. Let $\mathcal{M}_0 = \{m_{0,1}, m_{0,2}, \cdots, m_{0,q_0}\}$ where $|\mathcal{M}_0| = q_0$.
- If x is a query belonging to **Type 1**, the message m in x will be added into the message set $\mathcal{M}_1$. Let $\mathcal{M}_1 = \{m_{1,1}, m_{1,2}, \cdots, m_{1,q_1}\}$ where $|\mathcal{M}_1| = q_1$.
- If x is a query belonging to **Type 2**, the message m in x will be added into the message set $\mathcal{M}_2$. Let $\mathcal{M}_2 = \{m_{2,1}, m_{2,2}, \cdots, m_{2,q_2}\}$ where $|\mathcal{M}_2| = q_2$.

For easy understanding, all messages in the message sets are added sequentially. That is, $m_{i,1}$ is first added to the message set $\mathcal{M}_i$ followed by the message $m_{i,2}$.

Suppose the total number of hash queries made by the adversary is q and the adversary ever queries $H(m^*), H(m^*||\Sigma_{m^*}^1), H(m^*||\Sigma_{m^*}^2)$ for a message m^* in order to forge its signature. According to the first preliminary and Table 1, we have the following important observation

$$\mathcal{M}_2 \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_0, \quad 1 \leq q_2 \leq q_1 \leq q_0 < q.$$

The above relationship is always correct and independent of what messages are selected for hash queries, what messages are selected for signature queries and which message is selected for signature forgery. This is the third preliminary about the relationship of messages in those hash queries generated by the adversary.

[6] This can be done via encoding to make sure m does not contain the concatenation notation. For example, each bit $X \in \{0, 1\}$ of message and block signatures will be encoded into "0X" and the concatenation notation is represented using "11" such that the whole bit string as input to the hash function looks like '$0X_1 0X_2 0X_3 110X_4 0X_5$", where $X_1X_2X_3$ is the message and X_4X_5 is the block signature. We can force the scheme to use this encoding before hashing and signature generation. In the security proof, if a query made by the adversary does not use this encoding, this query is a query with an invalid structure and the simulator will randomly respond on the hash query because it will not be used in signature query or signature forgery.

Now, we are ready to describe how to program the security reduction for the simplified scheme. Suppose there exists an adversary who can break the signature scheme in the EU-CMA security model. The simulator programs the simulation as follows. Given a random instance of the CDH problem (g, g^a, g^b) under a pairing group $\mathbb{PG}$, the simulator sets $\mathbb{PG}$ as the system parameters and $h = g^a$ in the public key simulation, where the secret key $\alpha = a$ is unknown to the simulator. The simulator simulates the random oracle for H instead of giving the hash function to the adversary. During the random oracle simulation, the simulator programs the instance g^b in one of $H(x)$ such that $H(x)^\alpha$ contains[7] the CDH solution g^{ab}. For all other queries, the simulator computes $H(y)$ in the way that $H(y)^\alpha$ is computable by the simulator. The core of oracle simulation is which query should be responded using g^b. We name this query as *target query* and the corresponding message as *target message* denoted by $\hat{m}$. In our simulation, the simulator firstly chooses a random integer $c^* \in \{0, 1\}$ and then it works as follows.

- If $c^* = 0$, it randomly chooses another integer $k^* \in [1, q^{1-\frac{c^*}{2}}]$, where the range is equal to $[1, q]$. In this case, the target query is the one whose message will be the k^*-th message added into the message set $\mathcal{M}_0$, if such a message exists in the message set. Otherwise, aborts. That is, the target query is

  $$m_{c^*,k^*} = \hat{m}.$$

  We have $H(\hat{m})$ contains g^b. When the message $\hat{m} \in \mathcal{M}_0$ also appears in the message set $\mathcal{M}_1$, it means the adversary ever submitted the following query to the random oracle

  $$\hat{m}||\Sigma^1_{\hat{m}},$$

  where the block signature σ_1 in $\Sigma_{\hat{m}}$ is equal to $H(\hat{m})^\alpha$ containing the solution g^{ab} to the hard problem. That is, the simulator embeds g^b in the response to one of **Type 0** queries and will use a **Type 1** query to solve the underlying hard problem.
- If $c^* = 1$, it randomly chooses another integer $k^* \in [1, q^{1-\frac{c^*}{2}}]$ (Note: this range is different from the first case), where the range is equal to $[1, q^{\frac{1}{2}}]$. In this case, the target query is the query whose message will be the k^*-th message added into the message set $\mathcal{M}_1$, if such a message exists in the message set. Otherwise, aborts. That is, the target query is

  $$m_{c^*,k^*}||\Sigma^1_{m_{c^*,k^*}} = \hat{m}||\Sigma^1_{\hat{m}}.$$

  We have $H(\hat{m}||\Sigma^1_{\hat{m}})$ contains g^b. When the message $\hat{m} \in \mathcal{M}_1$ also appears in the message set $\mathcal{M}_2$, it means the adversary ever submitted the following query to the random oracle

  $$\hat{m}||\Sigma^2_{\hat{m}},$$

  where the second block signature σ_2 in $\Sigma^2_{\hat{m}}$ is equal to $H(\hat{m}||\Sigma^1_{\hat{m}})^\alpha$ containing the solution g^{ab} to the hard problem. That is, the simulator embeds g^b in the response

[7] The word "contain" here means that a group element can be extracted from another group element. For example, $H(x) = (g^b)^z$ where $z \in \mathbb{Z}_p^*$ is a random number chosen by the simulator. Then, we have g^{ab} can be extracted from $H(x)^\alpha$ by computing $(H(x)^\alpha)^{\frac{1}{z}}$.

to one of **Type 1** queries and will use a **Type 2** query to solve the underlying hard problem.

We emphasize that the target message $\hat{m}$ does not have to be equal to the message m^* for the signature forgery. This is the main reason why our security reduction has a small loss factor. This completes the core of our simulation.

Now, we analyze the loss factor of our reduction. Let $\Pr[Success]$ be the success probability of reduction when the adversary can successfully forge a valid signature. According to the simulation setting, the reduction is successful when one of hash queries contains the solution to the hard problem. We claim the loss factor is of at most $2q^{\frac{1}{2}}$. A compact probability analysis is given as follows.

First, since $c^* \in \{0, 1\}$ is randomly chosen, we have

$$\begin{aligned}&\Pr[Success]\\ &= \Pr[Success|c^* = 0]\Pr[c^* = 0] + \Pr[Success|c^* = 1]\Pr[c^* = 1]\\ &= \frac{1}{2}\Big(\Pr[Success|c^* = 0] + \Pr[Success|c^* = 1]\Big).\end{aligned}$$

We calculate one of conditional probabilities only, which is decided by q_1 as follows.

- If $q_1 \geq q^{\frac{1}{2}}$, we calculate $\Pr[Success|c^* = 0]$ only where

$$q_0 < q = q^{1-\frac{0}{2}} \quad \text{and} \quad q_1 \geq q^{\frac{1}{2}} = q^{1-\frac{1}{2}}.$$

- Otherwise, $q_1 < q^{\frac{1}{2}}$, we calculate $\Pr[Success|c^* = 1]$ only where

$$q_1 < q^{\frac{1}{2}} = q^{1-\frac{1}{2}} \quad \text{and} \quad q_2 \geq 1 = q^{1-\frac{2}{2}}.$$

One can note that no matter $q_1 \geq q^{\frac{1}{2}}$ or $q_1 < q^{\frac{1}{2}}$, there exists an integer $i^* \in [0, 1]$ (i.e. $i^* \in \{0, 1\}$) such that

$$q_{i^*} < q^{1-\frac{i^*}{2}} \quad \text{and} \quad q_{i^*+1} \geq q^{1-\frac{i^*+1}{2}}.$$

Then, we prove $\Pr[Success|c^* = i^*] \geq 1/q^{\frac{1}{2}}$. The target query is associated with a target message from the message set $\mathcal{M}_{c^*}$, where the target message is m_{c^*,k^*} denoted by $\hat{m}$. When $c^* = i^*$, we have

$$|\mathcal{M}_{c^*}| = q_{c^*} = q_{i^*} < q^{1-\frac{i^*}{2}} \quad \text{and} \quad k^* \in [1, q^{1-\frac{i^*}{2}}],$$

such that each message in $\mathcal{M}_{c^*}$ will be chosen as the target message $\hat{m}$ with probability $1/q^{1-\frac{i^*}{2}}$. Furthermore, we have

$$\mathcal{M}_{c^*+1} \subseteq \mathcal{M}_{c^*} \quad \text{and} \quad |\mathcal{M}_{c^*+1}| = q_{i^*+1} \geq q^{1-\frac{i^*+1}{2}}$$

such that

$$\Pr[Success|c^* = i^*] = \Pr[\hat{m} \in \mathcal{M}_{c^*} \cap \mathcal{M}_{c^*+1}] = \frac{q_{i^*+1}}{q^{1-\frac{i^*}{2}}}.$$

Finally, we have

$$\begin{aligned}\Pr[Success] &= \Pr[Success|c^*=0]\Pr[c^*=0] + \Pr[Success|c^*=1]\Pr[c^*=1]\\ &\geq \Pr[Success|c^*=i^*]\Pr[c^*=i^*]\\ &= \frac{1}{2}\cdot \Pr[\hat{m}\in \mathcal{M}_{c^*}\cap \mathcal{M}_{c^*+1}]\\ &= \frac{1}{2}\cdot \frac{q_{i^*+1}}{q^{1-\frac{i^*}{2}}}\\ &\geq \frac{1}{2}\cdot \frac{q^{1-\frac{i^*+1}{2}}}{q^{1-\frac{i^*}{2}}}\\ &= \frac{1}{2q^{\frac{1}{2}}}.\end{aligned}$$

This completes the high level description of our security reduction for the simplified unique signature scheme where $n=2$.

1.2 Examples

We give four simple examples to verify the success probability. In these examples, suppose $q=9$ and the message chosen by the adversary for signature forgery is $m^*=m_3$. No matter what the adversary has queried to the random oracle, we must have all queried messages satisfy $m_3\in\mathcal{M}_2\subseteq\mathcal{M}_1\subseteq\mathcal{M}_0$. According to our simulation result, we should have that the loss factor is of at most $2\cdot 9^{\frac{1}{2}}=6$. That is, we should have

$$\Pr[Success]\geq \frac{1}{2q^{\frac{1}{2}}}=\frac{1}{6}.$$

Recall that messages in all message sets are in sequence and they are added according to their orders. For instance, the second and the third messages added into the message set $\mathcal{M}_1$ in the Example 2 are m_4 and m_1, respectively. Furthermore,

- A message m is added to the message set $\mathcal{M}_0$ meaning that the adversary generated and submitted the **Type 0** query $x=m$ to the random oracle.
- A message m is added to the message set $\mathcal{M}_1$ meaning that the adversary generated and submitted the **Type 1** query $x=m||\Sigma_m^1$ to the random oracle.
- A message m is added to the message set $\mathcal{M}_2$ meaning that the adversary generated and submitted the **Type 2** query $x=m||\Sigma_m^2$ to the random oracle.

If the message m under **Type 0** and **Type 1** is queried and g^b is embedded in the response to the **Type 0** query m, we can use the **Type 1** query $m||\Sigma_m^1$ to solve the underlying hard problem because Σ_m^1 is equal to $H(m)^\alpha$. Similarly, if the message m under **Type 1** and **Type 2** is queried and g^b is embedded in the response to the **Type 1** query $m||\Sigma_m^1$, we can use the **Type 2** query $m||\Sigma_m^2$ to solve the underlying hard problem because σ_2 in Σ_m^2 is equal to $H(m||\Sigma_m^1)^\alpha$.

According to the setting of (c^*,k^*), the k^*-th message in the message set $\mathcal{M}_{c^*}$ is chosen as the target message $\hat{m}=m_{c^*,k^*}$, where g^b will be embedded in the element

Table 2. The given four examples where $q = 9$.

(a) Example 1

$\mathcal{M}_0 = \{ m_1, m_2, m_3 \}$
$\mathcal{M}_1 = \{ m_1, m_2, m_3 \}$
$\mathcal{M}_2 = \{ m_1, m_2, m_3 \}$

(b) Example 2

$\mathcal{M}_0 = \{ m_1, m_2, m_3, m_4 \}$
$\mathcal{M}_1 = \{ m_2, m_4, m_1, m_3 \}$
$\mathcal{M}_2 = \{ m_3 \}$

(c) Example 3

$\mathcal{M}_0 = \{ m_1, m_2, m_3, m_4, m_5 \}$
$\mathcal{M}_1 = \{ m_3, m_5 \}$
$\mathcal{M}_2 = \{ m_5, m_3 \}$

(d) Example 4

$\mathcal{M}_0 = \{ m_1, m_2, m_3, m_4, m_5, m_6, m_7 \}$
$\mathcal{M}_1 = \{ m_3 \}$
$\mathcal{M}_2 = \{ m_3 \}$

$H(m_{c^*,k^*}||\Sigma^{c^*}_{m_{c^*,k^*}})$. Notice that the integer k^* is randomly chosen from $[1, 9]$ when $c^* = 0$ and randomly chosen from $[1, 3]$ when $c^* = 1$. In the following examples, we only need to consider one case ($c^* = 0$ or $c^* = 1$), whose success probability is at least $1/9^{\frac{1}{2}} = 1/3$.

1. In the first example, the adversary makes the hash queries $H(m), H(m||\Sigma^1_m)$, $H(m||\Sigma^2_m)$ for messages $\{m_1, m_2, m_3\}$. When $c^* = 0$, k^* is randomly chosen from $[1, 9]$. According to the queried messages, we have the reduction is successful if $c^* = 0$ and $k^* \in \{1, 2, 3\}$, because the message m_{c^*,k^*} under **Type 0** and **Type 1** will be both queried. Since k^* is randomly chosen from $[1, 9]$, we therefore have
$$\Pr[Success] \geq \Pr[k^* \in \{1, 2, 3\}] \cdot \Pr[c^* = 0] = \frac{3}{9} \cdot \frac{1}{2} = \frac{1}{6}.$$
2. In the second example, the adversary makes the hash query $H(m)$ for messages $\{m_1, m_2, m_3, m_4\}$, the hash query $H(m||\Sigma^1_m)$ for messages $\{m_2, m_4, m_1, m_3\}$ and the hash query $H(m||\Sigma^2_m)$ for the message m_3 only. When $c^* = 0$, k^* is randomly chosen from $[1, 9]$. According to the queried messages, we have the reduction is successful if $c^* = 0$ and $k^* \in \{1, 2, 3, 4\}$, because the message m_{c^*,k^*} under **Type 0** and **Type 1** will be both queried. Since k^* is randomly chosen from $[1, 9]$, we therefore have
$$\Pr[Success] \geq \Pr[k^* \in \{1, 2, 3, 4\}] \cdot \Pr[c^* = 0] = \frac{4}{9} \cdot \frac{1}{2} \geq \frac{1}{6}.$$
3. In the third example, the adversary makes the hash query $H(m)$ for messages $\{m_1, m_2, m_3, m_4, m_5\}$, the hash queries $H(m||\Sigma^1_m), H(m||\Sigma^2_m)$ for messages $\{m_3, m_5\}$ only. When $c^* = 1$, k^* is randomly chosen from $[1, 3]$. According to the queried messages, we have the reduction is successful if $c^* = 1$ and $k^* \in \{1, 2\}$, because the message m_{c^*,k^*} under **Type 1** and **Type 2** will be both queried. Since k^* is randomly chosen from $[1, 3]$, we therefore have
$$\Pr[Success] \geq \Pr[k^* \in \{1, 2\}] \cdot \Pr[c^* = 1] = \frac{2}{3} \cdot \frac{1}{2} \geq \frac{1}{6}.$$

4. In the fourth example, the adversary makes the hash query $H(m)$ for messages $\{m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$, the hash queries $H(m||\Sigma_m^1), H(m||\Sigma_m^2)$ for the message m_3 only. When $c^* = 1$, k^* is randomly chosen from $[1, 3]$. According to the queried messages, we have the reduction is successful if $c^* = 1$ and $k^* = 1$, because the message m_{c^*,k^*} under **Type 1** and **Type 2** will be both queried. Since k^* is randomly chosen from $[1, 3]$, we therefore have
$$\Pr[Success] \geq \Pr[k^* = 1] \cdot \Pr[c^* = 1] = \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{6}.$$

1.3 Security Reduction for the Full Scheme and Other Discussions

The above reduction is programmed for the simplified unique signature scheme, where each signature is composed of three block signatures only. The reduction for the full unique signature scheme with $n+1$ block signatures is more complex but similar, which is given in Section 5. In the corresponding reduction, $n+1$ message sets $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_n$ will be defined which also satisfy
$$\mathcal{M}_n \subseteq \mathcal{M}_{n-1} \subseteq \cdots \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_0.$$
Similarly, we prove there exists an integer $i^* \in [0, n-1]$ such that
$$|\mathcal{M}_{i^*}| < q^{1-\frac{i^*}{n}} \text{ and } |\mathcal{M}_{i^*+1}| \geq q^{1-\frac{i^*+1}{n}}.$$
In the corresponding simulation, the random values (c^*, k^*) will be chosen from the following ranges
$$c^* \in [0, n-1], \quad k^* \in [1, q^{1-\frac{c^*}{n}}].$$
When g^b is programmed in the response to the target query $\hat{m}||\Sigma_{\hat{m}}^{i^*}$ for the message $\hat{m} \in \mathcal{M}_{i^*}$ and the target message $\hat{m}$ also appears in the message set $\mathcal{M}_{i^*+1}$ (i.e. the adversary also submits the hash query $\hat{m}||\Sigma_{\hat{m}}^{i^*+1}$ to the random oracle), we have the hash query $\hat{m}||\Sigma_{\hat{m}}^{i^*+1}$ contains the solution to the CDH problem. The success reduction requires that c^* happens to be equal to i^* mentioned above. Since $|\mathcal{M}_{i^*}| < q^{1-\frac{i^*}{n}}$ and $k^* \in [1, q^{1-\frac{i^*}{n}}]$, we have $H(m||\Sigma_m^{i^*})$ for any message m in $\mathcal{M}_{i^*}$ will be responded using g^b with probability $1/q^{1-\frac{i^*}{n}}$. Furthermore, since $\mathcal{M}_{i^*+1} \subseteq \mathcal{M}_{i^*}$ and $|\mathcal{M}_{i^*+1}| = q_{i^*+1} \geq q^{1-\frac{i^*+1}{n}}$, $\hat{m}$ can be any one of q_{i^*+1} messages within $\mathcal{M}_{i^*+1}$. Therefore, we have the success probability
$$\begin{aligned}\Pr[Success] &= \sum_{i=0}^{n-1} \Pr[Success|c^* = i] \Pr[c^* = i] \\ &\geq \Pr[Success|c^* = i^*] \Pr[c^* = i^*] \\ &= \frac{1}{n} \cdot \frac{q_{i^*+1}}{q^{1-\frac{i^*}{n}}} \\ &\geq \frac{1}{n} \cdot \frac{q^{1-\frac{i^*+1}{n}}}{q^{1-\frac{i^*}{n}}} \\ &= \frac{1}{nq^{\frac{1}{n}}}.\end{aligned}$$

This completes the overview of our counterexample with a tight reduction.

Our proposed scheme can select other signature schemes to generate block signatures. The security reduction is still tight as long as only the block signature of the message $m_{c^*,k^*} \in \mathcal{M}_{c^*}$ is reducible and the block signatures of other messages are simulatable. The proposed signature structure and the reduction method towards a tight reduction are independent of the block signatures and hence are universal.

Comparison of reductions. For easy understanding of the query-based reduction, we compare it with the traditional forgery-based reduction according to the above reduction as follows.

- The forgery-based reduction utilizes a forged signature from the adversary to solve an underlying hard problem, while the query-based reduction utilizes one of hash queries from the adversary to solve an underlying hard problem.
- In the forgery-based reduction, the solution must be associated with the message of forged signature. While in the query-based reduction, the solution does not have to be related to the message for signature forgery.
- In the forgery-based reduction, the simulator cannot stop the simulation before it receives a forged signature from the adversary. While in the query-based reduction, the simulator can stop the simulation immediately after receiving a query containing the solution to the underlying hard problem.
- In the forgery-based reduction, the simulator cannot compute the forged signature from the adversary. While in the query-based reduction, the simulator can compute signatures on all messages as long as the adversary's hash queries have already contained the solution to the underlying hard problem.
- The forgery-based reduction can be with or without random oracles, while the query-based reduction must be with random oracles.

Remark 2 *In our given signature scheme, each signature comprises of three block signatures in the simplified scheme and $n+1$ block signatures in the full signature scheme. The last block signature is not used in our security reduction, except forcing the adversary to make at least a* **Type 2** *query or a* **Type n** *query (full scheme). We note that the number of block signatures can be reduced by one if we partially adopt the forgery-based reduction. Taking the simplified scheme as an example. It will be composed of two block signatures instead of three, such that there is no* **Type 2** *query. When $c^* = 1$, g^b is programmed in the response to a* **Type 1** *query. That is, the target query is*

$$m_{c^*,k^*}||\Sigma^1_{m_{c^*,k^*}} = \hat{m}||\Sigma^1_{\hat{m}}.$$

We have that $H(\hat{m}||\Sigma^1_{\hat{m}})$ contains g^b. When the message $\hat{m} \in \mathcal{M}_1$ is chosen as the message m^* to be forged, we have the second block signature in the forged signature is $H(\hat{m}||\Sigma^1_{\hat{m}})^\alpha$ which contains the solution to the CDH problem. That is, we use the forged signature to solve the underlying hard problem. We stress that this reduction is also correct. However, we do not adopt this method because it is relatively simpler to use only query-based reduction in the security proof.

1.4 The Gap in The Proofs of Impossibilities

We explain why our query-based reduction for the counterexample cannot be captured by the meta-reduction proofs [10, 15, 3]. Specifically, we would like to clarify the gap in the existing proofs for the impossibilities of tight reductions. We start with briefly recalling the notion of *meta-reduction*, which is the main technique adopted to derive their impossibility results. It is worth mentioning that Bader *et al.* [3] proposed a different meta-reduction to derive more generalized impossibility results. However, we note that they have the same principle. We provide more details as follows.

The meta-reduction proof for impossibilities. Roughly speaking, a meta-reduction is associated with the following three entities.

- The reducer $\mathcal{R}$ who can reduce a forged signature to solving an underlying hard problem. Let $\epsilon_{\mathcal{R}}$ be the success probability of solving the underlying hard problem.
- The real forger (adversary) $\mathcal{F}$ who is able to forge a valid signature. Let $\epsilon_{\mathcal{F}}$ be the advantage (probability) of forging a valid signature.
- The simulated forger $\overline{\mathcal{F}}$ who cannot forge a valid signature and aims to solve an underlying hard problem with the help of $\mathcal{R}$ but without the help of $\mathcal{F}$.

In the traditional security reduction, the interaction is between $\mathcal{R}$ and $\mathcal{F}$, while in the meta-reduction, the interaction is between $\overline{\mathcal{F}}$ and $\mathcal{R}$ only without the real forger. Specifically, in the meta-reduction, a simulated forger $\overline{\mathcal{F}}$ is constructed to solve a hard problem with the help of the reducer $\mathcal{R}$. Precisely, $\overline{\mathcal{F}}$ firstly asks $\mathcal{R}$ to sign a number of messages and then rewinds $\mathcal{R}$ to the state before signature queries and re-starts the security game with $\mathcal{R}$. Finally, it forges a valid signature using one of the signed message/signature pairs (say (m^*, Σ_{m^*})) obtained before the rewinding. When the simulated forger $\overline{\mathcal{F}}$ is indistinguishable from the real forger $\mathcal{F}$, the reducer $\mathcal{R}$ should be able to output a solution to the underlying hard problem with a certain probability. It has been proved in [10] that through the above meta-reduction, $\overline{\mathcal{F}}$ can obtain a solution to the underlying hard problem from the reducer $\mathcal{R}$ with a success probability at least

$$\epsilon_{\overline{\mathcal{F}}} = \epsilon_{\mathcal{R}} - \frac{\epsilon_{\mathcal{F}}}{\Omega(q_s)},$$

for q_s number of signature queries to the reducer $\mathcal{R}$.

If $\epsilon_{\overline{\mathcal{F}}}$ is positive and non-negligible, it means we can efficiently solve a hard problem using the simulated forger $\overline{\mathcal{F}}$ without the help of the real forger. This would contradict with the underlying hardness assumption. It therefore implies that $\epsilon_{\mathcal{R}}$ must be no more than $\epsilon_{\mathcal{F}}/\Omega(q_s)$ for all security reductions. Hence, the loss factor in any security reduction is of at least q_s.

The central argument in the meta-reduction requires that the simulated forger $\overline{\mathcal{F}}$ is indistinguishable from the real forger $\mathcal{F}$. To achieve this, the forged signature generated by the simulated forger must be indistinguishable from the forged signature generated by the real forger, such that $\mathcal{R}$ is convinced that it is interacting with a real forger and thus outputs a solution to the hard problem. In the literature, unique signatures [10] and efficiently re-randomizable signatures [15, 3] are shown to be able to capture this indistinguishability.

The gap in the proofs. One could note that the existing work [10, 15, 3] implicitly claimed that the aforementioned meta-reduction works as long as the simulated forger $\overline{\mathcal{F}}$ outputs a *correctly distributed signature* (and hence indistinguishable from a real forged signature) to convince $\mathcal{R}$ that it is a real forger. In fact, this is the central argument for the existing meta-reductions in proving the impossibility results for unique signatures and their generalization. Let $\Sigma(pk, m)$ be the set of signatures Σ_m with regards to the message m and the public key pk under the system parameters. During the proofs [10, 15, 3], they are silently assumed that this signature set $\Sigma(pk, m)$ in the simulation is determined by pk, m and possibly by additional random oracle queries that are assumed to be efficiently computable. However, this assumption does not necessarily hold in all security reductions. In our example, each useful hash query except **Type 0** requires to compute a block signature that is not efficiently computable without knowing the signing key. Any successful (real) forger $\mathcal{F}$ must make some non-efficiently computable hash queries to the random oracle. This is the reason why the meta-reduction requiring that all hash queries are efficiently computable will fail to simulate the forger.

More precisely, as shown in our query-based reduction for the simplified scheme, a successful reduction essentially relies on the fact that the real forger must at least generate hash queries to random oracle satisfying $q_2 \geq 1$ (or $q_n \geq 1$ in the full scheme). This is guaranteed by the assumption that the real forger is able to forge a valid signature. Suppose the aforementioned meta-reduction utilizes our constructed reducer $\mathcal{R}'$. We stress that the simulated forger $\overline{\mathcal{F}}$ is not able to generate hash queries satisfying $q_2 \geq 1$. Otherwise, it will be the real forger. During the first round before rewinding, the simulated forger $\overline{\mathcal{F}}$ should make some hash queries and signature queries. When the simulated forger $\overline{\mathcal{F}}$ rewinds the interaction, it is worth noting that $\overline{\mathcal{F}}$ must rewind $\mathcal{R}'$ to the state after the hash queries. This is because if it rewinds $\mathcal{R}'$ to the state before the hash queries, all hash values before and after the rewinding will be completely different in the random oracle model and hence the simulated forger $\overline{\mathcal{F}}$ cannot use one of queried signatures to derive the forged signature. That is, all previous hash queries are not changed and the simulated forger $\overline{\mathcal{F}}$ could query more. Finally, even the simulated forger $\overline{\mathcal{F}}$ returns a valid signature indistinguishable from the one generated by the real forger, we still have that the hash queries do not satisfy the requirement of $q_2 \geq 1$. In this case, the reducer $\mathcal{R}'$ fails and the meta-reduction breaks down, because the simulated forger $\overline{\mathcal{F}}$ cannot solve an underlying hard problem.

2 Preliminaries

2.1 Definition of Digital Signatures

A digital signature scheme $\mathsf{Sig} = (\mathsf{SysGen}, \mathsf{KeyGen}, \mathsf{Sig}, \mathsf{Verify})$ consists of the following four algorithms:

SysGen(1^λ). Taking as input a security parameter 1^λ, this system generation algorithm returns the system parameters $params$.

KeyGen$(params)$. Taking as input the system parameters $params$, this key generation algorithm returns a public/secret key pair (pk, sk).

Sign$(params, sk, m)$. Taking as input the system parameters $params$, a secret key sk and a message m, this signing algorithm returns a signature of m denoted by Σ_m.

Verify$(params, pk, \Sigma_m, m)$. Taking as input the system parameters $params$, a public key pk, a signature Σ_m and a message m, this verification algorithm outputs 0 (reject) or 1 (accept).

We say that a signature is *valid* if $\textbf{Verify}(params, pk, \Sigma_m, m) = 1$ for all generated system parameters $params$, public key pk and any message m. In the rest of paper, we omit the input of the system parameters $params$ in the signing algorithm and the verification algorithm unless otherwise stated explicitly.

2.2 Definition of Unique Signatures and Efficiently Re-Randomizable Signatures

Given a public key pk and a message m, we denote

$$\Sigma(pk, m) = \Big\{\Sigma_m : \textbf{Verify}(pk, \Sigma_m, m) = 1\Big\}$$

as the set of signatures Σ_m with regards to the message m and the public key pk under the system parameters $params$. In particular, this signature set could be associated with specific instantiations of hash functions adopted as part of system parameters. This notion was introduced in [3] to define unique signatures and re-randomizable signatures.

Definition 1 (Unique Signatures). *A signature scheme is unique if*

$$\Big|\Sigma(pk, m)\Big| = 1$$

holds for all public keys under the system parameters $params$ and all messages.

Definition 2 (Efficiently Re-Randomizable Signatures). *A signature scheme is efficiently re-randomizable with* t-re-randomizable*, if there exists an efficient algorithm* ReRand *running in polynomial time at most t, such that for all (pk, Σ_m, m) under the system parameters $params$ with $\textbf{Verify}(pk, \Sigma_m, m) = 1$ holds that the output distribution of*

$$\mathsf{ReRand}(pk, \Sigma_m, m)$$

is identical to the uniform distribution over $\Sigma(pk, m)$.

Unique signatures are a particular case of efficiently re-randomizable signatures where

$$\mathsf{ReRand}(pk, \Sigma_m, m) = \Sigma_m.$$

Therefore, the efficiently re-randomizable signatures can be seen as the generalization of unique signatures. An important requirement of efficiently re-randomizable signatures is the uniform distribution over $\Sigma(pk, m)$. Generally speaking, efficiently re-randomizable signatures are signatures whose random salts used in signature generations can be changed uniformly without the knowledge of the secret key, such that we cannot use a random salt to switch signatures between simulatable and reducible. That is, all signatures on the same message must be either simulatable or reducible. Therefore, all efficiently re-randomizable signatures are believed to have optimal security reductions with a loss factor of at least q_s, same as unique signatures.

2.3 Security Model

We recall the notion of existential unforgeability against chosen message attacks (EU-CMA) played between a challenger and an adversary $\mathcal{A}$.

Setup: The challenger runs the SysGen algorithm to generate the system parameters $params$ and runs the KeyGen algorithm to generate a key pair (pk, sk). The system parameters and the public key pk are sent to the adversary.

Signature-Query: The adversary sequentially asks the signature of any message that is adaptively chosen by the adversary. To query the signature of m, $\mathcal{A}$ submits the message m to the challenger. The challenger runs the Sign algorithm and returns a signature Σ_m of m to the adversary.

Forgery: The adversary outputs a message m^* and the corresponding signature Σ_{m^*}. It wins the game if Σ_{m^*} is a valid signature of m^* and m^* has never been queried for its signature.

We refer to such adversary as an EU-CMA adversary and define the advantage of $\mathcal{A}$ in winning the above game as

$$\Pr\left[\mathsf{EU\text{-}CMA}^{\mathcal{F}}_{\mathsf{Sig}}\left(1^\lambda\right) \to 1\right] = \epsilon.$$

Definition 3. *We say that a signature scheme* Sig *is* (t, q_s, ϵ)*-secure in the EU-CMA security model if any PPT adversary* $\mathcal{A}$ *who runs in* t *polynomial time and makes at most* q_s *signature queries has advantage at most* ϵ *in winning the game, where* ϵ *is negligible function of the input security parameter.*

2.4 Complexity Assumption

Let $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ be a pairing group where p is the order of the groups $\mathbb{G}, \mathbb{G}_T$, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is the bilinear map and g is a generator of $\mathbb{G}$. Our proposed unique signature scheme is based on the BLS signature scheme in the symmetric pairing stated in $\mathbb{PG}$. The underlying hard assumption for our scheme is the Computational Diffie-Hellman (CDH) assumption, which says it is hard to compute g^{ab} from a given instance (g, g^a, g^b) in the pairing group $\mathbb{PG}$, where a, b are unknown and uniformly chosen from the space $\mathbb{Z}_p$. The formal definition of the CDH assumption is omitted here.

3 The Full Counterexample of Unique Signature Scheme

Our full unique signature scheme is described as follows.

SysGen: The system generation algorithm takes as input a security parameter 1^λ and a small integer n. It chooses a pairing group $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, p, e, g)$ and a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}$ that will be viewed as a random oracle in the security proof. The system parameters $params$ are composed of ($\mathbb{PG}$, H, n). A signature will be composed of $n + 1$ block signatures.

KeyGen: The key generation algorithm randomly chooses $\alpha \in \mathbb{Z}_p$ and computes $h = g^\alpha$. The algorithm returns a public/secret key pair $(pk, sk) = (h, \alpha)$.

Sign: The signing algorithm takes as input the system parameters $params$, a message $m \in \{0,1\}^*$ and the secret key sk. It returns the signature $\varSigma_m$ on m as

$$\begin{aligned}\varSigma_m &= \Big(\sigma_1, \sigma_2, \sigma_3, \cdots, \sigma_n, \sigma_{n+1}\Big)\\ &= \Big(H(m)^\alpha,\ H(m||\varSigma_m^1)^\alpha,\ H(m||\varSigma_m^2)^\alpha, \cdots, H(m||\varSigma_m^{n-1})^\alpha, H(m||\varSigma_m^n)^\alpha\Big),\end{aligned}$$

where $\varSigma_m^i = (\sigma_1, \sigma_1, \cdots, \sigma_i)$ and the final signature $\varSigma_m$ is equivalent to $\varSigma_m^{n+1}$.

Verify: The verification algorithm takes as input the system parameters $params$, a signed message $(m, \varSigma_m)$ and the public key pk. It accepts the signature if and only if the $n+1$ pairing computations are correct. That is, $e(\sigma_1, g) = e(H(m), h)$ and $e(\sigma_{i+1}, g) = e(H(m||\varSigma_m^i), h)$ holds for all $i = 1, 2, \cdots, n$.

4 Two Essential Lemmas for Security Reduction

Let $m \in \{0,1\}^*$ be a message and $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_n$ be $n+1$ message sets defined as follows.

- The message set $\mathcal{M}_{i+1}$ is a subset of $\mathcal{M}_i$ for all $i = 0, 1, 2, \cdots, n-1$.

$$\mathcal{M}_n \subseteq \mathcal{M}_{n-1} \subseteq \mathcal{M}_{n-2} \subseteq \cdots \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_0.$$

- The set $\mathcal{M}_i$ has q_i distinct messages and is composed of $\{m_{i,1}, m_{i,2}, \cdots, m_{i,q_i}\}$.

$$\begin{array}{l}\hline \mathcal{M}_0 = \{\ m_{0,1}\ ,\ m_{0,2}\ ,\ m_{0,3}\ ,\ \cdots\ ,\ \cdots\ ,\ \cdots\ ,\ \cdots\ ,\ m_{0,q_0}\ \}\\ \mathcal{M}_1 = \{\ m_{1,1}\ ,\ m_{1,2}\ ,\ m_{1,3}\ ,\ \cdots\ ,\ \cdots\ ,\ \cdots\ ,\ m_{1,q_1}\ \}\\ \mathcal{M}_2 = \{\ m_{2,1}\ ,\ m_{2,2}\ ,\ m_{2,3}\ ,\ \cdots\ ,\ \cdots\ ,\ m_{2,q_2}\ \}\\ \qquad\quad \cdots\\ \mathcal{M}_n = \{\ m_{n,1},\ m_{n,2}\ ,\ m_{n,3}\ ,\ \cdots\ ,\ m_{n,q_n}\ \}\\ \hline\end{array}$$

We have the following two essential lemmas based on the above message set definitions for our security reduction. The first lemma is proposed for proving the second lemma.

Lemma 1 (Range Lemma). *Suppose* $q_0 < q$ *and* $q_n \geq 1$. *There exists an integer* $i^* \in [0, n-1]$ *satisfying*

$$q_{i^*} < q^{1-\frac{i^*}{n}} \quad \textit{and} \quad q_{i^*+1} \geq q^{1-\frac{i^*+1}{n}}.$$

Proof of Lemma 1. We use a proof via contradiction to prove the existence of i^*. If the integer i^* does not exist, it means

$$q_i \geq q^{1-\frac{i}{n}} \quad \text{or} \quad q_{i+1} < q^{1-\frac{i+1}{n}}$$

holds for all $i = 0, 1, 2, 3, \cdots, n-1$.

For any integer $j \in [1, n-1]$, we have

- When $i = j - 1$, it means

$$q_{j-1} \geq q^{1-\frac{j-1}{n}} \quad \text{or} \quad q_j < q^{1-\frac{j}{n}}.$$

- When $i = j$, it means

$$q_j \geq q^{1-\frac{j}{n}} \quad \text{or} \quad q_{j+1} < q^{1-\frac{j+1}{n}}.$$

That is, we have

$$\Big(q_{j-1} \geq q^{1-\frac{j-1}{n}} \bigvee q_j < q^{1-\frac{j}{n}}\Big) \bigwedge \Big(q_j \geq q^{1-\frac{j}{n}} \bigvee q_{j+1} < q^{1-\frac{j+1}{n}}\Big).$$

It indicates that

$$\text{If } q_j \geq q^{1-\frac{j}{n}} \quad \text{then} \quad q_{j-1} \geq q^{1-\frac{j-1}{n}} \ \text{ for } j = n-1, n-2, n-3, \cdots, 2, 1.$$

When $i = n - 1$, we also have

$$q_{n-1} \geq q^{1-\frac{n-1}{n}} \quad \text{or} \quad q_n < q^{1-\frac{n}{n}}.$$

Since $q_n < q^{1-\frac{n}{n}} = 1$ contradicts with the assumption $q_n \geq 1$, we therefore have $q_{n-1} \geq q^{1-\frac{n-1}{n}}$ and deduct

$$q_{n-2} \geq q^{1-\frac{n-2}{n}}, \ q_{n-3} \geq q^{1-\frac{n-3}{n}}, \ \cdots \ q_2 \geq q^{1-\frac{2}{n}} \ q_1 \geq q^{1-\frac{1}{n}} \ q_0 \geq q^{1-\frac{0}{n}}.$$

The deduction $q_0 \geq q^{1-\frac{0}{n}} = q$ contradicts with the assumption $q_0 < q$. Therefore, the assumption at the beginning of the proof is wrong and hence the integer $i^* \in [0, n-1]$ exists. This completes the proof. □

Lemma 2 (Probability Lemma). *Suppose $q_0 < q$ and $q_n \geq 1$. If we randomly choose two integers (c^*, k^*) satisfying $c^* \in [0, n-1]$ and $k^* \in [1, q^{1-\frac{c^*}{n}}]$, then the target message $m_{c^*,k^*} \in \mathcal{M}_{c^*}$ will appear in the message set $\mathcal{M}_{c^*+1}$ with probability at least*

$$\frac{1}{n} \cdot \frac{1}{q^{\frac{1}{n}}}$$

for any $n+1$ defined message sets $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_n$. Here, if the chosen message $m_{c^*,k^*} \notin \mathcal{M}_{c^*}$ (the size of message set is less than k^*), we define $m_{c^*,k^*} \notin \mathcal{M}_{c^*+1}$.

Proof of Lemma 2. Let $\Pr[Success]$ be the success probability that the chosen message $m_{c^*,k^*} \in \mathcal{M}_{c^*}$ also appears in the message set $\mathcal{M}_{i^*+1}$. We are going to prove

$$\Pr[Success] \geq \frac{1}{n} \cdot \frac{1}{q^{\frac{1}{n}}}.$$

The success probability is associated with the choice of (c^*, k^*), where c^* is randomly chosen from $[0, n-1]$. Hence, we have $\Pr[c^* = i] = \frac{1}{n}$ for any $i \in [0, n-1]$. According to the setting, we have

$$\Pr[Success] = \sum_{i=0}^{n-1} \Pr[Success|c^* = i] \Pr[c^* = i] \geq \frac{1}{n} \cdot \Pr[Success|c^* = i^*],$$

where i^* is the integer defined in Lemma 1.

According to Lemma 1, we have

$$|\mathcal{M}_{i^*}| = q_{i^*} < q^{1-\frac{i^*}{n}} \text{ and } |\mathcal{M}_{i^*+1}| = q_{i^*+1} \geq q^{1-\frac{i^*+1}{n}}.$$

When $c^* = i^*$, the target message m_{c^*,k^*} is chosen from the k^*-th message of the message set $\mathcal{M}_{i^*}$. Since $|\mathcal{M}_{i^*}| < q^{1-\frac{i^*}{n}}$ and k^* is randomly chosen from $[1, q^{1-\frac{i^*}{n}}]$, any message in $\mathcal{M}_{i^*}$ will be selected as the target message with probability $1/q^{1-\frac{i^*}{n}}$. Furthermore, since $\mathcal{M}_{i^*+1} \subseteq \mathcal{M}_{i^*}$, the success event will happen when m_{c^*,k^*} also appears in $\mathcal{M}_{i^*+1}$. There are q_{i^*+1} messages in $\mathcal{M}_{i^*+1}$ and hence we have the success probability

$$\begin{aligned}
\Pr[Success] &= \sum_{i=0}^{n-1} \Pr[Success|c^* = i] \Pr[c^* = i] \\
&\geq \frac{1}{n} \cdot \Pr[Success|c^* = i^*] \\
&= \frac{1}{n} \cdot \Pr[m_{c^*,k^*} \in \mathcal{M}_{i^*} \cap \mathcal{M}_{i^*+1}] \\
&= \frac{1}{n} \cdot \frac{q_{i^*+1}}{q^{1-\frac{i^*}{n}}} \\
&\geq \frac{1}{n} \cdot \frac{q^{1-\frac{i^*+1}{n}}}{q^{1-\frac{i^*}{n}}} \\
&= \frac{1}{nq^{\frac{1}{n}}}.
\end{aligned}$$

This completes the proof. □

5 Security Proof With A Tight Reduction

In this section, we prove the proposed full unique signature scheme is secure under the CDH assumption in the EU-CMA security model with a tight reduction, where H is set as a random oracle in the security proof.

Theorem 1. *Let H be a random oracle and q be the number of queries to the random oracle. If the proposed full signature scheme can be broken with (t, q_s, ϵ) in the EU-CMA model, the CDH problem is solvable with*

$$\left(t + O(q_s n), \frac{\epsilon}{nq^{1/n}}\right).$$

Proof. Suppose there exists an adversary $\mathcal{A}$ who can break the signature scheme in the EU-CMA model. We construct a simulator $\mathcal{B}$ that uses one of hash queries generated by the adversary to solve the CDH problem. Given as input the instance (g, g^a, g^b) in the pairing group $\mathbb{PG}$, the simulator aims to compute g^{ab}. $\mathcal{B}$ interacts with the adversary as follows.

Setup: The system parameters are the pairing group $\mathbb{PG}$ and the integer n, where H is set as a random oracle controlled by the simulator. For the key pair, the simulator sets $\alpha = a$ and hence the public key $h = g^{\alpha} = g^{a}$ is available from the instance. The system parameters and the public key are sent to the adversary.

Hash-Query: We state how to simulate the random oracle here. The adversary can access the random oracle any time, such as after the signature query. We note that the random oracle simulation is the core of our proof for having a tight reduction.

Before simulating the random oracle, the simulator firstly chooses a random integer $c^* \in [0, n-1]$ and then chooses another random value k^* from the range

$$\left[1, q^{1-\frac{c^*}{n}}\right].$$

In particular, we have the range is $[1, q]$ when $c^* = 0$ and the range is $[1, q^{\frac{1}{n}}]$ when $c^* = n-1$. The size of range for k^* is dependent on the integer c^*.

Let $\mathcal{L}$ be the list which records all queries and responses. Each query and its response are stored in a tuple. For each tuple, the format is

$$(x,\ I_x,\ T_x, O_x,\ U_x,\ z_x),$$

which is explained as follows.

x refers to the query input.
I_x refers to the identity either the adversary $\mathcal{A}$ or the simulator $\mathcal{B}$.
T_x refers to the type of the hash query.
O_x refers to the order index of the query within the same type.
U_x refers to the response to x, i.e., $U_x = H(x)$.
z_x refers to the secret for computing U_x.

For a query on x, if there already has a tuple $(x,\ I_x,\ T_x, O_x,\ U_x,\ z_x)$ in the list, the simulator responds with U_x. Otherwise, the simulator responds on this new query as follows.

Response of object I_x. The object I_x is to identify who first time generates and submits x to the random oracle. For easy understanding, we comprehend the query in the way that both the adversary and the simulator can query to the random oracle, although the random oracle is controlled by the simulator. If a query on x is first time generated and submitted by the adversary, we say this query is made by the adversary and set $I_x = \mathcal{A}$. Otherwise, we set $I_x = \mathcal{B}$.

Taking a new message m as the example. Suppose the adversary firstly queries $H(m)$, $H(m||\Sigma_m^1)$, $H(m||\Sigma_m^2)$ to the random oracle and then queries the signature of m to the simulator. Notice that the signature generation on the message m requires to know all the following values

$$H(m),\ H(m||\Sigma_m^1),\ H(m||\Sigma_m^2),\ H(m||\Sigma_m^3),\ \cdots,\ H(m||\Sigma_m^n).$$

The hash list does not record how to respond on hash queries $H(m||\Sigma_m^i)$ for all $i = 3, 4, \cdots, n$. Therefore, the simulator must add all these hash queries to the random oracle first before generating its signature. Notice that the hash queries $H(m||\Sigma_m^3), \cdots,$ $H(m||\Sigma_m^n)$ could be made by the adversary again for signature verification, but they are first time generated and made by the simulator. Hence, we define

- For any $x \in \{m,\ m||\Sigma_m^1,\ m||\Sigma_m^2\}$, the corresponding I_x for x is $I_x = \mathcal{A}$.
- For any $x \in \{m||\Sigma_m^3, \cdots, m||\Sigma_m^n\}$, the corresponding I_x for x is $I_x = \mathcal{B}$.

Response of object T_x. We assume "||" is a concatenation notation that will never appear within messages after encoding (See the footnote in the introduction section to know how to achieve it.). The simulator can also run the verification algorithm to verify whether each block signature is correct or not. Therefore, it is easy to distinguish the input structure of all hash queries. We define $n + 2$ types of hash queries to the random oracle.

Type i. $x = m||\Sigma_m^i$. Here, Σ_m^i denotes the first i block signatures of m and i refers to any integer $i \in [0, n]$. We assume $m||\Sigma_m^0 = m$ without "||" in x for easy analysis.
Type D. x is a query different from the previous $n+1$ types. For example, $x = m||R_m$ but $R_m \neq \Sigma_m^i$ for any $i \in [1, n]$, or $x = m||\Sigma_m^{i'}$ for any $i' \geq n + 1$.

The object T_x is set as follows.

- If $I_x = \mathcal{B}$, then $T_x = \perp$.
- Otherwise, suppose $I_x = \mathcal{A}$. Then, the simulator can run the verification algorithm to know which type x belongs to and set

$$T_x = \begin{cases} i & \text{if } x \text{ belongs to } \textbf{Type i} \text{ for any } i \in [0, n] \\ \perp & \text{otherwise } x \text{ belongs to } \textbf{Type D} \end{cases}.$$

We emphasize that T_x and O_x are used to mark "valid" queries generated by the adversary only, which will be used in signature generation. We define **Type D** is to match the truth that the adversary can generate any arbitrary string as a query to the random oracle. Notice that the last type of queries will never be used in signature generation or signature forgery.

Response of object O_x. The object O_x is set as follows.

- If $T_x = \perp$, then $O_x = \perp$.
- Otherwise, suppose $T_x = c$. Then, $O_x = k$ if x is the k-th new query added into the list $\mathcal{L}$ in those queries where $T_x = c$.

To calculate the integer k for the new query x, the simulator must count how many queries have been added in $\mathcal{L}$, where only those queries with the same T_x will be counted. We emphasize that the setting of O_x needs to know the value T_x first.

For the objects I_x, T_x and O_x on the query x, there are only three cases in all tuples in the list.

$$(I_x, T_x, O_x) = (\mathcal{A}, c, k),\ (I_x, T_x, O_x) = (\mathcal{A}, \perp, \perp),\ (I_x, T_x, O_x) = (\mathcal{B}, \perp, \perp),$$

where $c \in [0, n]$ and $k \in [1, q]$.

Response of objects (U_x, z_x)**.** Let (I_x, T_x, O_x) be the response to the query x according to the above description. The simulator randomly chooses $z_x \in \mathbb{Z}_p$ and sets the response U_x to x according to the chosen (c^*, k^*) as follows.

$$U_x = H(x) = \begin{cases} (g^b)^{z_x} & \text{if } (T_x, O_x) = (c^*, k^*) \\ g^{z_x} & \text{otherwise} \end{cases}.$$

We use z_x to denote the secret for response to x. In the following, if the query x needs to be written as $x = m||\Sigma_m^i$, the corresponding secret will be rewritten as z_m^i.

Finally, the simulator adds the defined tuple $(x, I_x, T_x, O_x, U_x, z_x)$ for the new query x to the list. This completes the description of the hash query and its response.

For the tuple $(x, I_x, T_x, O_x, U_x, z_x)$, we have $H(x)^\alpha = U_x^a = (g^a)^{z_x}$ is computable by the simulator for any query x as long as $(T_x, O_x) \neq (c^*, k^*)$. Notice that when $(T_x, O_x) = (c^*, k^*)$, we have

$$H(x)^\alpha = U_x^a = g^{abz_x} = (g^{ab})^{z_x}.$$

For the tuple $(x, I_x, T_x, O_x, U_x, z_x)$, we use $m_{i,j}$ to denote the message in the query input x if $(T_x, O_x) = (i, j)$. We define

$$\mathcal{M}_i = \{m_{i,1}, m_{i,2}, \cdots, m_{i,q_i}\}$$

to be the message set with q_i messages, where $\mathcal{M}_i$ contains all messages in those tuples with $T_x = i$. According to the setting of oracle response, for those hash queries belonging to **Type i** for all $i \in [0, n]$, there have $n + 1$ message sets $\mathcal{M}_0,\ \mathcal{M}_1,\ \mathcal{M}_2,\ \cdots,\ \mathcal{M}_n$ at most to capture all messages in these queries.

Without knowing the signature Σ_m of m, the adversary must make hash queries $H(m), H(m||\Sigma_m^1), H(m||\Sigma_m^2), \cdots, H(m||\Sigma_m^n)$ in sequence because Σ_m^i in the query $m||\Sigma_m^i$ contains

$$H(m)^\alpha, H(m||\Sigma_m^1)^\alpha, \cdots,\ H(m||\Sigma_m^{i-1})^\alpha.$$

For a message before its signature query, the adversary could not query all $n + 1$ hash queries for this message. Therefore, we have the following inequality and subset relationship hold.

$$q_n \leq q_{n-1} \leq \cdots \leq q_2 \leq q_1 \leq q_0,\ \ \mathcal{M}_n \subseteq \mathcal{M}_{n-1} \subseteq \cdots \subseteq \mathcal{M}_2 \subseteq \mathcal{M}_1 \subseteq \mathcal{M}_0.$$

All queried messages mentioned above can be described in Table 3. Suppose the adversary can finally forge a valid signature on a message m^*. The adversary must at least make the hash query $H(m^*||\Sigma_{m^*}^n)$ in order to compute $H(m^*||\Sigma_{m^*}^n)^\alpha$, which guarantees $q_n \geq 1$. Since the number of hash queries is of at most q, we have $q_0 < q$.

The queried messages in $n + 1$ message sets fulfill the message set description in Section 4. Since $q_n \geq 1$ and $q_0 < q$, we have the **Range Lemma** and the **Probability**

Table 3. Messages queried by the adversary where $T_x \neq \perp$.

$\mathcal{M}_0 =$	$\{\ m_{0,1}\ ,\ m_{0,2}\ ,\ m_{0,3}\ ,\ \cdots\ ,\ \cdots\ ,\ \cdots\ ,\ \cdots\ ,\ m_{0,q_0}\ \}$
$\mathcal{M}_1 =$	$\{\ m_{1,1}\ ,\ m_{1,2}\ ,\ m_{1,3}\ ,\ \cdots\ ,\ \cdots\ ,\ \cdots\ ,\ m_{1,q_1}\ \}$
$\mathcal{M}_2 =$	$\{\ m_{2,1}\ ,\ m_{2,2}\ ,\ m_{2,3}\ ,\ \cdots\ ,\ \cdots\ ,\ m_{2,q_2}\ \}$
	$\cdots$
$\mathcal{M}_n =$	$\{\ m_{n,1},\ m_{n,2}\ ,\ m_{n,3}\ ,\ \cdots\ ,\ m_{n,q_n}\ \}$

Lemma can also be applied to the above message sets, even these message sets are adaptively generated by the adversary.

Signature-Query: For a signature query on the message m that is adaptively chosen by the adversary, the simulation is described as follows.

If m is never queried to the random oracle, the simulator works as follows from $i = 1$ to $i = n + 1$, where i is increased by one for each time.

- Add a query on $m||\Sigma_m^{i-1}$ to the list (We define $m||\Sigma_m^0 = m$). According to the setting of the random oracle simulation, we have the corresponding tuple is

$$(m||\Sigma_m^{i-1},\ \mathcal{B},\ \perp,\ \perp,\ g^{z_m^{i-1}},\ z_m^{i-1}).$$

- Compute the block signature σ_i as

$$\sigma_i = H(m||\Sigma_m^{i-1})^\alpha = (g^a)^{z_m^{i-1}}.$$

Notice that σ_i for all i are computable by the simulator and the signature of Σ_m is equal to $\Sigma_m^{n+1} = (\sigma_0, \sigma_1, \cdots, \sigma_{n+1})$. Therefore, the signature of m is computable by the simulator.

Suppose the message m is ever queried to the random oracle by the adversary, where the following queries associated with the message m are made by the adversary

$$m||\Sigma_m^0,\ m||\Sigma_m^1,\ m||\Sigma_m^2, \cdots,\ m||\Sigma_m^{r_m}:\ \ r_m \in [0, n].$$

Here, the integer r_m is adaptively decided by the adversary. Let $(x, I_x, T_x, O_x, U_x, z_x)$ be the tuple for $x = m||\Sigma_m^{r_m}$. That is, $T_x = r_m$.

- If $(T_x, O_x) = (c^*, k^*)$, the simulator aborts because

$$H(m||\Sigma_m^{r_m}) = g^{bz_x},\ \ \sigma_{r_m+1} = H(m||\Sigma_m^{r_m})^\alpha = U_x^a = (g^{bz_x})^a = (g^{ab})^{z_x},$$

  which cannot be computed by the simulator and hence the simulator fails in simulating the signature for the adversary, especially the block signature σ_{r_m+1}.
- Otherwise, $(T_x, O_x) \neq (c^*, k^*)$. Then, σ_{r_m+1} is computable by the simulator because

$$H(m||\Sigma_m^{r_m}) = g^{z_x},\ \ \sigma_{r_m+1} = H(m||\Sigma_m^{r_m})^\alpha = (g^a)^{z_x}.$$

Similarly to the case that m is never queried to the random oracle, the simulator can generate and make hash queries

$$H(m||\Sigma_m^{r_m+1}),\ \ H(m||\Sigma_m^{r_m+2}),\cdots,H(m||\Sigma_m^n)$$

to the random oracle. Finally, it computes the signature Σ_m for the adversary.

This completes the description of signature simulation. Once the simulator generates the signature of m, it forwards the signature to the adversary. It is easy to verify that the computed signature is a valid signature.

Forgery: The adversary outputs a valid signature Σ_{m^*} on a message m^* that is not asked for a signature. Since the adversary cannot make a signature query on m^*, we have the following queries to the random oracle were made by the adversary

$$m^*||\Sigma_{m^*}^0,\ m^*||\Sigma_{m^*}^1,\ m^*||\Sigma_{m^*}^2,\ \cdots,\ m^*||\Sigma_{m^*}^n.$$

The solution to the hard problem does not have to be associated with the forged message m^*. The simulator solves the hard problem as follows.

- The simulator searches $\mathcal{L}$ to find the first tuple (x,I_x,T_x,O_x,U_x,z_x) satisfying

  $$(T_x,O_x)=(c^*,k^*).$$

  If this tuple does not exist, abort. Otherwise, let the message m_{c^*,k^*} in this tuple be denoted by $\hat{m}$ for short. That is, $m_{c^*,k^*}=\hat{m}$ and we have $\hat{m}\in\mathcal{M}_{c^*}$. We note that $\hat{m}$ could be different from m^*. This tuple therefore is equivalent to

  $$(x,I_x,T_x,O_x,U_x,z_x)=\Big(\hat{m}||\Sigma_{\hat{m}}^{c^*},\ \mathcal{A},\ c^*,\ k^*,\ g^{bz_{\hat{m}}^{c^*}},\ z_{\hat{m}}^{c^*}\Big).$$

  That is $H(\hat{m}||\Sigma_{\hat{m}}^{c^*})=g^{bz_{\hat{m}}^{c^*}}$ contains the instance g^b.
- The simulator searches $\mathcal{L}$ to find the second tuple $(x',I_{x'},T_{x'},O_{x'},U_{x'},z_{x'})$, where x' is the query about the message $\hat{m}$ and $T_{x'}=c^*+1$. If this tuple does not exist, abort. Otherwise, we have $\hat{m}\in\mathcal{M}_{c^*+1}$ and

  $$x'=\hat{m}||\Sigma_{\hat{m}}^{c^*+1},$$

  where $\Sigma_{\hat{m}}^{c^*+1}$ contains $\sigma_{c^*+1}=H(m||\Sigma_m^{c^*})^\alpha$.
- The simulator computes and outputs

  $$\Big(H(\hat{m}||\Sigma_{\hat{m}}^{c^*})^\alpha\Big)^{\frac{1}{z_{\hat{m}}^{c^*}}}=\Big(g^{abz_{\hat{m}}^{c^*}}\Big)^{\frac{1}{z_{\hat{m}}^{c^*}}}=g^{ab}$$

  as the solution to the CDH problem.

Analysis. This completes the description of simulation and solution. The scheme simulation is indistinguishable from the real scheme because the signing key α is simulated using the random exponent a in the instance and there is no random number in the signature simulation. The random oracle response is correct because the response U_x to each query x is computed using a random integer z_x, independent of other integers.

According to the assumption, the adversary will break the signature scheme with advantage ϵ. We have the adversary will make the hash query $H(m^*||\Sigma^n_{m^*})$ with probability at least ϵ,[8] such that $m^* \in \mathcal{M}_n$ and hence $q_n \geq 1$. The number of hash query is q. Since $q_0 + q_1 + q_2 + \cdots + q_n = q$, we have $q_0 < q$. Therefore, the conditions of the **Probability Lemma** hold with success probability ϵ in our reduction.

The reduction is successful when the simulator does not abort in the query phase and the forgery phase. According to the setting of the simulation, we found

- The simulator aborts in the query phase when $m_{c^*,k^*} \in \mathcal{M}_{c^*}$ but $m_{c^*,k^*} \notin \mathcal{M}_{c^*+1}$, because the simulator cannot compute the queried signature on the message m_{c^*,k^*}.
- The simulator aborts in the forgery phase when $m_{c^*,k^*} \notin \mathcal{M}_{c^*}$ because the simulator cannot embed g^b in the response to the query m_{c^*,k^*} under **Type** c^*, or when $m_{c^*,k^*} \notin \mathcal{M}_{c^*+1}$ because the solution to the CDH problem does not exist in the hash list.

Therefore, we have the reduction is successful when $m_{c^*,k^*} \in \mathcal{M}_{c^*}$ and $m_{c^*,k^*} \in \mathcal{M}_{c^*+1}$. According to the **Probability Lemma**, if $q_0 < q$ and $q_n \geq 1$, we have $\hat{m} = m_{c^*,k^*} \in \mathcal{M}_{c^*} \cap \mathcal{M}_{c^*+1}$ holds with probability $1/(nq^{\frac{1}{n}})$. Therefore, the reduction is successful and the simulator can solve the hard problem with success probability at least $\epsilon/(nq^{\frac{1}{n}})$.

The simulation time is mainly dominated by the signature simulation, where all signature computations cost $O(n \cdot q_s)$ point multiplications. Notice that the simulation time does not consider the time cost of oracle responses. This completes the proof. □

Very recently, the work of [13] showed how to find a correct solution to a computational hard problem from hash queries for encryption schemes under the indistinguishability model. The loss factor is also $nq^{\frac{1}{n}}$ with a similar chain-like structure in hash queries. However, we stress that the approaches towards tight reductions are totally different.

Remark 3 *In the above probability analysis, we use the condition that the adversary can forge a valid signature to guarantee $q_n \geq 1$, which is a desired condition in the* **Probability Lemma**. *We note that this condition is sufficient but not necessary in our reduction as long as $q_n \geq 1$ holds.*

6 Conclusion

Optimal security reductions with impossibility results of tight reductions for unique signatures and efficiently re-randomizable signatures have been well studied in the literature (Coron, Eurocrypt 2002; Hofheinz *et al.*, PKC 2012; Bader *et al.*, Eurocrypt

[8] The adversary can forge a valid signature via randomly choosing group elements as the forged signature without making the corresponding hash queries. However, the success probability is less than $1/p$. The actual probability of making such a hash query is $\epsilon - 1/p$. Since $1/p$ is negligible compared to ϵ, we simplify the probability $\epsilon - 1/p$ into ϵ.

2016). It has been proved and claimed that any security reduction for a unique signature scheme or an efficiently re-randomizable signature scheme must loose a factor of at least q_s for q_s number of signature queries under a non-interactive assumption in the EU-CMA security model. In this work, we pointed out that their optimal security reductions cannot cover all security reductions and proposed a counterexample with a tight reduction in the random oracle model. We can program the security reduction with a very small loss factor, e.g., 100, for 2^{50} hash queries to random oracles under the CDH assumption in the EU-CMA security model. We stress that our counterexample just bypasses the given proofs of optimal security reduction, because the proofs via meta-reduction break down in our query-based reduction. The given counterexample implies a new way of constructing a signature scheme with a tight reduction in the random oracle model from a signature scheme with a loose reduction. This transformation is universal and also applicable for unique signatures. This transformation method is somewhat impractical because of inefficiency due to the growth of signature length by n times. Nevertheless, this is the first method for enabling a tight proof without the use of a random salt in the signature generation.

Acknowledgment

We would like to thank Yannick Seurin for his helpful comments to improve the clarity of this paper. We would also like to thank Tibor Jager for his insightful comments especially for helping identify the gap between the proofs of impossibilities and our example. Finally, we would like to thank anonymous reviewers of CRYPTO 2017 for their insightful comments which help us improve the quality of this work. This work was partially supported by ARC Discovery Early Career Researcher Award (DECRA) DE170100641.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer (2001)
2. Abdalla, M., Fouque, P., Lyubashevsky, V., Tibouchi, M.: Tightly-secure signatures from lossy identification schemes. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 572–590. Springer (2012)
3. Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 273–304. Springer (2016)
4. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and rabin. In: Maurer, U.M. (ed.) EUROCRYPT '96. LNCS, vol. 1070, pp. 399–416. Springer (1996)
5. Bernstein, D.J.: Proving tight security for rabin-williams signatures. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 70–87. Springer (2008)
6. Blazy, O., Kakvi, S.A., Kiltz, E., Pan, J.: Tightly-secure signatures from chameleon hash functions. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 256–279. Springer (2015)

7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer (2001)
8. Boyd, C., Carr, C.: Fair client puzzles from the bitcoin blockchain. In: Liu, J.K., Steinfeld, R. (eds.) ACISP 2016. LNCS, vol. 9722, pp. 161–177. Springer (2016)
9. Chevallier-Mames, B., Joye, M.: A practical and tightly secure signature scheme without hash function. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 339–356. Springer (2006)
10. Coron, J.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer (2002)
11. Goh, E., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the diffie-hellman problems. J. Cryptology 20(4), 493–514 (2007)
12. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
13. Guo, F., Susilo, W., Mu, Y., Chen, R., Lai, J., Yang, G.: Iterated random oracle: A universal approach for finding loss in security reduction. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 745–776 (2016)
14. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer (2012)
15. Hofheinz, D., Jager, T., Knapp, E.: Waters signatures with optimal security reduction. In: Fischlin, M., Buchmann, J.A., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 66–83. Springer (2012)
16. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 537–553. Springer (2012)
17. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) CCS 2003. pp. 155–164. ACM (2003)
18. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM Conference on Computer and Communications Security 2016. pp. 17–30. ACM (2016)
19. Schäge, S.: Tight proofs for signature schemes without random oracles. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 189–206. Springer (2011)
20. Zhang, F., Safavi-Naini, R., Susilo, W.: An efficient signature scheme from bilinear pairings and its applications. In: Bao, F., Deng, R.H., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 277–290. Springer (2004)