Optimal Tightness for Chain-Based Unique Signatures

Fuchun Guo[0000−0001−6939−7710] and Willy Susilo[0000−0002−1562−5105]

Institute of Cybersecurity and Cryptology (iC^2)
School of Computing and Information Technology
University of Wollongong, Wollongong, NSW, Australia
{fuchun,wsusilo}@uow.edu.au

Abstract. Unique signatures are digital signatures with exactly one unique and valid signature for each message. The security reduction for most unique signatures has a natural reduction loss (in the existentially unforgeable against chosen-message attacks, namely EUF-CMA, security model under a non-interactive hardness assumption). In Crypto 2017, Guo *et al.* proposed a particular chain-based unique signature scheme where each unique signature is composed of n BLS signatures computed sequentially like a blockchain. Under the computational Diffie-Hellman assumption, their reduction loss is $n \cdot q_H^{1/n}$ for q_H hash queries and it is logarithmically tight when $n = \log q_H$. However, it is currently unknown whether a better reduction than logarithmical tightness for the chain-based unique signatures exists.

We show that the proposed chain-based unique signature scheme by Guo *et al.* must have the reduction loss $q^{1/n}$ for q signature queries when each unique signature consists of n BLS signatures. We use a meta reduction to prove this lower bound in the EUF-CMA security model under any non-interactive hardness assumption, and the meta-reduction is also applicable in the random oracle model. We also give a security reduction with reduction loss $4 \cdot q^{1/n}$ for the chain-based unique signature scheme (in the EUF-CMA security model under the CDH assumption). This improves significantly on previous reduction loss $n \cdot q_H^{1/n}$ that is logarithmically tight at most. The core of our reduction idea is a *non-uniform* simulation that is specially invented for the chain-based unique signature construction.

Keywords: Unique Signatures · Optimal reduction.

1 Introduction

A digital signature scheme is a unique signature scheme if there exists a unique and valid signature for each message [16, 34, 5]. That is, for any message, we cannot find two different signatures that are both valid for that message. Unique signatures prohibit the use of randomness in the signature generation.

It is non-trivial to construct a digital signature scheme that is *tightly secure* in the existentially unforgeable against chosen-message attacks (EUF-CMA) security model under a non-interactive hardness assumption. Intensive research

such as [7, 43, 26, 38, 10, 8, 49, 33, 1, 37, 42, 9, 6, 13, 32, 39, 27, 3, 24, 36, 25, 4, 19, 14, 18, 17, 28, 48] have been conducted in this security model or the more advanced multi-user setting model. Most methods must employ randomness in signature generations and are therefore not suitable for unique signatures.

It looks "paradoxical" when proving tight security for a unique signature scheme (in the EUF-CMA model under a non-interactive hardness assumption). Taking the BLS (unique) signature by Boneh *et al.* [11] as an example: upon receiving the public key (g, g^{α}), the adversary might first query messages to random oracle to know $H(m_1), H(m_2), \cdots, H(m_q)$. For each signature $H(m_i)^{\alpha}, i \in [1, q]$, it must be either simulatable (the signature is signable by the simulator) or reducible (the signature is unsignable and problem solution can be extracted from the signature) and it cannot be switched. After receiving the public key and all responses to hash queries, the adversary first picks $q-1$ random messages out of q for their signature queries, and forges the signature on the remaining message. It has been proved in [16, 34, 5] that the probability of successfully reduction for BLS-like unique signatures is at most $1/q$.

So far, the only known tight security method for unique signature[1] was proposed by Guo *et al.* [27] in Crypto 2017. They constructed a chain-based unique signature scheme (see Subsection 1.2), where BLS signatures are generated, hashed into messages, and then signed again like a blockchain. Each unique signature has n BLS signatures as block signatures. With this signature structure, they can program the reduction tightly because the simulator will already solve the hard problem before each unique signature is "committed" into simulatable or reducible. In their tight reduction, the adversary must generate and make a special hash query that carries the CDH (Computational Diffie-Hellman) problem solution to the random oracle with probability at least $1/(n \cdot q_H^{1/n})$ if the adversary can successfully forge a signature after making q_H hash queries. This chain-based method was later adopted to construct tightly secure and short unique signatures from RSA signatures by Shacham in [51].

Even though our community has rich methods of tight reduction for digital signatures and other primitives such as the recent results for key exchange [25, 15, 31, 35], the only tightness method applicable to unique signatures is the chain-based construction[2]. However, it is currently unknown whether there exists a better reduction than [27] for the chain-based unique signatures.

1.1 Our Contributions

In this paper, we analyze the optimal tightness of reductions for the chain-based unique signature scheme in [27] and then propose a reduction with optimal tightness for this scheme.

[1] In 2012, Kakvi and Kiltz [37] introduced a conceptual level RSA-FDH scheme with unique signatures and a tight security reduction.

[2] We meant reductions against general adversaries. It is worth noting that BLS-like unique signatures can be proved tight security in the Algebraic Group Model [22] when adversaries are restricted in algebraic operations.

We show that any reduction proof for the chain-based unique signature scheme must have a reduction loss of at least $q^{1/n}$ for q signature queries if each unique signature has n BLS block signatures. This optimal analysis is also applicable in the random oracle model and is proved via meta-reduction under any non-interactive computationally hardness assumption in the EUF-CMA security model. The given optimal analysis indicates that it is necessary to generate a chain-based unique signature having $n = \log(q)$ block signatures in order to obtain tight security. The chain-based unique signature scheme is actually the BLS scheme when $n = 1$ and our corresponding result is in line with the negative results in [16, 34, 5].

We propose a completely different security reduction for the chain-based BLS scheme in length n with optimal tightness. The core of our reduction idea is a non-uniform programming that perfectly suits the chain-based construction. Our reduction loss is at most $4 \cdot q^{1/n}$ for q signature queries under the CDH assumption in the EUF-CMA security model (using random oracles). This improves significantly on previous $n \cdot q_H^{1/n}$ in [27], because the previous result is logarithmically tight only when $n = \log q_H$ while ours is fully tight when $n = \log q$. Our fully tight reduction does not require to increase the length of signatures (depending on n) because $q \leq q_H$ is always true. In particular, the signatures in our reduction have the same size as in [27] when $q \approx q_H$, and are much shorter than [27] when $q << q_H$ such as $q = 2^{30}$ and $q_H = 2^{100}$.

Our results are also applicable to the Shacham's tightly secure and short RSA unique signatures [51]. Our optimal analysis is general and also applicable to this unique signature scheme. The reduction loss is reduced from logarithmically large to constant, and the computational efficiency is improved because of the decrease of length n when $q << q_H$. The details will be given in the full version.

1.2 Technical Idea

We first review the chain-based BLS scheme proposed in [27] as follows.

KeyGen: Let $(\mathbb{G}, \mathbb{G}_T, p, e, g)$ be a bilinear pairing. The key generation algorithm chooses a random integer $\alpha \in \mathbb{Z}_p$ and a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}$ that will be viewed as a random oracle in the security proof. It computes $h = g^\alpha$ and chooses an integer n as the scheme parameter. The public key pk is $(\mathbb{G}, \mathbb{G}_T, p, e, g, H, h, n)$, and the secret key sk is α.

Sign: The signing algorithm takes as input a message $m \in \{0,1\}^*$ and the key pair (pk, sk). It computes the signature $\Sigma_m = (\sigma_1, \sigma_2, \cdots, \sigma_n)$ on m as

$$(\sigma_1, \sigma_2, \sigma_3, \cdots, \sigma_n) = \Big(H(m|\Sigma_m^0)^\alpha,\ H(m|\Sigma_m^1)^\alpha,\ H(m|\Sigma_m^2)^\alpha, \cdots, H(m|\Sigma_m^{n-1})^\alpha\Big),$$

where σ_i for all $i \in [1, n]$ is called block signature, $\Sigma_m^0 = (\ )$, and $\Sigma_m^i = (\sigma_1, \sigma_2, \cdots, \sigma_i)$. The final signature Σ_m on m is Σ_m^n.

Note: To be able to distinguish messages and signatures, we must include the symbols " $|$ " and brackets "()" as part of hash inputs. In particular, $m|\Sigma_m^0 = m|(\ )$.

Verify: The verification algorithm takes as input the public key pk, a message m, and its signature $\Sigma_m = (\sigma_1, \sigma_2, \cdots, \sigma_n)$. It accepts the signature if

$$e\Big(\sigma_{i+1}, g\Big) = e\Big(H(m|\Sigma_m^i), h\Big) : \text{ for all } i \in [0, n-1].$$

In the security reduction for the chain-based BLS signature in the random oracle model, a hash query $x = m|\Sigma_m^i$ to H is called type-i query of m. To forge a valid signature on m^*, the adversary must make the type-0 query of m^*, compute $\Sigma_{m^*}^1$, make the type-1 query of m^*, compute $\Sigma_{m^*}^2$ and so on until make the type-$(n-1)$ query of m^*, and compute $\Sigma_{m^*}^n$ as the forged signature.

The chain-based construction enables the simulator to solve the CDH problem with the adversary's hash queries. Given a problem instance (g, g^a, g^b), if $g^\alpha = g^a$ and the type-i query of m is responded with $H(m|\Sigma_m^i) = g^b$, the type-$(i+1)$ query of m generated and made by the adversary contains $\sigma_{i+1} = H(m|\Sigma_m^i)^\alpha = g^{ab}$ that is the solution to the CDH problem. It is worth noting that the hash query used to solve the hard problem does not have to the query of m^* but can be the query of any message generated by the adversary.

Optimal Analysis of the Chain-Based BLS Scheme. We prove that any security reduction $\mathcal{R}$ for the chain-based BLS scheme in the EUF-CMA security model under any non-interactive computationally hard assumption must be bounded with success probability $1/q^{\frac{1}{n}}$. Otherwise, we construct a meta-reduction $\mathcal{B}$ to break this hardness assumption by following the meta-reduction framework given by Coron in [16], which is described as follows.

- We construct a special hypothetical adversary that can break the chain-based BLS scheme with probability $\epsilon_{\mathcal{A}}$. When interacting with such a hypothetical adversary, $\mathcal{R}$ would break the hardness assumption with probability $\epsilon_{\mathcal{R}}$.
- We simulate this hypothetical adversary via rewinding $\mathcal{R}$. When interacting with the simulated adversary, if we can efficiently simulate the hypothetical adversary except with error probability ϵ_E, $\mathcal{R}$ would break the hardness assumption with probability $\epsilon_{\mathcal{R}} - \epsilon_E$.
- The meta-reduction therefore shows that $\epsilon_{\mathcal{R}}$ must be not larger than ϵ_E. Otherwise, we can run $\mathcal{R}$ as an oracle to break the hardness assumption.

Based on the Coron's framework, our optimal analysis is to show how a hypothetical adversary will possibly attack and how to simulate this hypothetical adversary successfully except with error probability $\epsilon_E = \epsilon_{\mathcal{A}}/q^{\frac{1}{n}}$.

The Hypothetical Adversary. A hypothetical adversary might make hash queries and signature queries in the sequence $\mathcal{T}_0(\mathcal{M}_0) \rightarrow \mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) \rightarrow \mathcal{T}_1(\mathcal{M}_1) \rightarrow \mathcal{S}(\mathcal{M}_1 \setminus \mathcal{M}_2) \rightarrow \cdots \rightarrow \mathcal{S}(\mathcal{M}_{n-1} \setminus \mathcal{M}_n) \rightarrow \mathcal{T}_n(\mathcal{M}_n)$ as long as $\mathcal{R}$ does not fail in responding to queries. The queries are explained as follows.

- $\mathcal{M}_0, \mathcal{M}_1, \cdots, \mathcal{M}_n$ are $n+1$ message sets satisfying that the subset relationship $\mathcal{M}_0 \supset \mathcal{M}_1 \supset \mathcal{M}_1 \supset \mathcal{M}_2 \supset \cdots \supset \mathcal{M}_n$ holds and the set $\mathcal{M}_i$ has $q^{1-\frac{i}{n}}$ messages (we simply treat $q^{\frac{1}{n}}$ as an integer). All messages in the these message sets are randomly chosen by the adversary. In particular, we have $|\mathcal{M}_0| = q$ and $|\mathcal{M}_n| = 1$. We define $\mathcal{M}_i \setminus \mathcal{M}_{i+1}$ to be the set of messages in $\mathcal{M}_i$ but not in $\mathcal{M}_{i+1}$.
- $\mathcal{T}_i(\mathcal{M}_i)$ is the set of type-i queries of all messages in $\mathcal{M}_i$.
- $\mathcal{S}(\mathcal{M}_i \setminus \mathcal{M}_{i+1})$ is the set of signature queries on all messages in $\mathcal{M}_i \setminus \mathcal{M}_{i+1}$.

That is, the (computationally unbounded) adversary first generates type-0 queries of all messages in $\mathcal{M}_0$ and submits all of them to the random oracle. Upon receiving all responses to hash queries, the adversary makes the signature queries on all messages in $\mathcal{M}_0 \setminus \mathcal{M}_1$. If $\mathcal{R}$ aborts, the adversary stops. Otherwise, the adversary generates type-1 queries of all messages in $\mathcal{M}_1$ and repeats the above queries and computations until $\mathcal{T}_n(\mathcal{M}_n)$. Suppose $m^* \in \mathcal{M}_n$. After $\mathcal{T}_n(\mathcal{M}_n)$, the adversary has already generated the type-n query of m^*, namely $m^*|\Sigma^n_{m^*}$. The type-n query implies the signature on m^*. Therefore, at the end of the above query sequence, the adversary can easily return $\Sigma_{m^*} = \Sigma^n_{m^*}$ as the forged signature on a new message m^* whose signature was not queried.

The Simulated Adversary. It is easy to simulate the hash queries $\mathcal{T}_0(\mathcal{M}_0)$ because all hash queries are type-0 queries (namely $m|()$ for $m \in \mathcal{M}_0$) and composed of messages only. The challenge of simulating the hypothetical adversary is to generate and make hash queries $\mathcal{T}_i(\mathcal{M}_i)$ for all $i \in [1, n]$ because the type-i query of m ($m|\Sigma^i_m$) contains block signatures $\Sigma^i_m = (\sigma_1, \sigma_2, \cdots, \sigma_i)$ that cannot be efficiently simulated without knowing the secret key.

We simulate the hypothetical adversary with a rewinding argument. Different from the original meta-reduction in [16], we need to rewind at most n times in order to successfully simulate the hypothetical adversary. The rewinding works as follows from $i = 1$ to $i = n$.

$$\begin{array}{ccccccccc} & & \text{1-st rewind} & & & \text{2-nd rewind} & & & \text{n-th rewind} \\ & \nearrow & \mathcal{S}(\mathcal{M}_1) & & \nearrow & \mathcal{S}(\mathcal{M}_2) & & \nearrow & \mathcal{S}(\mathcal{M}_n) \\ \mathcal{T}_0(\mathcal{M}_0) \to & & \mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) & \to \mathcal{T}_1(\mathcal{M}_1) \to & & \mathcal{S}(\mathcal{M}_1 \setminus \mathcal{M}_2) & \to \cdots \to & & \mathcal{S}(\mathcal{M}_{n-1} \setminus \mathcal{M}_n) \to \mathcal{T}_n(\mathcal{M}_n) \end{array}$$

- Before the i-th time rewind, we first make signature queries $\mathcal{S}(\mathcal{M}_i)$ to $\mathcal{R}$ to obtain signatures $\Sigma_{\mathcal{M}_i} = \{\Sigma^n_m : m \in \mathcal{M}_i\}$.
- Then we rewind $\mathcal{R}$ (the i-th time rewind) to the state it was after the hash queries $\mathcal{T}_{i-1}(\mathcal{M}_{i-1})$. This time, we make signature queries $\mathcal{S}(\mathcal{M}_{i-1} \setminus \mathcal{M}_i)$. We can continue the simulation of making hash queries $\mathcal{T}_i(\mathcal{M}_i) = \{m|\Sigma^i_m : m \in \mathcal{M}_i\}$ with the help of $\Sigma_{\mathcal{M}_i}$ if $\mathcal{R}$ does not abort before the rewind.

The Error Probability. If $\mathcal{R}$ does not abort, the simulated adversary must continue to complete the queries $\mathcal{T}_0(\mathcal{M}_0) \to \mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) \to \mathcal{T}_1(\mathcal{M}_1) \to \mathcal{S}(\mathcal{M}_1 \setminus \mathcal{M}_2) \to \cdots \to \mathcal{S}(\mathcal{M}_{n-1} \setminus \mathcal{M}_n) \to \mathcal{T}_n(\mathcal{M}_n)$. We note that if $\mathcal{R}$ cannot respond to one of signature queries in this query sequence, the simulation on the hypothetical adversary is still successful because the reduction is aborted by $\mathcal{R}$. We claim that the error happens when there exists an integer $i^\# \in [1, n]$ such that

– $\mathcal{R}$ cannot respond to queries $\mathcal{S}(\mathcal{M}_{i^\#})$ before the $i^\#$-th time rewind, but
– $\mathcal{R}$ can respond to queries $\mathcal{S}(\mathcal{M}_{i^\#-1} \setminus \mathcal{M}_{i^\#})$ after the $i^\#$-th time rewind.

The simulation on the hypothetical adversary fails because the simulated adversary must continue to make the type-$i^\#$ queries $\mathcal{T}_{i^\#}(\mathcal{M}_{i^\#})$, but the simulated adversary does not have $\Sigma_{\mathcal{M}_{i^\#}}$ to simulate these type-$i^\#$ queries.

Let $\mathcal{S}(\mathcal{M}) = 1$ denote that $\mathcal{R}$ can simulate all signature queries on $\mathcal{M}$ and $\mathcal{S}(\mathcal{M}) = 0$ denote the opposite case. When the hypothetical adversary attacks the scheme, no matter what $i \in [1, n]$ is, we have

$$\frac{|\mathcal{M}_i|}{|\mathcal{M}_{i-1}|} = \frac{q^{1-\frac{i}{n}}}{q^{1-\frac{i-1}{n}}} = \frac{1}{q^{\frac{1}{n}}}.$$

Then the error probability will be the same no matter what $i^\# \in [1, n]$ is during the reduction.

Taking $i^\# = 1$ as the example. After the type-0 queries of all messages in $\mathcal{M}_0$ namely $\mathcal{T}_0(\mathcal{M}_0)$, it means that $\mathcal{R}$ cannot compute $H(m|\Sigma_m^0)^\alpha$ for some $m \in \mathcal{M}_0$, while $\mathcal{R}$ can respond to signature queries $\mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1)$ meaning that $m \notin \mathcal{M}_0 \setminus \mathcal{M}_1$. Since the message set $\mathcal{M}_1$ is randomly chosen, we prove that

$$\Pr\left[\mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) = 1 \middle| \mathcal{S}(\mathcal{M}_0) = 0, \frac{|\mathcal{M}_1|}{|\mathcal{M}_0|} = \frac{1}{q^{\frac{1}{n}}}\right] \leq \frac{1}{q^{\frac{1}{n}}}.$$

The above probability is in line with [16, 34, 5] when $\mathcal{M}_1$ has one message only. Our case needs to consider multiple messages in $\mathcal{M}_1$.

With the above analysis, we shall prove that the error probability of simulating the hypothetical adversary is at most $\epsilon_{\mathcal{A}}/q^{\frac{1}{n}}$. This completes the intuitive observation of our optimal analysis.

Optimal Tightness of the Chain-Based BLS Scheme. The security of the chain-based BLS scheme is based on the CDH hard assumption where it is hard to compute $g^{\hat{a}\hat{b}}$ from $(g, g^{\hat{a}}, g^{\hat{b}})$. In the proof, the simulator sets $\alpha = \hat{a}$ and controls the random oracle. How to respond to each hash query in the random oracle model is the core for obtaining a tight reduction.

Classifications of Hash Queries. All hash queries to the random oracle will be classified into two types called **Normal Query** and **Challenge Query** according to the ways of response by the simulator.

– **Normal Query**. A hash query x is called a normal query if the simulator sets $H(x) = g^z$ in response where $z \in \mathbb{Z}_p$ is randomly chosen by the simulator for the query x. Then $H(x)^\alpha = (g^{\hat{a}})^z$ is computable by the simulator.
– **Challenge Query**. A hash query x is called a challenge query if the simulator sets $H(x) = g^{\hat{b}+z}$ in response, where $z \in \mathbb{Z}_p$ is randomly chosen by the simulator for the query x. Then $H(x)^\alpha = g^{\hat{a}\hat{b}+\hat{a}z}$ and the CDH problem solution $g^{\hat{a}\hat{b}}$ can be extracted from $H(x)^\alpha$ by computing $H(x)^\alpha/(g^{\hat{a}})^z$.

Most importantly, suppose that the type-i query of m, namely $m|\Sigma_m^i$, is set as the challenge query and the adversary makes the type-$(i+1)$ query of m, denoted by $m|\Sigma_m^{i+1}$. We have that $\Sigma_m^{i+1} = (\sigma_1, \sigma_2, \cdots, \sigma_{i+1})$ and the block signature $\sigma_{i+1} = H(m|\Sigma_m^i)^\alpha = g^{\hat{a}\hat{b}+\hat{a}z}$ carries the CDH problem solution.

The Idea in [27]. Let q_H be the number of hash queries. The authors proved that no matter how the adversary adaptively makes hash queries and signature queries in the EUF-CMA security model, there exists an integer $i^* \in [0, n-1]$ such that

- The number of type-i^* queries is not more than $(q_H)^{1-\frac{i^*}{n}}$.
- The number of type-(i^*+1) queries is larger than $(q_H)^{1-\frac{i^*+1}{n}}$.

The integer i^* is dependent on how the adversary adaptively makes hash queries.

In [27], the simulator randomly picks an integer $c^* \in [0, n-1]$ and an integer $l^* \in [1, (q_H)^{1-\frac{c^*}{n}}]$. Then the simulator will set the l^*-th new type-c^* query (of any message m) generated and made by the adversary as the challenge query. When $c^* = i^*$, their proof result guarantees that the adversary will generate and make the type-(c^*+1) query of the same message m with probability

$$\frac{(q_H)^{1-\frac{i^*+1}{n}}}{(q_H)^{1-\frac{i^*}{n}}} = \frac{1}{(q_H)^{\frac{1}{n}}}.$$

Therefore, their reduction loss is $n \cdot (q_H)^{1/n}$ for the chain-based BLS scheme. The features of this logarithmically tight reduction are summarized as follows.

- **Single Challenge.** Only one of q_H hash queries is set as the challenge query. All other hash queries are set as normal queries by the simulator.
- **Uniform Choice.** The simulator will set one type-c^* query as the challenge query, and $c^* \in [0, n-1]$ is uniformly chosen to capture the success probability $\Pr[c^* = i^*] = \frac{1}{n}$ for any adaptive i^* decided by the adversary.
- **Static Setting.** In this reduction, the integers c^* and l^* are chosen by the simulator before the start of hash queries. Which hash query will be set as the challenge query is therefore static.

Our Main Idea. The Single-Uniform-Static approach in [27] (same as in [51]) is based on a natural rule of queries from the adversary. We invent a completely new approach called Multiple-Non-Uniform-Dynamic approach. This approach will allow the simulator to control the simulation such that the probability of success reduction will be increased when the adversary makes more hash queries.

- **Multiple Challenges.** For every message m, the simulator will choose an independent integer $c_m \in [0, n-1]$ and set the type-c_m query of m, denoted by $m|\Sigma_m^{c_m}$, as a challenge query. Then, the type-(c_m+1) query of m carries the problem solution. The integer c_m for m will be chosen when the adversary makes the type-0 query of m. The number of challenge queries is therefore multiple and depends on how many messages are involved in all hash queries.

- **Non-Uniform Choice.** We choose $c_m \in [0, n-1]$ in a non-uniform way. That is, c_m is not uniformly distributed in $[0, n-1]$. In our formal reduction description, we give a general approach of choosing c_m for any scheme parameter n. Here we give a specific choice for $n = \log(q)$ and set

$$\Pr[c_m = i] = \frac{2^i}{2 \cdot 2^n} = \frac{1}{2 \cdot 2^{n-i}}.$$

  That is, for each message m, the challenge query will be set at its type-0 query with probability $1/2^{n+1}$, at its type-1 query with probability $2/2^{n+1}$, and at its type-i query with probability $2^i/2^{n+1}$. It is not hard to achieve this non-uniformity[3].
- **Dynamic Setting.** Generally speaking, the adversary makes type-0 query, type-1 query, and so on until type-k_i query of m_i before signature query on m_i for an adaptive integer $k_i \in [0, n-1]$. If $k_i < c_{m_i}$, it means that no hash query of m_i has yet been set as a challenge query by the simulator. To enable signature simulation, upon receiving the signature query on m_i, the simulator will change $c_{m_i} = \infty$ such that all hash queries of m_i will be set as normal queries and the signature on m_i is computable by the simulator.

This completes the description of our approach. We define that the adversary adaptively chooses $k_i \in [0, n-1]$ and makes type-0 query, type-1 query, and so on until type-k_i query of m_i, denoted by $\mathcal{H}^{k_i}(m_i) = \{m_i|\Sigma^0_{m_i}, m_i|\Sigma^1_{m_i}, \cdots, m_i|\Sigma^{k_i}_{m_i}\}$, before the signature query on m_i, denoted by $\mathcal{S}(m_i)$. We define that the adversary will query the signatures on messages $(m_1, m_2, \cdots, m_q)$ before forging the signature on m^*. According to the setting, we have:

- If $c_{m_i} < k_i$, the simulator can solve the hard problem with type-$(c_{m_i}+1)$ query of m_i from the adversary according to the truth of $c_{m_i} + 1 \leq k_i$ and the setting of multiple challenges.
- If $c_{m_i} = k_i$, namely the type-k_i query of m_i is set as the challenge query, the simulator has to abort.
- If $c_{m_i} > k_i$, the simulator can simulate the signature according to the dynamic setting.

We cannot directly analyze how the simulator solves the CDH problem because it depends on the adversary's adaptive choice of k_i and the simulator's setting parameter c_{m_i}. What we do instead is to prove **the lower bound of probability of successful reduction.** We are going to prove that

$$\frac{1}{4} \approx \Pr[\mathcal{Q}^*_{2,0}] \leq \Pr[\mathcal{Q}^*_{2,1}] \leq \Pr[\mathcal{Q}^*_{2,2}] \leq \cdots \leq \Pr[\mathcal{Q}^*_{2,q}] \leq \Pr[\mathcal{Q}^*_{\mathcal{A}}],$$

where $\mathcal{Q}^*$ is the query sequence (mixture of hash queries and/or signature queries) made by the adversary $\mathcal{A}$, $\Pr[\mathcal{Q}^*]$ is the success probability of reduction under the query sequence $\mathcal{Q}^*$, $\mathcal{Q}^*_{\mathcal{A}}$ is the real query sequence launched by the adversary during attacks, and all query sequences are defined in Table 1.

[3] To implement such a non-uniform choice, we firstly randomly choose an integer $w \in [1, 2^{n+1}]$. Then we find the integer i satisfying $2^i \leq w < 2^{i+1}$ and set $c_m = i$. It is not hard to verify that $\Pr[w \leftarrow_R [1, 2^{n+1}] : 2^i \leq w < 2^{i+1}] = 2^i/2^{n+1} = \Pr[c_m = i]$.

Table 1. The defined query sequences. $\mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1)$ means that the adversary will query $\mathcal{H}^{k_1}(m_1)$ first and then query $\mathcal{S}(m_1)$. The differences between two neighbor queries have been highlighted in the same color and $\mathcal{H}^{k_i}(m_i) = \{m_i|\Sigma^0_{m_i}, m_i|\Sigma^1_{m_i}, \cdots, m_i|\Sigma^{k_i}_{m_i}\}$. If the adversary can forge and return the signature $\Sigma_{m^*} = \Sigma^n_{m^*}$ on m^*, it implies that the adversary is able to generate and make hash queries $\mathcal{H}^n(m^*) = \{m^*|\Sigma^0_{m^*}, m^*|\Sigma^1_{m^*}, \cdots, m^*|\Sigma^n_{m^*}\}$.

$\mathcal{Q}^*_{2,0}$:	$\mathcal{H}^{0}(m_1) \to \mathcal{S}(m_1) \to \mathcal{H}^{0}(m_2) \to \mathcal{S}(m_2) \to \cdots \to \mathcal{H}^0(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*)$
$\mathcal{Q}^*_{2,1}$:	$\mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1) \to \mathcal{H}^{0}(m_2) \to \mathcal{S}(m_2) \to \cdots \to \mathcal{H}^0(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*)$
$\mathcal{Q}^*_{2,2}$:	$\mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1) \to \mathcal{H}^{k_2}(m_2) \to \mathcal{S}(m_2) \to \cdots \to \mathcal{H}^0(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*)$
$\vdots$	$\vdots$
$\mathcal{Q}^*_{2,q-1}$:	$\mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1) \to \mathcal{H}^{k_2}(m_2) \to \mathcal{S}(m_2) \to \cdots \to \mathcal{H}^{0}(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*)$
$\mathcal{Q}^*_{2,q}$:	$\mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1) \to \mathcal{H}^{k_2}(m_2) \to \mathcal{S}(m_2) \to \cdots \to \mathcal{H}^{k_q}(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*)$
$\mathcal{Q}^*_{\mathcal{A}}$:	Real query sequence includes $\mathcal{H}^{k_1}(m_1), \mathcal{H}^{k_2}(m_2), \cdots, \mathcal{H}^{k_q}(m_q), \mathcal{H}^n(m^*)$

Now we prove the above inequalities of probabilities step by step.

♠(Step 1) Suppose the adversary can forge a signature without signature queries and the adversary's query sequence is denoted by $\mathcal{Q}^*_1 = \mathcal{H}^n(m^*)$. According to the non-uniform parameter $c_{m^*} \in [0, n-1]$ satisfying $\Pr[c_{m^*} = i] = \frac{1}{2\cdot 2^{n-i}}$ and the adversary will make the type-0 query, the type-1 query and so on until the type-n query of m^*, we immediately have

$$\Pr[\mathcal{Q}^*_1] = \Pr[c_{m^*} \leq n-1] = \sum_{i=0}^{n-1} \Pr[c_{m^*} = i] = \frac{1}{2\cdot 2^n} + \frac{1}{2\cdot 2^{n-1}} + \cdots + \frac{1}{2\cdot 2} = \frac{1}{2} - \frac{1}{2^{n+1}}.$$

♠(Step 2.0) Suppose the adversary's query sequence during attacks is denoted by $\mathcal{Q}^*_{2,0} = \mathcal{H}^0(m_1) \to \mathcal{S}(m_1) \to \mathcal{H}^0(m_2) \to \cdots \to \mathcal{H}^0(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*)$. That is, the adversary only makes the type-0 query of m_i before its signature query for all $i \in [1, q]$.

For each signature query $\mathcal{S}(m_i)$, the simulator aborts if only if the type-0 query of m_i is set as a challenge query and the probability is $\Pr[c_{m_i} = 0] = \frac{1}{2\cdot 2^n}$. We have $n = \log(q)$. Therefore, the simulator does not abort after q signature queries with probability $(1 - \frac{1}{2^{n+1}})^q = (1 - \frac{1}{2q})^q \geq (1 - \frac{1}{2})^1 = \frac{1}{2}$. Then the simulator will solve the hard problem from $\mathcal{Q}^*_1$ with probability $\Pr[\mathcal{Q}^*_1] \approx \frac{1}{2}$. We therefore approximately obtain $\Pr[\mathcal{Q}^*_{2,0}] \geq \frac{1}{4}$.

♣(Step 2.i) We prove that $\Pr[\mathcal{Q}^*_{2,i-1}] \leq \Pr[\mathcal{Q}^*_{2,i}]$ hold for all $i \in [1, q]$ because of the non-uniform choice c_{m_i} in the programming.

We have the comparison of $\mathcal{Q}^*_{2,i-1}$ and $\mathcal{Q}^*_{2,i}$ as follows.

$$\mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1) \to \cdots \to \mathcal{S}(m_{i-1}) \begin{cases} \nearrow \mathcal{H}^{0}(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]} : \mathcal{Q}^*_{2,i-1} \\ \searrow \mathcal{H}^{k_i}(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]} : \mathcal{Q}^*_{2,i} \end{cases}$$

where $\mathcal{Q}^*_{2,[>i]} = \mathcal{H}^0\ (m_{i+1}) \rightarrow \mathcal{S}(m_{i+1}) \rightarrow \cdots \rightarrow \mathcal{H}^0(m_q) \rightarrow \mathcal{S}(m_q) \rightarrow \mathcal{H}^n(m^*)$. Since the two query sequences have the identical sub-sequence before $\mathcal{S}(m_{i-1})$, we have that the following inequality

$$\Pr[\mathcal{H}^0(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}] \leq \Pr[\mathcal{H}^{k_i}(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}]$$

implies that $\Pr[\mathcal{Q}^*_{2,i-1}] \leq \Pr[\mathcal{Q}^*_{2,i}]$.

Next we prove the correctness of the above inequality. In both query sequences $\mathcal{H}^0(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}$ and $\mathcal{H}^{k_i}(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}$, the simulator will either (1) solve hard problem from hash queries of m_i or (2) solve hard problem from $\mathcal{Q}^*_{2,[>i]}$ when the simulator neither succeeds nor aborts after $\mathcal{S}(m_i)$. Let $S_i^{(k_i)}$ be the event that the problem solution appears in $\mathcal{H}^{k_i}(m_i)$ of m_i for integer k_i, and $F_i^{(k_i)}$ be the corresponding event that the simulator fails in responding to $\mathcal{S}(m_i)$. We have

$$\Pr[\mathcal{H}^0(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}\ ] = \Pr[S_i^{(0)}]\ + \Big(1 - \Pr[S_i^{(0)}]\ - \Pr[F_i^{(0)}]\ \Big) \Pr[\mathcal{Q}^*_{2,[>i]}]$$

$$\Pr[\mathcal{H}^{k_i}(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}] = \Pr[S_i^{(k_i)}] + \Big(1 - \Pr[S_i^{(k_i)}] - \Pr[F_i^{(k_i)}]\Big) \Pr[\mathcal{Q}^*_{2,[>i]}]$$

We have the following equations and a positive value X according to the non-uniform setting $\Pr[c_m = i] = \frac{1}{2 \cdot 2^{n-i}}$.

$$\begin{aligned}
\Pr[S_i^{(k_i)}] - \Pr[S_i^{(0)}] &= \Pr[c_{m_i} < k_i] - \Pr[c_{m_i} < 0] \\
&= \Pr[c_{m_i} = 0] + \Pr[c_{m_i} = 1] + \cdots + \Pr[c_{m_i} = k_i - 1] - 0 \\
&= \frac{1}{2}\Big(\frac{1}{2^{n-k_i}} - \frac{1}{2^n}\Big) = X, \\
\Pr[F_i^{(k_i)}] - \Pr[F_i^{(0)}] &= \Pr[c_{m_i} = k_i] - \Pr[c_{m_i} = 0] \\
&= \frac{1}{2}\Big(\frac{1}{2^{n-k_i}} - \frac{1}{2^n}\Big) = X.
\end{aligned}$$

We further have $\Pr[\mathcal{Q}^*_{2,[>i]}] \leq \Pr[\mathcal{Q}^*_1] \leq \frac{1}{2}$ for any $i \in [1, q]$ because signature queries in $\mathcal{Q}^*_{2,[>i]}$ will decrease the success probability of reduction compared to no signature query in $\mathcal{Q}^*_1$. We therefore obtain

$$\begin{aligned}
&\Pr[\mathcal{H}^{k_i}(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}] - \Pr[\mathcal{H}^0(m_i) \rightarrow \mathcal{S}(m_i) \rightarrow \mathcal{Q}^*_{2,[>i]}] \\
&= \Pr[S_i^{(k_i)}] + \Big(1 - \Pr[S_i^{(k_i)}] - \Pr[F_i^{(k_i)}]\Big) \Pr[\mathcal{Q}^*_{2,[>i]}] \\
&\quad - \Big(\Pr[S_i^{(0)}]\ + \Big(1 - \Pr[S_i^{(0)}]\ - \Pr[F_i^{(0)}]\ \Big) \Pr[\mathcal{Q}^*_{2,[>i]}]\Big) \\
&= X - 2X \cdot \Pr[\mathcal{Q}^*_{2,[>i]}] \\
&\geq X - 2X \cdot \frac{1}{2} \\
&= 0.
\end{aligned}$$

♣(Step *Final*) We have $\Pr[\mathcal{Q}^*_{2,q}] \leq \Pr[\mathcal{Q}^*_{\mathcal{A}}]$. In comparison with $\mathcal{Q}^*_{2,q}$, the query sequence $\mathcal{Q}^*_{\mathcal{A}}$ allows the adversary to (1) generate and make hash queries of any

message $m \notin \{m_1, m_2, \cdots, m_q, m^*\}$ without signature query on m, and to (2) make hash queries without following the sequence $\mathcal{Q}^*_{2,q}$, where the adversary could make hash queries of m_i before $\mathcal{S}(m_{i-1})$.

All hash queries will be responded by the simulator without abort. Making hash queries of additional messages without signature queries on them will not increase the failure probability of simulation. In our simulation, a hash query associated with m_i is responded according to the parameter c_{m_i}, which is chosen independently for each message. Whether or not the simulator fails in $\mathcal{S}(m_i)$ depends on (k_i, c_{m_i}) and is not related to when the adversary made hash queries of m_i. We therefore have $\Pr[\mathcal{Q}^*_{2,q}] \leq \Pr[\mathcal{Q}^*_{\mathcal{A}}]$.

This completes the high-level intuition of our reduction with success probability $\frac{1}{4}$ for the chain-based BLS scheme in the EUF-CMA security model under the CDH assumption, no matter what (the polynomial number) q is as long as we have $n = \log q$.

1.3 Impossibility of Reductions

Many excellent research results in the literature have focused on disproving the equivalence between constructed schemes and underlying hardness assumptions.

The impossibility of efficient reduction includes the result [12] that inverting low-exponent RSA may not be equivalent to factoring, the result [47] that breaking ElGamal like discrete-log-based signatures may not be equivalent to discrete log, and the result [41] that breaking some HIBE or ABE system cannot be efficiently reduced to breaking a non-interactive hardness assumption.

The impossibility of reduction better than optimal tightness was first studied by Coron in [16] by introducing the meta-reduction technique. So far, the analysis of optimal tightness has been studied for many primitives including any "simple" reduction for unique signatures or re-randomizable signatures in [16, 34, 5, 44], for specific schemes (like Schnorr-type signatures) in [47, 23, 50, 20, 21], for encryption in [5, 29], for signatures from identification in [40], for non-interactive key-exchange in [5, 30, 15], for MACs and PRFs in [45], and the recent result for verifiable random functions in [46].

2 Definitions

Definition 1 (Digital Signatures). *A digital signature scheme consists of the following three algorithms and fulfills correctness.*

$\mathsf{KeyGen}(1^\kappa)$. *The key generation algorithm takes as input a security parameter κ and returns a key pair denoted by (pk, sk).*

$\mathsf{Sign}(pk, sk, m)$. *The signing algorithm takes as input (pk, sk) and a message m to be signed. It returns a signature on m denoted by Σ_m.*

$\mathsf{Verify}(pk, \Sigma_m, m)$. *The verification algorithm takes as input pk and a signed message (m, Σ_m). It returns true or false.*

The correctness requires that for any key pair (pk, sk), any message m from message space, and its signature Σ_m, we have $\Pr[\mathsf{Verify}(pk, \Sigma_m, m) = \mathsf{true}] = 1$.

Definition 2 (Unique Signatures [5]). *Let* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a signature scheme and* $\Sigma(pk, m)$ *be the set of valid signatures on* m *under* pk*, defined as* $\Sigma(pk, m) = \{\Sigma_m : \mathsf{Verify}(pk, \Sigma_m, m) = \mathsf{true}\}$*. We say that* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is a unique signature scheme if* $|\Sigma(pk, m)| = 1$ *for all* pk *and* m*.*

We stress that deterministic signatures (such as [38]) and unique signatures are different. In deterministic signatures, the signature on m generated by the signer is unique. In unique signatures, the signature on m that can pass the verification is unique, which implies that the generated signature must be also unique. That is, a deterministic signature scheme may not be a unique signature scheme, while a unique signature scheme must be a deterministic scheme.

Definition 3 (EUF-CMA Security Model). *The existentially unforgeable against chosen-message attacks (EUF-CMA) security model is defined as follows.*

- **Setup:** *The challenger takes as input security parameter* κ *and generates a key pair* (pk, sk)*. The public key is given to the adversary.*
- **Query:** *The adversary adaptively chooses any message* m *for its signature query. The challenger runs the signing algorithm and sends the output signature* Σ_m *to the adversary.*
- **Forgery:** *The adversary outputs a forged signature* Σ_{m^*} *on message* m^* *and wins the game if* Σ_{m^*} *is valid and no signature query was made on* m^**.*

A digital signature scheme is (t, q, ϵ)*-secure in the EUF-CMA security model if no probabilistic polynomial time adversary can win the game with probability* ϵ *in polynomial time* t *after making at most* q *signature queries, where* ϵ *is a negligible function in* κ*.*

3 Optimal Analysis for the Chain-based BLS Scheme

We first give a general definition of non-interactive computationally hard assumption that was originally given in [2, 5].

Definition 4. *A non-interactive computationally hard assumption, denoted by* (T, V)*, consists of two probabilistic polynomial time algorithms.*

- *Taking as input a security parameter* κ*, the instance generation algorithm* T *outputs a problem instance* ins *and a witness* wit*.*
- *Taking as input* $(\mathsf{ins}, \mathsf{wit})$ *and a candidate solution* sol*, the verification algorithm* V *returns* true *or* false*. If* $\mathsf{V}(\mathsf{ins}, \mathsf{wit}, \mathsf{sol}) = \mathsf{true}$*, then we say that* sol *is a valid solution to the problem instance* ins*.*

We say that the assumption (T, V) *is* $(t(\kappa), \epsilon(\kappa))$ *computationally hard if every probabilistic polynomial time algorithm* $\mathcal{B}$ *that stops in* $t(\kappa)$ *polynomial time can only return* sol *to a given instance* ins *with negligible success probability* $\epsilon(\kappa)$*, where the probability is taken over the random coins consumed by* T *and* $\mathcal{B}$*.*

The original definition has been simplified because we focus on computationally hard assumptions instead of general assumptions that include the decisionally hard assumptions.

Our optimal analysis is given below for the chain-based BLS scheme. In comparison with other meta-reductions [16, 34, 5] in simulating the adversary via rewinding, our meta-reduction proof requires to rewind at most n times in order to simulate the adversary successfully in attacking the chain-based BLS scheme.

Theorem 1. *Let* (T, V) *be a non-interactive computationally hard assumption. Let* $\mathcal{A}$ *be an adversary who can* $(t_{\mathcal{A}}, q, \epsilon_{\mathcal{A}})$*-break the chain-based BLS scheme in the EUF-CMA model. Suppose there exists a reduction* $\mathcal{R}$ *that can* $(t_{\mathcal{A}}, q, \epsilon_{\mathcal{A}}, t_{\mathcal{R}}, \epsilon_{\mathcal{R}})$*-reduce from breaking* (T, V) *assumption to breaking the chain-based BLS scheme by* $\mathcal{A}$*. We can construct an algorithm* $\mathcal{B}$ *that* $(t_{\mathcal{B}}, \epsilon_{\mathcal{B}})$*-breaks* (T, V) *with*

$$t_{\mathcal{B}} \leq O(n \cdot t_{\mathcal{R}}),\ \epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{R}} - \frac{\epsilon_{\mathcal{A}}}{q^{\frac{1}{n}}}.$$

Proof. We first describe a potentially hypothetical and inefficient adversary $\mathcal{A}$. Then this adversary will be simulated by us (namely we construct a simulated adversary) in order to run $\mathcal{R}$ to break the hardness assumption.

The Hypothetical Adversary. The hypothetical adversary attacks the chain-based BLS scheme in the corresponding EUF-CMA security model as follows.

Setup: Given an instance ins of (T, V), $\mathcal{R}$ generates a public key pk that is given to the hypothetical adversary.

Query: The adversary flips a biased coin with $\Pr[Coin = 1] = \epsilon_{\mathcal{A}}$ and $\Pr[Coin = 0] = 1 - \epsilon_{\mathcal{A}}$. If $Coin = 0$, abort the attack. Otherwise, the adversary picks q random messages denoted by $\mathcal{M}_0 = \{m_1^*, m_2^*, \cdots, m_q^*\}$, where $q-1$ of them will be randomly picked for signature queries (satisfies the definition of at most q in the EUF-CMA model) and the last one is used for signature forgery. For simplicity, we assume that the adversary runs an inefficient algorithm that computes the secret key sk from the received public key pk and uses it to generate all involved hash queries.

The adversary computes and makes queries as follows.

- Make type-0 queries of all $m \in \mathcal{M}_0$, denoted by $x = m|\Sigma_m^0 = m|(\ )$.
- Upon receiving all responses to type-0 queries of all messages in $\mathcal{M}_0$, randomly pick $q - q^{1-\frac{1}{n}}$ numbers of messages from $\mathcal{M}_0$ for their signature queries. If $\mathcal{R}$ aborts, the adversary stops. Otherwise, let the remaining messages whose signatures are not queried be in $\mathcal{M}_1$. We have

$$|\mathcal{M}_1| = q - (q - q^{1-\frac{1}{n}}) = q^{1-\frac{1}{n}}.$$

- For $i = 1, 2, 3, \cdots, n-1$, the adversary makes queries as follows.

- Make type-i queries of all $m \in \mathcal{M}_i$, denoted by $x = m|\Sigma_m^i$.
- Upon receiving all responses to type-i queries of all messages, randomly pick $q^{1-\frac{i}{n}} - q^{1-\frac{i+1}{n}}$ numbers of messages from $\mathcal{M}_i$ for their signature queries. If $\mathcal{R}$ aborts, the adversary stops. Otherwise, let the remaining messages whose signatures are not queried be in $\mathcal{M}_{i+1}$.

$$|\mathcal{M}_{i+1}| = q^{1-\frac{i}{n}} - (q^{1-\frac{i}{n}} - q^{1-\frac{i+1}{n}}) = q^{1-\frac{i+1}{n}}.$$

– Make the type-n query of all $m \in \mathcal{M}_n$, denoted by $x = m|\Sigma_m^n$.

Forgery: We have $|\mathcal{M}_n| = q^0 = 1$. Let the type-$n$ query of message m^* be denoted by $m^*|\Sigma_{m^*}^n$. We have $\Sigma_{m^*}^n = \Sigma_{m^*}$ which is the signature on m^*. Notice that there is no signature query on m^*. The adversary outputs Σ_{m^*} as the forged signature on m^* and $\mathcal{R}$ outputs sol as the solution to ins to break the hardness assumption (T, V).

In summary, the hypothetical adversary makes hash queries and signature queries in the following sequence as long as $\mathcal{R}$ does not abort.

$$\mathcal{T}_0(\mathcal{M}_0) \rightarrow \mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) \rightarrow \mathcal{T}_1(\mathcal{M}_1) \rightarrow \mathcal{S}(\mathcal{M}_1 \setminus \mathcal{M}_2) \rightarrow \cdots \rightarrow \mathcal{S}(\mathcal{M}_{n-1} \setminus \mathcal{M}_n) \rightarrow \mathcal{T}_n(\mathcal{M}_n).$$

– $\mathcal{T}_i(\mathcal{M}_i)$ is the set of type-i queries of all messages in $\mathcal{M}_i$ defined as

$$\Big\{m|\Sigma_m^i : m \in \mathcal{M}_i, \Sigma_m^i = \Big(H(m|\Sigma_m^0)^\alpha, H(m|\Sigma_m^1)^\alpha, \cdots, H(m|\Sigma_m^{i-1})^\alpha\Big)\Big\}.$$

– $\mathcal{S}(\mathcal{M}_i \setminus \mathcal{M}_{i+1})$ is the set of signature queries on all messages in $\mathcal{M}_i \setminus \mathcal{M}_{i+1}$.
– We have $\mathcal{M}_i \subset \mathcal{M}_{i-1}$ and $|\mathcal{M}_i| = q^{1-\frac{i}{n}}$ for all $i \in [1, n]$.

When interacting with such a hypothetical adversary, $\mathcal{R}$ would break the hardness assumption with probability $\epsilon_{\mathcal{R}}$ according to the definition.

The Simulated Adversary. Given as input an instance ins of (T, V), $\mathcal{R}$ generates a public key pk and gives it to the simulated adversary. The simulated adversary also tosses a biased coin the same as the hypothetical adversary to continue or abort. When $Coin = 1$, the simulated adversary aims to simulate the hypothetical adversary in making queries as follows unless $\mathcal{R}$ aborts.

$$\mathcal{T}_0(\mathcal{M}_0) \rightarrow \mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) \rightarrow \mathcal{T}_1(\mathcal{M}_1) \rightarrow \mathcal{S}(\mathcal{M}_1 \setminus \mathcal{M}_2) \rightarrow \cdots \rightarrow \mathcal{S}(\mathcal{M}_{n-1} \setminus \mathcal{M}_n) \rightarrow \mathcal{T}_n(\mathcal{M}_n).$$

It is easy to simulate the adversary in computing hash queries in $\mathcal{T}_0(\mathcal{M}_0)$ because all hash queries are plain messages, namely $m|\Sigma_m^0 = m|()$, without any block signature (BLS signature). The main difficulty of simulating the adversary is to generate and make all hash queries in $\mathcal{T}_i(\mathcal{M}_i)$ for all $i \in [1, n]$. This is because all these hash queries contain block signatures Σ_m^i that cannot be efficiently computed without knowing the secret key α.

We are going to simulate this hypothetical adversary with a rewinding argument. We will be able to successfully simulate the hypothetical adversary after rewinding $\mathcal{R}$ with the help of signature computed by $\mathcal{R}$ before the rewind. More precisely, for all $i \in [1, n]$, we make signature queries $\mathcal{S}(\mathcal{M}_i)$ to $\mathcal{R}$ and then rewind $\mathcal{R}$ once to simulate hash queries $\mathcal{T}_i(\mathcal{M}_i)$. The details are as follows.

1. Let the state after the hash queries $\mathcal{T}_0(\mathcal{M}_0)$ be st_0.
2. At the state st_0, we make signature queries on $\mathcal{M}_1$ to $\mathcal{R}$ as follows.

$$\mathcal{T}_0(\mathcal{M}_0) \overset{st_0}{\to} \mathcal{S}(\mathcal{M}_1).$$

   If $\mathcal{R}$ does not abort, we will receive signatures $\Sigma_{\mathcal{M}_1}$ on messages in $\mathcal{M}_1$.
3. We rewind $\mathcal{R}$ to the state st_0. This time, we make signature queries on $\mathcal{M}_0 \backslash \mathcal{M}_1$ to $\mathcal{R}$. That is, $\mathcal{T}_0(\mathcal{M}_0) \overset{st_0}{\to} \mathcal{S}(\mathcal{M}_0 \backslash \mathcal{M}_1)$. If $\mathcal{R}$ aborts, we stop the interaction with $\mathcal{R}$ the same as the hypothetical adversary. Otherwise, $\mathcal{R}$ does not abort and we continue type-1 queries

$$\mathcal{T}_0(\mathcal{M}_0) \overset{st_0}{\to} \mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) \to \mathcal{T}_1(\mathcal{M}_1),$$

   where the hash queries $\mathcal{T}_1(\mathcal{M}_1)$ will be simulated with signatures $\Sigma_{\mathcal{M}_1}$ on $\mathcal{M}_1$ received from $\mathcal{R}$ in the step 2.
4. Let the state after the hash queries $\mathcal{T}_1(\mathcal{M}_1)$ be st_1.

When we are at the state st_1 after the hash queries $\mathcal{T}_1(\mathcal{M}_1)$, we can continue the simulation on $\mathcal{T}_2(\mathcal{M}_2)$ in an analogous way for $\mathcal{T}_1(\mathcal{M}_1)$. In general, when we are at the state st_i after the hash queries $\mathcal{T}_i(\mathcal{M}_i)$ for any $i \in [0, n-1]$ and seeing all responses, we continue the simulation on the adversary as follows.

1. Let the state after the hash queries $\mathcal{T}_i(\mathcal{M}_i)$ be st_i.
2. At the state st_i, we make signature queries on $\mathcal{M}_{i+1}$ to $\mathcal{R}$ as follows.

$$\mathcal{T}_0(\mathcal{M}_0) \to \cdots \to \mathcal{T}_i(\mathcal{M}_i) \overset{st_i}{\to} \mathcal{S}(\mathcal{M}_{i+1}).$$

   If $\mathcal{R}$ does not abort, we will receive signatures $\Sigma_{\mathcal{M}_{i+1}}$ on $\mathcal{M}_{i+1}$.
3. We rewind $\mathcal{R}$ to the state st_i. This time, we make signature queries on $\mathcal{M}_i \backslash \mathcal{M}_{i+1}$ to $\mathcal{R}$. That is, $\mathcal{T}_0(\mathcal{M}_0) \to \cdots \to \mathcal{T}_i(\mathcal{M}_i) \overset{st_i}{\to} \mathcal{S}(\mathcal{M}_i \backslash \mathcal{M}_{i+1})$. If $\mathcal{R}$ aborts, we stop the interaction with $\mathcal{R}$. Otherwise, $\mathcal{R}$ does not abort and we continue type-$(i+1)$ queries

$$\mathcal{T}_0(\mathcal{M}_0) \to \cdots \to \mathcal{T}_i(\mathcal{M}_i) \overset{st_i}{\to} \mathcal{S}(\mathcal{M}_i \setminus \mathcal{M}_{i+1}) \to \mathcal{T}_{i+1}(\mathcal{M}_{i+1}),$$

where the hash queries $\mathcal{T}_{i+1}(\mathcal{M}_{i+1})$ will be simulated with signatures $\Sigma_{\mathcal{M}_{i+1}}$ on $\mathcal{M}_{i+1}$ received from $\mathcal{R}$ in the step 2.

4. Let the state after the hash queries $\mathcal{T}_{i+1}(\mathcal{M}_{i+1})$ be st_{i+1}.

If $\mathcal{R}$ does not abort in responding to signature queries, after the n-th time rewind, we have successfully simulated the adversary who generated and made queries $\mathcal{T}_0(\mathcal{M}_0) \to \mathcal{S}(\mathcal{M}_0 \setminus \mathcal{M}_1) \to \mathcal{T}_1(\mathcal{M}_1) \to \cdots \to \mathcal{S}(\mathcal{M}_{n-1} \setminus \mathcal{M}_n) \to \mathcal{T}_n(\mathcal{M}_n)$. This completes the description of how to simulate the hypothetical adversary.

Next we analyze the correctness of simulating $\mathcal{T}_{i+1}(\mathcal{M}_{i+1})$ in step 3 with $\Sigma_{\mathcal{M}_{i+1}}$ from step 2. At the state st_i, when $\mathcal{R}$ does not abort before the rewind (step 2), we have

$$\Sigma_{\mathcal{M}_{i+1}} = \left\{ \left(H(m|\Sigma_m^0)^\alpha, H(m|\Sigma_m^1)^\alpha, \cdots, H(m|\Sigma_m^{n-1})^\alpha \right) : m \in \mathcal{M}_{i+1} \right\}.$$

When $\mathcal{R}$ does not abort after the rewind (step 3), we need Σ_m^{i+1} to simulate all type-$(i+1)$ queries of $m \in \mathcal{M}_{i+1}$:

$$\mathcal{T}_{i+1}(\mathcal{M}_{i+1}) = \left\{ m \middle| \left(H(m|\Sigma_m^0)^\alpha, H(m|\Sigma_m^1)^\alpha, \cdots, H(m|\Sigma_m^i)^\alpha \right) : m \in \mathcal{M}_{i+1} \right\}.$$

At the state st_i it was after $\mathcal{T}_i(\mathcal{M}_i)$, $\mathcal{R}$ should have responded to all type-i queries of messages in $\mathcal{M}_i$. That is, $\alpha, H(m|\Sigma_m^0), \cdots, H(m|\Sigma_m^i)$ for all $m \in \mathcal{M}_{i+1} \subseteq \mathcal{M}_i$ must be identical before the rewind and after the rewind at the state st_i. Then, the unique block signatures $(H(m|\Sigma_m^0)^\alpha, H(m|\Sigma_m^1)^\alpha, \cdots, H(m|\Sigma_m^i)^\alpha)$ in $\Sigma_{\mathcal{M}_{i+1}}$ and in $\mathcal{T}_{i+1}(\mathcal{M}_{i+1})$ must be identical and therefore we can correctly use $\Sigma_{\mathcal{M}_{i+1}}$ to simulate $\mathcal{T}_{i+1}(\mathcal{M}_{i+1})$.

The Error Probability. We fail in simulating the hypothetical adversary if there exists $i^\# \in [1, n]$ such that we need to continue type-$i^\#$ queries $\mathcal{T}_{i^\#}(\mathcal{M}_{i^\#})$ in step 3 but we did not receive signatures $\Sigma_{\mathcal{M}_{i^\#}}$ in step 2 due to the failure of $\mathcal{R}$.

We define $Stop_i, Bad_i$ for all $i \in [1, n]$ to be events as follows.

$$Bad_i : \mathcal{S}(\mathcal{M}_i) = 0 \wedge \mathcal{S}(\mathcal{M}_{i-1} \setminus \mathcal{M}_i) = 1$$
$$Stop_i : \mathcal{S}(\mathcal{M}_{i-1}) = 0$$

- $\mathcal{S}(\mathcal{M}) = 0$ means that $\mathcal{R}$ cannot respond to signature queries $\mathcal{S}(\mathcal{M})$, while $\mathcal{S}(\mathcal{M}) = 1$ means that $\mathcal{R}$ can respond to signature queries $\mathcal{S}(\mathcal{M})$.
- Bad_i refers to the event that we fail in simulating the adversary after the state st_{i-1} and before the state st_i. More precisely, this event occurs when $\mathcal{R}$ cannot respond to signature queries $\mathcal{S}(\mathcal{M}_i)$ in step 2 denoted by $\mathcal{S}(\mathcal{M}_i) = 0$, but $\mathcal{R}$ can respond to signature queries $\mathcal{S}(\mathcal{M}_{i-1} \setminus \mathcal{M}_i)$ in step 3 denoted by $\mathcal{S}(\mathcal{M}_{i-1} \setminus \mathcal{M}_i) = 1$.
- $Stop_i$ refers to the event that the simulation stops after the state st_{i-1} and before the state st_i. It stops either because $\mathcal{R}$ fails or we fail in simulating the adversary.

Let A_i be the event that the simulation first stops due to the event $Stop_i$.

$$A_i = \overline{Stop_1} \wedge \overline{Stop_2} \wedge \cdots \wedge \overline{Stop_{i-1}} \wedge Stop_i, \text{ where } A_1 = Stop_1.$$

Then we fail in simulating the hypothetical adversary with probability $\Pr[Bad]$:

$$\Pr[Bad] = \sum_{i=1}^{n} \Big(\Pr[Bad_i | A_i] \cdot \Pr[A_i] \Big)$$

We deduct $\Pr[Bad] \leq 1/q^{\frac{1}{n}}$ according to the following two results.

- $\Pr[Bad_i|A_i] \leq \frac{1}{q^{\frac{1}{n}}}$ for all $i \in [1, n]$, which is proved in Lemma 1.
- $\sum_{i=1}^{n} \Pr[A_i] \leq 1$, which is proved as follows. We have the following equation

  $\Pr[\overline{Stop_1} \wedge \overline{Stop_2} \wedge \cdots \wedge \overline{Stop_{i-1}} \wedge \overline{Stop_i}] + \Pr[A_i] = \Pr[\overline{Stop_1} \wedge \overline{Stop_2} \wedge \cdots \wedge \overline{Stop_{i-1}}]$

  for all $i \in [1, n]$ by applying the rule $\Pr[B \wedge \overline{C}] + \Pr[B \wedge C] = \Pr[B]$ for any events B and C. With this equation, we have

  $$\Pr[\overline{Stop_1} \wedge \overline{Stop_2} \wedge \cdots \wedge \overline{Stop_n}] + \sum_{i=1}^{n} \Pr[A_i] = \Pr[\overline{Stop_1}] + \Pr[A_1] = 1,$$

  which implies $\sum_{i=1}^{n} \Pr[A_i] \leq 1$.

Finally, we fail in simulating the adversary when the events $Coin = 1$ and Bad both occur with probability $\epsilon_{\mathcal{A}} \cdot \Pr[Bad]$. Otherwise, the bad event does not occur. When interacting with this simulated adversary, according to the meta-reduction framework by Coron in [16], $\mathcal{R}$ would break the hardness assumption for us with probability $\epsilon_{\mathcal{R}} - \frac{\epsilon_{\mathcal{A}}}{q^{\frac{1}{n}}}$.

This completes the proof of the theorem. □

Lemma 1. $\Pr[Bad_i|A_i] \leq \frac{1}{q^{\frac{1}{n}}}$ *for all* $i \in [1, n]$.

Proof. The event $A_i = \overline{Stop_1} \wedge \overline{Stop_2} \wedge \cdots \wedge \overline{Stop_{i-1}} \wedge Stop_i$ indicates $\mathcal{S}(\mathcal{M}_{i-1}) = 0$ at the state st_{i-1}. Let $\mathcal{M}^s_{i-1}$ be the largest subset of $\mathcal{M}_{i-1}$ such that $\mathcal{S}(\mathcal{M}^s_{i-1}) = 1$, namely $\mathcal{R}$ can simulate all signatures on messages in $\mathcal{M}^s_{i-1}$. We have

$$|\mathcal{M}^s_{i-1}| \leq |\mathcal{M}_{i-1}| - 1 = q^{1-\frac{i-1}{n}} - 1.$$

By putting all above analysis together, we obtain

$$\begin{aligned} \Pr[Bad_i|A_i] &= \Pr[\mathcal{S}(\mathcal{M}_i) = 0 \wedge \mathcal{S}(\mathcal{M}_{i-1} \setminus \mathcal{M}_i) = 1 |\ \mathcal{S}(\mathcal{M}_{i-1}) = 0] \\ &\leq \Pr[\mathcal{S}(\mathcal{M}_{i-1} \setminus \mathcal{M}_i) = 1 |\ \mathcal{S}(\mathcal{M}_{i-1}) = 0] \\ &= \Pr[(\mathcal{M}_{i-1} \setminus \mathcal{M}_i) \subseteq \mathcal{M}^s_{i-1}] \end{aligned}$$

The probability $\Pr[(\mathcal{M}_{i-1} \setminus \mathcal{M}_i) \subseteq \mathcal{M}^s_{i-1}]$ is equivalent to that $|\mathcal{M}_{i-1} \setminus \mathcal{M}_i|$ distinct messages randomly picked from the set $\mathcal{M}_{i-1}$ lie in $\mathcal{M}^s_{i-1}$. By picking

messages one by one, the first one lies in $\mathcal{M}^s_{i-1}$ with probability $\frac{|\mathcal{M}^s_{i-1}|}{|\mathcal{M}_{i-1}|}$ and the second one lies in $\mathcal{M}^s_{i-1}$ with probability $\frac{|\mathcal{M}^s_{i-1}|-1}{|\mathcal{M}_{i-1}|-1}$ and so on.

Based on the above analysis, we have

$$\begin{aligned}
&\Pr[Bad_i|A_i]\\
&\leq \Pr[(\mathcal{M}_{i-1} \setminus \mathcal{M}_i) \subseteq \mathcal{M}^s_{i-1}]\\
&= \frac{|\mathcal{M}^s_{i-1}|}{|\mathcal{M}_{i-1}|} \cdot \frac{|\mathcal{M}^s_{i-1}|-1}{|\mathcal{M}_{i-1}|-1} \cdot \frac{|\mathcal{M}^s_{i-1}|-2}{|\mathcal{M}_{i-1}|-2} \cdots \frac{|\mathcal{M}^s_{i-1}|-|\mathcal{M}_{i-1} \setminus \mathcal{M}_i|+1}{|\mathcal{M}_{i-1}|-|\mathcal{M}_{i-1} \setminus \mathcal{M}_i|+1}\\
&\leq \frac{|\mathcal{M}_{i-1}|-1}{|\mathcal{M}_{i-1}|} \cdot \frac{|\mathcal{M}_{i-1}|-2}{|\mathcal{M}_{i-1}|-1} \cdot \frac{|\mathcal{M}_{i-1}|-3}{|\mathcal{M}_{i-1}|-2} \cdots \frac{|\mathcal{M}_{i-1}|-|\mathcal{M}_{i-1} \setminus \mathcal{M}_i|}{|\mathcal{M}_{i-1}|-|\mathcal{M}_{i-1} \setminus \mathcal{M}_i|+1}\\
&= \frac{|\mathcal{M}_{i-1}|-|\mathcal{M}_{i-1} \setminus \mathcal{M}_i|}{|\mathcal{M}_{i-1}|}\\
&= \frac{q^{1-\frac{i-1}{n}} - \left(q^{1-\frac{i-1}{n}} - q^{1-\frac{i}{n}}\right)}{q^{1-\frac{i-1}{n}}}\\
&= \frac{1}{q^{\frac{1}{n}}}.
\end{aligned}$$

This completes the proof of the lemma. □

4 Optimal Tightness for the Chain-based BLS Scheme

In this section, we formally show how to prove the security of the chain-based BLS scheme with optimal tightness in the EUF-CMA security model under the CDH hardness assumption. In comparison with the high-level intuition proof given in the introduction,

- The formal proof here defines and classifies hash queries including useless or dummy hash queries generated and made by the adversary.
- The formal proof here considers the general parameter setting for any scheme parameter n instead of $n = \log q$. More precisely, the proof will show that given a chain-based BLS scheme instantiated with any integer n, the reduction loss is at most $4 \cdot q^{\frac{1}{n}}$ for q signature queries.

Our security reduction is split into two theorems. In the first theorem, we provide the framework of the reduction without calculating the success probability. We analyze the success probability in the second theorem.

4.1 Framework of Security Reduction

Theorem 2. *Let H be the hash function viewed as the random oracle. Suppose there exists an adversary $\mathcal{A}$ who can (t, q, ϵ)-break the chain-based BLS scheme in the EUF-CMA security model. We can construct a simulator $\mathcal{B}$ that (t', ϵ')-solves the CDH problem, where*

$$t' = t + O(q_H + q \cdot n), \quad \epsilon' = \Pr[Suc] \cdot \epsilon.$$

Here q_H is the number of hash queries to the random oracle and $\Pr[Suc]$ *is the success probability of solving the CDH problem when the adversary successfully forges a valid signature.*

Proof. Suppose there exists an adversary who can (t, q, ϵ)-break the chain-based BLS scheme in the EUF-CMA model. A simulator can be constructed to solve the CDH problem defined over a bilinear pairing $(\mathbb{G}, \mathbb{G}_T, p, e, g)$. Given as input a random instance $(g, g^{\hat{a}}, g^{\hat{b}})$, the simulator aims to compute $g^{\hat{a}\hat{b}}$ and constructs the simulated scheme for the adversary as follows.

Setup: The simulator sets $h = g^{\alpha} = g^{\hat{a}}$ and gives $pk = (\mathbb{G}, \mathbb{G}_T, p, e, g, h, n)$ to the adversary. The hash function H is set as a random oracle controlled by the simulator.

Hash Query: Our reduction uses the adversary's hash queries to the random oracle H to solve the CDH problem. We clarify all hash queries before introducing how to program their responses.

- A hash query x is called type-j query of m if $x = m|\Sigma_m^j$, where $\Sigma_m^j = (\sigma_1, \sigma_2, \cdots, \sigma_j)$ is the first j block (basic) BLS signatures on m and $\sigma_j = H(m|\Sigma_m^{j-1})^{\alpha}$. In particular, the type-0 query of m is $m|(\ )$ composed of m and $\Sigma_m^0 = (\ )$. We define $\mathcal{H}^i(m)$ to be the set of queries of m from type-0 to type-i as $\mathcal{H}^i(m) = \{m|\Sigma_m^0, m|\Sigma_m^1, m|\Sigma_m^2, \cdots, m|\Sigma_m^i\}$. Except hash queries in $\mathcal{H}^n(m)$, other queries of m in our reduction are defined as useless queries because the signature on m is not related to useless queries.
- The simulator uses a hash list to record all hash queries from the adversary $\mathcal{A}$ and their responses. Suppose x is a hash query and y is the response, namely $y = H(x)$. If x is a useless query, the simulator will add the tuple (x, y) into the hash list. Otherwise, $x = m|\Sigma_m^j \in \mathcal{H}^n(m)$ is a type-j query of m, and the simulator will add the tuple $(m, j, \Sigma_m^j, c_m, x, y, z_m^j)$ into the hash list where c_m is a secret related to message m and z_m^j is a secret related to the query x (to be explained later).
- The simulator can easily verify whether a candidate query x lies in $\mathcal{H}^n(m)$. Given a query x, the simulator will judge it as a type-j query of m if x can be parsed as $x = m|(\sigma_1, \cdots, \sigma_j)$ and

$$e\Big(\sigma_{i+1}, g\Big) = e\Big(H(m|\Sigma_m^i), h\Big), \text{ for all } i \in [0, j-1].$$

  In the above computation, if the type-i query of m for any $i \in [0, j-1]$ was never being queried to the random oracle by the adversary before x, the simulator will firstly respond to the hash query $m|\Sigma_m^i$ before verifying the query x. It guarantees queries in a sequence that if x is the valid type-i query of m, the type-1, type-2, $\cdots$, type-$(i-1)$ queries of m have already been added and responded by the simulator before the type-i query.
- An integer $c_m \in [0, n-1]$ for each message m is chosen independently but non-uniformly satisfying $\Pr[c_m = j] = a_j$. Here $a_0, a_1, \cdots, a_{n-1} > 0$ are values satisfying $0 < a_0 + a_1 + a_2 + \cdots + a_{n-1} \leq \frac{1}{2}$ defined in Theorem 3.

Upon receiving a hash query x from the adversary, the simulator programs the response $y = H(x)$ as follows.

- **Step 1:** If x has already been queried, the simulator responds to this query following the tuple (x, y) or $(m, j, \Sigma_m^j, c_m, x, y, z_m^j)$ in the hash list. Otherwise, the simulator uses the symbols "|, ()" to parse the query $x = *|(*)$ for arbitrary strings denoted by $*$. It might be a candidate type-i query or a useless query with structures different from the description in the scheme.
- **Step 2:** Suppose $x \in \mathcal{H}^n(m)$. If message m inside the query $x = m|\Sigma_m^j$ is a new message to the random oracle, the simulator should first choose $c_m \in [0, n-1]$ as stated previously. Otherwise, c_m can be obtained from other tuples about message m in the hash list. The simulator randomly chooses $z_m^j \in \mathbb{Z}_p$ and sets

$$y = H(x) = H(m|\Sigma_m^j) = \begin{cases} g^{z_m^j} & : \quad j \neq c_m \\ g^{\hat{b}+z_m^j} & : \quad j = c_m \end{cases}.$$

  The simulator adds $(m, j, \Sigma_m^j, c_m, x, y, z_m^j)$ into the list.
- **Step 3:** Suppose $x \notin \mathcal{H}^n(m)$. The simulator chooses a random $y \in \mathbb{G}$ and adds (x, y) into the hash list.

This completes the description of hash queries and their responses. We classify all hash queries in $\mathcal{H}^n(m)$ into the following three kinds.

- We name the type-c_m query of m as a **challenge query**.
- We name the type-$(c_m + 1)$ query of m as a **solution query**.
- Other queries in $\mathcal{H}^n(m)$ are named as **normal queries**.

That is, if x is a type-c_m query of m, it will be set as a challenge query by the simulator and responded with

$$H(x) = H(m|\Sigma_m^{c_m}) = g^{\hat{b}+z_m^{c_m}}.$$

Then the type-$(c_m + 1)$ query of m, denoted by $m|\Sigma_m^{c_m+1}$, carries the block signatures $\Sigma_m^{c_m+1} = (\sigma_1, \sigma_2, \cdots, \sigma_{c_m+1})$ and σ_{c_m+1} is equal to

$$\sigma_{c_m+1} = H(m|\Sigma_m^{c_m})^{\alpha} = \left(g^{\hat{b}+z_m^{c_m}}\right)^{\hat{a}} = g^{\hat{a}\hat{b}+\hat{a}z_m^{c_m}}.$$

The CDH problem solution can be extracted from this block signature with the known $z_m^{c_m}$ in the hash list by

$$\frac{\sigma_{c_m+1}}{(g^{\hat{a}})^{z_m^{c_m}}} = \frac{g^{\hat{a}\hat{b}+\hat{a}z_m^{c_m}}}{(g^{\hat{a}})^{z_m^{c_m}}} = g^{\hat{a}\hat{b}}.$$

Therefore, a solution query is a hash query that carries the CDH problem solution. The simulator is able to solve the CDH problem if the adversary generates

and makes a solution query to the random oracle. We note that once the simulator extracts the problem solution $g^{\hat{a}\hat{b}}$, it can immediately stops the simulation or can continue the simulation successfully without abort using $g^{\hat{a}\hat{b}}$ until the adversary returns a forged signature.

Signature Query: The adversary adaptively chooses a message m_i for its signature query. Before the signature query on m_i, we assume that the adversary has generated and made the following hash queries to the random oracle

$$\mathcal{H}^{k_i}(m_i) = \{m_i|\Sigma^0_{m_i}, m_i|\Sigma^1_{m_i}, \cdots, m_i|\Sigma^{k_i}_{m_i}\},$$

where $k_i \in [0, n-1]$ is an integer adaptively decided by the adversary and the range is explained as follows. The type-0 query $m_i|\Sigma^0_{m_i} = m_i|(\ )$ is the plain message. Therefore we simply assume that the adversary makes at least the type-0 query of m_i ($k_i \geq 0$). If the adversary also generates and makes the type-n query of m_i, namely $m_i|\Sigma^n_{m_i}$, the hash query already implies a valid signature on m_i and then there is no need to make its signature query. We therefore also assume that $k_i \leq n-1$.

Recalling that the simulator chose an integer $c_i \in [0, n-1]$ for the message m_i (the symbol c_{m_i} is simplified into c_i for message m_i) in the response to hash queries. The signature simulation falls into the following three cases.

- **Case 1:** $c_i < k_i$. That is, $c_i + 1 \leq k_i$. In this case, the type-c_i query is set as a challenge query and the adversary has already generated and made the type-(c_i+1) query (the solution query) to the random oracle. The simulator is able to extract the problem solution from this solution query.
- **Case 2:** $c_i = k_i$. The simulator aborts and fails in the signature simulation.
- **Case 3:** $k_i < c_i$. In this case, the adversary has not yet generated and made the type-c_i query of m_i. Then no hash query of m_i is set as the challenge query. Upon receiving the signature query on m_i, the simulator updates c_i with $c_i = \infty$ in all tuples related to message m_i in the hash list. According to the setting of the oracle response, the hash queries

  $$x = m_i|\Sigma^j_{m_i} \text{ for all } j = 0, 1, 2, \cdots, n-1$$

  will be all normal queries before and after the signature query on message m_i. According to the response to normal queries, we have (the symbol $z^j_{m_i}$ is simplified into z^j_i for message m_i below)

  $$H(m_i|\Sigma^j_{m_i}) = g^{z^j_i}, \text{ for all } j = 0, 1, 2, \cdots, n-1.$$

  Then all block signatures are equal to

  $$\sigma_{j+1} = H(m_i|\Sigma^j_{m_i})^\alpha = \left(g^{z^j_i}\right)^{\hat{a}} = (g^{\hat{a}})^{z^j_i}, \text{ for all } j = 0, 1, 2, \cdots, n-1.$$

  Therefore, the simulator is able to compute $\Sigma_{m_i} = (\sigma_1, \sigma_2, \cdots, \sigma_n)$ on m_i for the adversary without knowing the secret key α.

Forgery: The adversary outputs a forged signature on a new message m^* to break the scheme where the signature on m^* was not queried before. Let the forged signature be

$$\Sigma_{m^*} = \Big(H(m^*|\Sigma^0_{m^*})^\alpha,\ H(m^*|\Sigma^1_{m^*})^\alpha, \cdots, H(m^*|\Sigma^{n-1}_{m^*})^\alpha\Big).$$

With the forged signature $\Sigma_{m^*} = \Sigma^n_{m^*}$, one can easily create a type-n query $m^*|\Sigma^n_{m^*}$ of m^*. Therefore, if the adversary can forge a valid signature on m^*, it is equivalent that the adversary must ever generate and make all hash queries in $\mathcal{H}^n(m^*)$ to the random oracle.

This completes the simulation and the reduction. The problem solution will appear in one of the hash queries to the random oracle if a solution query of **any message** is generated and made by the adversary. In the simulation phase, the simulator will continue the simulation until it receives a solution query unless it has to abort.

Our simulated scheme is indistinguishable from the real scheme from the view of the adversary if the simulation is successful. Each hash query x is responded with a random and independent integer z in the computation. All responses therefore are random and independent from the view of the adversary. Further, all simulated signatures are correct by the construction of the random oracle. The simulation therefore is indistinguishable from the real signature scheme.

Each hash query requires $O(1)$ exponentiations and each signature query requires the simulator to generate at most n BLS signatures. Let the success probability of receiving a solution query from the adversary be $\Pr[Suc]$ when the adversary can successfully forge a valid signature on m^*. We obtain the results given in theorem and complete the proof. □

4.2 Probability Analysis

Theorem 3. *Let the positive values $a_0, a_1, \cdots, a_{n-1}$ in Theorem 2 be a geometric sequence satisfying:*

$$a_j = \frac{d-1}{2d(d^n-1)} \cdot d^j :\ j \in [0, n-1],$$

for the integer d satisfying $d^n = q$. Then $\Pr[Suc]$ in Theorem 2 satisfies

$$\Pr[Suc] \geq \frac{1}{4q^{\frac{1}{n}}},$$

where q is the number of signature queries and n is the scheme parameter.

Proof. We calculate the success probability $\Pr[Suc]$ on the condition that the adversary can successfully forge a valid signature. Namely,

$$\Pr[Suc] = \Pr[\text{The adversary made a solution query}|\mathsf{Verify}(pk, \Sigma_{m^*}, m^*) = \mathsf{true}].$$

Let $m_1, m_2, m_3, \cdots, m_q$ be the order of messages selected for signature queries by the adversary before it forges a valid signature on a new message m^*. We have the following important parameters in Theorem 2.

	m_1	m_2	m_3	$\cdots$	m_q	m^*
The adversary's adaptive choices	k_1	k_2	k_3	$\cdots$	k_q	k^*
The simulator's non-uniform choices	c_1	c_2	c_3	$\cdots$	c_q	c^*

- For each message m_i before its signature query, the adversary will generate and make type-0, type-1, $\cdots$, type-k_i queries of m_i, namely $\mathcal{H}^{k_i}(m_i)$. The integer $k_i \in [0, n-1]$ is adaptively decided by the adversary.
- For the message m^*, a successful forgery $\Sigma_{m^*} = \Sigma^n_{m^*}$ is equivalent to that the adversary will generate and make hash queries $\mathcal{H}^n(m^*)$ to the random oracle. Therefore, we have $k^* = n$.
- For each message m including $(m_1, m_2, \cdots, m_q, m^*)$, the simulator will set its type-c_m query, namely $m|\Sigma^{c_m}_m$, as a challenge query. We have that $c_m \in [0, n-1]$ for each message is chosen independently with

$$\Pr[c_m = j] = a_j = \frac{d-1}{2d(d^n-1)} \cdot d^j.$$

The reduction is successful if one of the hash queries of any message generated and made by the adversary is a solution query. We note that the solution query does not have to be of message m^*.

Let $\mathcal{Q}^*_{\mathcal{A}}$ be the mixture of hash queries and signature queries made by the adversary in attacking the scheme before returning a forged signature on m^*. In the query sequence $\mathcal{Q}^*_{\mathcal{A}}$, we have that

- The adversary makes queries $\mathcal{H}^{k_i}(m_i) = \{m_i|\Sigma^0_{m_i}, m_i|\Sigma^1_{m_i}, \cdots, m_i|\Sigma^{k_i}_{m_i}\}$ before the signature query on m_i denoted by $\mathcal{S}(m_i)$.
- The adversary makes queries $\mathcal{H}^n(m^*) = \{m^*|\Sigma^0_{m^*}, m^*|\Sigma^1_{m^*}, \cdots, m^*|\Sigma^n_{m^*}\}$ on the message m^*.
- The adversary could generate and make hash queries of any message $m \notin \{m_1, m_2, \cdots, m_q, m^*\}$ without signature query on m.
- The exact sequences of each hash query and each signature query are adaptively decided by the adversary. The adversary could make all hash queries first before signature queries, or will only make hash queries of m_i before signature query on m_i.

The success probability of reduction is rewritten as $\Pr[Suc] = \Pr[\mathcal{Q}^*_{\mathcal{A}}]$, where $\Pr[\mathcal{Q}^*]$ denotes the success probability of reduction against the adversary who makes hash/signature queries according to the sequence $\mathcal{Q}^*$.

It is hard to directly analyze the success probability $\Pr[\mathcal{Q}^*_{\mathcal{A}}]$ because the exact sequence of each hash query and each signature query are unknown by the simulator at the beginning. We solve this difficulty by firstly (1) analyzing the probability $\Pr[\mathcal{Q}^*_{2,0}]$ instead, where $\mathcal{Q}^*_{2,0}$ is a well-format and simplified query sequence whose success probability of reduction is easy in analysis, and then (2) proving that $\Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,0}]$.

More precisely, the query sequence $\mathcal{Q}^*_{2,0}$ is defined as $\mathcal{H}^0(m_1) \rightarrow \mathcal{S}(m_1) \rightarrow \mathcal{H}^0(m_2) \rightarrow \mathcal{S}(m_2) \rightarrow \cdots \rightarrow \mathcal{H}^0(m_q) \rightarrow \mathcal{S}(m_q) \rightarrow \mathcal{H}^n(m^*)$, where the adversary only makes type-0 query of m_i after $\mathcal{S}(m_{i-1})$ and before $\mathcal{S}(m_i)$ for all $i \in [1, q]$, and the adversary makes hash queries $\mathcal{H}^n(m^*)$ of m^*.

Next we prove

– $\Pr[\mathcal{Q}^*_{2,0}] \geq \frac{1}{4 \cdot q^{\frac{1}{n}}}$ in Lemma 2.
– $\Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,0}]$ in Lemma 3.

This completes the proof of $\Pr[Suc] = \Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,0}] \geq \frac{1}{4q^{1/n}}$. □

Lemma 2. $\Pr[\mathcal{Q}^*_{2,0}] \geq \frac{1}{4 \cdot q^{\frac{1}{n}}}$.

Proof. The probability $\Pr[\mathcal{Q}^*_{2,0}]$ is calculated in the way that the simulator does not abort in the signature query phase on messages $(m_1, m_2, \cdots, m_q)$ and the solution query appears in one of the hash queries in $\mathcal{H}^n(m^*)$.

The simulator aborts due to the signature query $\mathcal{S}(m_i)$ if and only if the type-0 query of m_i is set as the challenge query. Since the simulator will set the type-c_i query as a challenge query for m_i with probability $\Pr[c_i = 0] = a_0$, the simulator will not abort in the signature query phase with probability

$$\begin{aligned}
\prod_{i=1}^{q} \Big(1 - \Pr[c_i = 0]\Big) = \Big(1 - a_0\Big)^q &= \Big(1 - \frac{d-1}{2d(d^n - 1)}\Big)^q \\
&= \Big(1 - \frac{1}{2d(1 + d + d^2 + \cdots + d^{n-1})}\Big)^q \\
&\geq \Big(1 - \frac{1}{2d^n}\Big)^q \\
&= \Big(1 - \frac{1}{2q}\Big)^q \\
&\geq \Big(1 - \frac{1}{2 \cdot 1}\Big)^1 = \frac{1}{2}.
\end{aligned}$$

For the hash queries of message m^*, the simulator sets the type-c^* query as a challenge query. Since the adversary makes hash queries $\mathcal{H}^n(m^*)$, we have that the solution query appears in $\mathcal{H}^n(m^*)$ as long as $0 \leq c^* \leq n-1$.

$$\begin{aligned}
\Pr[c^* \leq n-1] &= a_0 + a_1 + \cdots + a_{n-1} \\
&= \frac{d-1}{2d(d^n-1)} + \frac{d-1}{2d(d^n-1)} \cdot d + \cdots + \frac{d-1}{2d(d^n-1)} \cdot d^{n-1} \\
&= \frac{d-1}{2d(d^n-1)} \Big(1 + d + d^2 + \cdots + d^{n-1}\Big) \\
&= \frac{d-1}{2d(d^n-1)} \cdot \frac{1-d^n}{1-d} \\
&= \frac{1}{2d}.
\end{aligned}$$

Therefore, we have

$$\Pr[\mathcal{Q}^*_{2,0}] = \prod_{i=1}^{q} \Big(1 - \Pr[c_i = 0]\Big) \cdot \Pr[c^* \leq n-1] \geq \frac{1}{2} \cdot \frac{1}{2d} = \frac{1}{4q^{\frac{1}{n}}}.$$

This completes the proof of the lemma.

Lemma 3. $\Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,0}]$.

The roadmap of the proof to this lemma is as follows. We define more intermediate query sequences, namely $\mathcal{Q}^*_{2,1}, \mathcal{Q}^*_{2,2}, \mathcal{Q}^*_{2,3}, \cdots, \mathcal{Q}^*_{2,q}$, and prove that

$$\Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,q}] \geq \cdots \geq \Pr[\mathcal{Q}^*_{2,1}] \geq \Pr[\mathcal{Q}^*_{2,0}].$$

- Firstly, we define the query sequences $\mathcal{Q}^*_{2,i}$ for all $i \in [1, q]$.
- Secondly, we prove $\Pr[\mathcal{Q}^*_{2,i}] \geq \Pr[\mathcal{Q}^*_{2,i-1}]$ for all $i \in [1, q]$.
- Finally, we prove $\Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,q}]$.

Proof. We define the query sequence $\mathcal{Q}^*_{2,i}$ for each $i \in [1, q]$ based on $\mathcal{Q}^*_{2,0}$ and the adaptive integers $(k_1, k_2, \cdots, k_i)$ in $\mathcal{Q}^*_{\mathcal{A}}$. We define $\mathcal{Q}^*_{2,i} = \mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1) \to \mathcal{H}^{k_2}(m_2) \to \mathcal{S}(m_2) \to \cdots \to \mathcal{H}^{k_i}(m_i) \to \mathcal{S}(m_i) \to \mathcal{H}^0(m_{i+1}) \to \mathcal{S}(m_{i+1}) \to \cdots \to \mathcal{H}^0(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*)$.

The only difference is the number of hash queries of $(m_1, m_2, \cdots, m_i)$ when compared to $\mathcal{Q}^*_{2,0}$. In $\mathcal{Q}^*_{2,0}$, the hash queries of messages are

$$\mathcal{Q}^*_{2,0}: \quad (m_1, m_2, \cdots, m_q) \Rrightarrow \Big(\mathcal{H}^0(m_1), \mathcal{H}^0(m_2), \cdots, \mathcal{H}^0(m_q)\Big).$$

While in the query sequence $\mathcal{Q}^*_{2,i}$, we define

$$\mathcal{Q}^*_{2,i}: \begin{cases} (m_1, \quad m_2, \quad \cdots, m_i) \Rrightarrow \Big(\mathcal{H}^{k_1}(m_1), \ \mathcal{H}^{k_2}(m_2), \cdots, \quad \mathcal{H}^{k_i}(m_i)\Big) \\ (m_{i+1}, m_{i+2}, \cdots, m_q) \Rrightarrow \Big(\mathcal{H}^0(m_{i+1}), \mathcal{H}^0(m_{i+2}), \cdots, \mathcal{H}^0(m_q)\Big) \end{cases}$$

Next, we compare the success probabilities of reduction under the two query sequences $\mathcal{Q}^*_{2,i-1}$ and $\mathcal{Q}^*_{2,i}$. The two query sequences $\mathcal{Q}^*_{2,i-1}$ and $\mathcal{Q}^*_{2,i}$ are compared as follows.

$$\mathcal{H}^{k_1}(m_1) \to \mathcal{S}(m_1) \to \cdots \mathcal{S}(m_{i-1}) \begin{cases} \nearrow \\ \searrow \end{cases} \begin{matrix} \mathcal{H}^{0}(m_i) \ \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}: \ \mathcal{Q}^*_{2,i-1} \\ \mathcal{H}^{k_i}(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}: \ \mathcal{Q}^*_{2,i} \end{matrix}$$

where $\mathcal{Q}^*_{2,[>i]}$ is a truncated query sequence of $\mathcal{Q}^*_{2,0}$ defined as

$$\mathcal{Q}^*_{2,[>i]} = \mathcal{H}^0(m_{i+1}) \to \mathcal{S}(m_{i+1}) \to \cdots \to \mathcal{H}^0(m_q) \to \mathcal{S}(m_q) \to \mathcal{H}^n(m^*).$$

The two query sequences $\mathcal{Q}^*_{2,i-1}$ and $\mathcal{Q}^*_{2,i}$ have the identical sub-sequence before $\mathcal{S}(m_{i-1})$, and then the reduction will have the identical success probability before $\mathcal{S}(m_{i-1})$ in these two query sequences. The two query sequences have the only difference that the adversary makes hash queries $\mathcal{H}^{k_i}(m_i)$ in $\mathcal{Q}^*_{2,i}$ instead of $\mathcal{H}^0(m_i)$ in $\mathcal{Q}^*_{2,i-1}$. Therefore, we have

$$\Pr[\mathcal{H}^{k_i}(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}] \geq \Pr[\mathcal{H}^0(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}] \qquad (1)$$

implies that $\Pr[\mathcal{Q}^*_{2,i}] \geq \Pr[\mathcal{Q}^*_{2,i-1}]$.

Next, we prove the correctness of the inequality (1). In both the two query sub-sequences $\mathcal{H}^0(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}$ and $\mathcal{H}^{k_i}(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}$, the simulator will either (1) obtain a solution query of m_i or (2) obtain a solution query from $\mathcal{Q}^*_{2,[>i]}$ when the simulator neither succeeds nor aborts after $\mathcal{S}(m_i)$. We have

$$\Pr[\mathcal{H}^0(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}\ ] = \Pr[S_i^{(0)}]\ + \Big(1 - \Pr[S_i^{(0)}]\ - \Pr[F_i^{(0)}]\ \Big) \Pr[\mathcal{Q}^*_{2,[>i]}]$$

$$\Pr[\mathcal{H}^{k_i}(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}] = \Pr[S_i^{(k_i)}] + \Big(1 - \Pr[S_i^{(k_i)}] - \Pr[F_i^{(k_i)}]\Big) \Pr[\mathcal{Q}^*_{2,[>i]}]$$

where $S_i^{(k_i)}$ is the event that the solution query appears in $\mathcal{H}^{k_i}(m_i)$, and $F_i^{(k_i)}$ is the corresponding event that the simulator fails due to $\mathcal{S}(m_i)$. In particular, we have $\Pr[S_i^{(0)}] = 0$ and $\Pr[F_i^{(0)}] = a_0$.

We have the following results hold according to the setting of a_i.

$$\begin{aligned}
\Pr[S_i^{(k_i)}] - \Pr[S_i^{(0)}] &= \Big(\Pr[c_i \le k_i - 1]\Big) - 0 \\
&= a_0 + a_1 + \cdots + a_{k_i - 1} \\
&= \frac{d-1}{2d(d^n-1)} + \frac{d-1}{2d(d^n-1)} \cdot d + \cdots + \frac{d-1}{2d(d^n-1)} \cdot d^{k_i - 1} \\
&= \frac{d-1}{2d(d^n-1)} \Big(1 + d + d^2 + \cdots + d^{k_i - 1}\Big) \\
&= \frac{d-1}{2d(d^n-1)} \cdot \frac{1 - d^{k_i}}{1-d} \\
&= \frac{d^{k_i} - 1}{2d(d^n-1)}
\end{aligned}$$

$$\begin{aligned}
\Pr[F_i^{(k_i)}] - \Pr[F_i^{(0)}] &= \Big(\Pr[c_i = k_i]\Big) - \Big(\Pr[c_i = 0]\Big) \\
&= a_{k_i} - a_0 \\
&= \frac{d-1}{2d(d^n-1)} \cdot d^{k_i} - \frac{d-1}{2d(d^n-1)} \cdot d^0 \\
&= \frac{d^{k_i} - 1}{2d(d^n-1)} \cdot (d-1)
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
&\Pr[\mathcal{H}^{k_i}(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}] - \Pr[\mathcal{H}^0(m_i) \to \mathcal{S}(m_i) \to \mathcal{Q}^*_{2,[>i]}] \\
&= \Big(\Pr[S_i^{(k_i)}] - \Pr[S_i^{(0)}]\Big) + \Pr[\mathcal{Q}^*_{2,[>i]}] \cdot \Big(\Pr[S_i^{(0)}] - \Pr[S_i^{(k_i)}] + \Pr[F_i^{(0)}] - \Pr[F_i^{(k_i)}]\Big) \\
&= \frac{d^{k_i} - 1}{2d(d^n-1)} + \Pr[\mathcal{Q}^*_{2,[>i]}] \cdot \Big(- \frac{d^{k_i} - 1}{2d(d^n-1)} - \frac{d^{k_i} - 1}{2d(d^n-1)} \cdot (d-1)\Big) \\
&= \frac{d^{k_i} - 1}{2d(d^n-1)} - \Pr[\mathcal{Q}^*_{2,[>i]}] \cdot \frac{d^{k_i} - 1}{2d(d^n-1)} \cdot d \\
&= \frac{d^{k_i} - 1}{2d(d^n-1)} \left(1 - \Pr[\mathcal{Q}^*_{2,[>i]}] \cdot d\right),
\end{aligned}$$

which is positive (≥ 0) because we have

- $d \geq 1$ since $d = q^{\frac{1}{n}}$ and $q \geq 1$.
- The inequality $\Pr[\mathcal{Q}^*_{2,[>i]}] \leq \Pr[\mathcal{H}^n(m^*)]$ holds since making type-0 query and then signature query will decrease the success probability of reduction only, such that

$$\begin{aligned}\Pr[\mathcal{Q}^*_{2,[>i]}] &= \Pr[\mathcal{H}^0(m_{i+1}) \rightarrow \mathcal{S}(m_i) \rightarrow \cdots \rightarrow \mathcal{H}^0(m_q) \rightarrow \mathcal{S}(m_q) \rightarrow \mathcal{H}^n(m^*)] \\ &\leq \Pr[\mathcal{H}^n(m^*)] \\ &= \Pr[c^* \leq n-1] \\ &= \frac{1}{2d} \leq \frac{1}{d}\end{aligned}$$

Therefore, from the above analysis, we obtain $\Pr[\mathcal{Q}^*_{2,i}] \geq \Pr[\mathcal{Q}^*_{2,i-1}]$.

Finally, we prove $\Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,q}]$. In comparison with $\mathcal{Q}^*_{2,q}$, $\mathcal{Q}^*_{\mathcal{A}}$ generates and makes the same hash queries of $(m_1, m_2, \cdots, m_q, m^*)$.

$$\mathcal{Q}^*_{\mathcal{A}}, \mathcal{Q}^*_{2,q}: \quad (m_1, m_2, \cdots, m_q, m^*) \Rrightarrow \Big(\mathcal{H}^{k_1}(m_1), \mathcal{H}^{k_2}(m_2), \cdots, \mathcal{H}^{k_q}(m_q), \mathcal{H}^n(m^*)\Big).$$

The only differences are as follows.

- In the real sequence $\mathcal{Q}^*_{\mathcal{A}}$, the adversary could also make hash queries of any message m whose signature was not queried by the adversary.
- In the real sequence $\mathcal{Q}^*_{\mathcal{A}}$, the adversary could make the mixture of hash queries and signature queries without following the well-format sequence. For example, the sub-sequence of $\mathcal{Q}^*_{\mathcal{A}}$ is $\mathcal{H}^{k_1}(m_1) \rightarrow \mathcal{H}^{k_2}(m_2) \rightarrow \mathcal{S}(m_1)$ instead of $\mathcal{H}^{k_1}(m_1) \rightarrow \mathcal{S}(m_1) \rightarrow \mathcal{H}^{k_2}(m_2)$ in $\mathcal{Q}^*_q$, where the adversary made hash queries of message m_2 before the signature query on m_1.

Note that any hash query will be responded by the random oracle correctly without abort. Therefore, any query sequence $\mathcal{Q}^*_{\mathcal{A}}$ different from $\mathcal{Q}^*_{2,q}$ would increase the success probability of reduction and then we have $\Pr[\mathcal{Q}^*_{\mathcal{A}}] \geq \Pr[\mathcal{Q}^*_{2,q}]$.

This completes the proof of the lemma. □

Acknowledgements We would like to thank Tibor Jager for insightful discussions on the first version of this work in 2020. We would also like to thank the anonymous reviewers from Eurocrypt 2021, Crypto 2021, and Eurocrypt 2022 for their constructive comments.

References

1. 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)
2. Abe, M., Groth, J., Ohkubo, M.: Separating short structure-preserving signatures from non-interactive assumptions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011. LNCS, vol. 7073, pp. 628–646. Springer (2011)

3. Abe, M., Hofheinz, D., Nishimaki, R., Ohkubo, M., Pan, J.: Compact structure-preserving signatures with almost tight security. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 548–580. Springer (2017)
4. Abe, M., Jutla, C.S., Ohkubo, M., Pan, J., Roy, A., Wang, Y.: Shorter QA-NIZK and SPS with tighter security. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 669–699. Springer (2019)
5. 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)
6. Bellare, M., Poettering, B., Stebila, D.: From identification to signatures, tightly: A framework and generic transforms. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 435–464 (2016)
7. Bellare, M., Rogaway, P.: The exact security of digital signatures - how to sign with RSA and rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer (1996)
8. 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)
9. 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)
10. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer (2004)
11. 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)
12. Boneh, D., Venkatesan, R.: Breaking RSA may not be equivalent to factoring. In: Nyberg, K. (ed.) EUROCRYPT '98. LNCS, vol. 1403, pp. 59–71. Springer (1998)
13. Boyen, X., Li, Q.: Towards tightly secure lattice short signature and id-based encryption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 404–434 (2016)
14. Chailloux, A., Debris-Alazard, T.: Tight and optimal reductions for signatures based on average trapdoor preimage sampleable functions and applications to code-based signatures. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 453–479. Springer (2020)
15. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 767–797. Springer (2019)
16. 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)
17. Diemert, D., Gellert, K., Jager, T., Lyu, L.: Digital signatures with memory-tight security in the multi-challenge setting. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13093, pp. 403–433. Springer (2021)
18. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 1–31. Springer (2021)
19. El Kaafarani, A., Katsumata, S., Pintore, F.: Lossy csi-fish: Efficient signature scheme with tight reduction to decisional CSIDH-512. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 157–186. Springer (2020)

20. Fischlin, M., Fleischhacker, N.: Limitations of the meta-reduction technique: The case of schnorr signatures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 444–460. Springer (2013)
21. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for schnorr signatures. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 512–531. Springer (2014)
22. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 201. LNCS, vol. 10992, pp. 33–62. Springer (2018)
23. Garg, S., Bhaskar, R., Lokam, S.V.: Improved bounds on security reductions for discrete log based signatures. In: Wagner, D.A. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 93–107. Springer (2008)
24. Gay, R., Hofheinz, D., Kohl, L., Pan, J.: More efficient (almost) tightly secure structure-preserving signatures. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 230–258. Springer (2018)
25. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 95–125. Springer, Cham (2018)
26. Goh, E., Jarecki, S.: A signature scheme as secure as the diffie-hellman problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 401–415. Springer (2003)
27. Guo, F., Chen, R., Susilo, W., Lai, J., Yang, G., Mu, Y.: Optimal security reductions for unique signatures: Bypassing impossibilities with a counterexample. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 517–547. Springer (2017)
28. Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 670–700. Springer (2021)
29. Han, S., Liu, S., Gu, D.: Key encapsulation mechanism with tight enhanced security in the multi-user setting: Impossibility result and optimal tightness. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13091, pp. 483–513. Springer (2021)
30. Hesse, J., Hofheinz, D., Kohl, L.: On tightly secure non-interactive key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 65–94. Springer (2018)
31. Hesse, J., Hofheinz, D., Kohl, L., Langrehr, R.: Towards tight adaptive security of non-interactive key exchange. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13044, pp. 286–316. Springer (2021)
32. Hofheinz, D.: Algebraic partitioning: Fully compact and (almost) tightly secure cryptography. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 251–281. Springer (2016)
33. 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)
34. 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)
35. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F. (eds.) EUROCRYPT 2021. LNCS, vol. 12696, pp. 117–146. Springer (2021)

36. Jutla, C.S., Ohkubo, M., Roy, A.: Improved (almost) tightly-secure structure-preserving signatures. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 123–152. Springer (2018)
37. 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)
38. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM CCS 2003. pp. 155–164. ACM (2003)
39. Kiltz, E., Loss, J., Pan, J.: Tightly-secure signatures from five-move identification protocols. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 68–94. Springer (2017)
40. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 33–61. Springer (2016)
41. Lewko, A.B., Waters, B.: Why proving HIBE systems secure is difficult. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 58–76. Springer (2014)
42. Libert, B., Joye, M., Yung, M., Peters, T.: Concise multi-challenge cca-secure encryption and signatures with almost tight security. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 1–21. Springer (2014)
43. Micali, S., Reyzin, L.: Improving the exact security of digital signature schemes. J. Cryptol. **15**(1), 1–18 (2002)
44. Morgan, A., Pass, R.: On the security loss of unique signatures. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11239, pp. 507–536. Springer, Cham (2018)
45. Morgan, A., Pass, R., Shi, E.: On the adaptive security of macs and prfs. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020. LNCS, vol. 12491, pp. 724–753. Springer (2020)
46. Niehues, D.: Verifiable random functions with optimal tightness. In: Garay, J.A. (ed.) PKC 2021. LNCS, vol. 12711, pp. 61–91. Springer (2021)
47. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer (2005)
48. Rotem, L., Segev, G.: Tighter security for schnorr identification and signatures: A high-moment forking lemma for Σ-protocols. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12825, pp. 222–250. Springer (2021)
49. 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)
50. Seurin, Y.: On the exact security of schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 554–571. Springer (2012)
51. Shacham, H.: Short unique signatures from RSA with a tight security reduction (in the random oracle model). In: Meiklejohn, S., Sako, K. (eds.) Financial Cryptography and Data Security 2018. LNCS, vol. 10957, pp. 61–79. Springer (2018)