Galindo-Garcia Identity-Based Signature, Improved

Sanjit Chatterjee and Chethan Kamath,

Indian Institute of Science, Bangalore.

{sanjit,chethan0510}@csa.iisc.ernet.in

November 14, 2013

Abstract

In Africacrypt 2009, Galindo and Garcia proposed a lightweight identity-based signature (IBS) scheme based on the Schnorr signature. The construction is simple and claimed to be the most efficient IBS till date. The security is argued, primarily, by using the Multiple-Forking (MF) Algorithm which is used to launch nested replay attack on the adversary and, consequently, contradict the discrete-log assumption. However, this security argument was shown to be flawed, and subsequently fixed, by Chatterjee *et al.*. The resulting security, though, is still quite *loose* with a tightness gap of $\mathrm{O}\left(q^6\right)$ (where q denotes the bound on the number of queries to the random oracle). The loss of tightness is, in fact, *inherited* from the MF Algorithm. In this paper, we contemplate a better security bound for Galindo-Garcia IBS (GG-IBS). To this end, we introduce two notions pertaining to the simulation of random oracles: "dependency" and "independency". The notion of independency follows naturally for GG-IBS; dependency, on the other hand, has to be induced by modifying the construction of the protocol in a clever manner. It turns out that the two notions can be applied in conjunction, and this leads to the nested replay attack being launched far more *effectively* than using the MF Algorithm. As a result, the effective degradation is reduced to $\mathrm{O}\left(q^3\right)$. The non-trivial aspect is to leverage these two notions in the security argument.

Keywords: Identity-Based Signature, Galindo-Garcia IBS, Multiple-Forking Lemma, Replay Attack, Provable Security, Tightness.

Contents

1 Introduction

In Africacrypt'09, Galindo and Garcia [GG09] used the technique of concatenated Schnorr's signature [Sch91] to propose an identity-based signature (IBS) [Sha85] that works in the discrete-log setting but does not require pairing. The authors came up with a security proof of the proposed IBS scheme in the standard EU-ID-CMA model using the Multiple-Forking (MF) Lemma [BN06, BPW12, PS00], and after a comparative study concluded that the proposed construction has an overall better performance than the existing RSA-based [FS87, GQ90] and pairing-based schemes [CC02, Her05, Hes03]. The Galindo-Garcia IBS (GG-IBS), due to its efficiency and simplicity, has been used as a building block for a couple of other cryptosystems [RS11, XW12].

Security. The security was argued in [GG09] through two reductions, $\mathcal{B}_1$ and $\mathcal{B}_2$, the choice of which is determined by an event E. The authors construct $\mathcal{B}_1$ to solve the DLP when the event E occurs; on the other hand, $\mathcal{B}_2$ is used to solve the DLP in case the complement of E occurs. Both the reductions use the Multiple-Forking (MF) Lemma [BPW12] to show that the DLP is reduced to breaking the IBS scheme. However, Chatterjee *et al.* [CKK13] made several observations on this security argument. In particular, they showed that the reduction $\mathcal{B}_1$ fails to provide a proper simulation of the unforgeability game in the standard security model for IBS [BNN04], while the reduction $\mathcal{B}_2$ is incomplete. They gave a new argument which consists of three reductions: $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$. At a high level, $\mathcal{R}_1$ addressed the problems identified in the original $\mathcal{B}_1$, while $\mathcal{R}_2$ and $\mathcal{R}_3$, together, addressed the incompleteness of the original $\mathcal{B}_2$. A comparison of the two arguments is given in **Table 1**. We refer the reader to the full version of [CKK13] for more (and updated) details.

1.1 Oracle Replay Attack: Rewinding vs. General Forking

As already noted, the security of GG-IBS is aided, primarily, by the Multiple-Forking (MF) Algorithm which is used to launch nested replay attack on the adversary. The concept of multiple forking finds its roots in the works of Pointcheval and Stern [PS00]. They came up with the machinery of oracle replay attack[1] to argue the security of a large class[2] of signature schemes [ElG85, Sch91, Oka93]. In the basic version of replay attack, one runs the adversary

This is a precursor to [CK13].

[1] We will use the terms forking and (the process of launching) oracle replay attack interchangeably.

[2] To be precise, the signature schemes obtained from three-round identification schemes (Σ-protocols) through the Fiat-Shamir transformation [FS87].

[GG09]	$\mathcal{B}_1$	$\mathcal{B}_2$	
Degradation	$\mathrm{O}\left(q_{\mathrm{G}}^3/\epsilon\right)$	$\mathrm{O}\left((q_{\mathrm{G}}q_{\mathrm{H}})^6/\epsilon^3\right)$	
Forking Algorithm	$\mathcal{M}_{\mathcal{W},1}$	$\mathcal{M}_{\mathcal{W},3}$	
[CKK13]	$\mathcal{R}_1$	$\mathcal{R}_2$	$\mathcal{R}_3$
Degradation	$\mathrm{O}\left(q_{\mathrm{G}}q_{\varepsilon}/\epsilon\right)$	$\mathrm{O}\left((q_{\mathrm{H}}+q_{\mathrm{G}})^2/\epsilon\right)$	$\mathrm{O}\left((q_{\mathrm{H}}+q_{\mathrm{G}})^6/\epsilon^3\right)$
Forking Algorithm	$\mathcal{F}_{\mathcal{W}}$	$\mathcal{M}_{\mathcal{W},1}$	$\mathcal{M}_{\mathcal{W},3}$
Our Result	$\mathcal{R}'_1$	$\mathcal{R}'_2$	
Degradation	$\mathrm{O}\left(q_{\mathrm{G}}q_{\varepsilon}/\epsilon\right)$	$\mathrm{O}\left((q_{\mathrm{G}}+q_{\mathrm{H}})^3/\epsilon^3\right)$	
Forking Algorithm	$\mathcal{F}_{\mathcal{W}}$	Nested forking via rewinding	

Table 1: A comparison of degradation for the security arguments in [GG09], [CKK13] and this paper. As usual, q_{G} and q_{H} denote the upper bound on the respective hash oracle queries, whereas, q_{ε} denotes upper bound on the extract queries.

twice (by rewinding[3] it, see **Figure 1**), on related inputs, in order to solve the underlying hard problem. The probability of success of the replay attack is bounded by the Forking Lemma of [PS00] which, informally, states that the cost of the forking is roughly $\mathrm{O}(q)$. The Splitting Lemma plays a crucial role in establishing this bound.

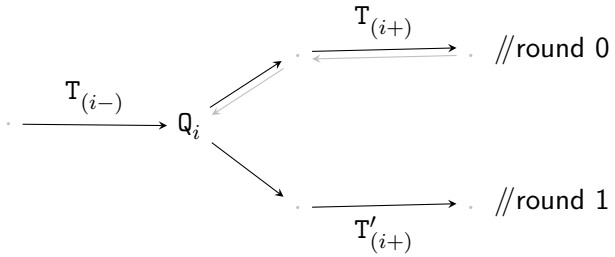

Figure 1: An elementary oracle replay attack: the adversary is "forked" at the point $\mathtt{Q}_i$. Here, T denotes the tape involved in the simulation.

The abstraction: general forking Bellare and Neven [BN06] proposed a more abstract version called the General Forking (GF) Lemma. The concept of general forking is formulated in terms of randomised algorithms and its outputs, leaving out the notions of signature scheme as

[3]We discuss a rather naïve approach that could be used to rewind the adversary, which is modelled as a probabilistic Turing machine. The state of progress of a Turing machine is (completely) determined by its current state (from the state register), the contents of the tape and the position of the head on the tape. The basic idea involved in rewinding is as follows. The simulator explicitly saves the state of progress of the adversary every time it queries the random oracle. To be precise, it saves the current state (from the state register), the contents of the tape and the position of the head on a separate, external, tape. The simulator, thus, may have to save at most q states of progress. At the end of the round, the simulator has sufficient information about the critical point. Besides, the state of progress of the adversary at this critical point–the critical state of progress–is highly likely to be one of the saved states of progress. Therefore, in order to rewind the adversary, the simulator has to just look up for this critical state of progress and restore the adversary to it.

well as random oracles [BR93] altogether. Rewinding is achieved by fixing the randomness of the adversary and, then, running it twice. The claimed advantage is to allow for more modular and easy to verify proof of cryptographic schemes that apply the notion of forking in their security argument. The gap between the abstract and the concrete is, then, bridged using the so-called "wrapper" algorithm. While the GF/MF Algorithm takes care of the replay attack, it is the wrapper that handles the simulation of the protocol environment to the actual adversary. The reduction consists of invoking the appropriate forking algorithm (on the associated wrapper) and utilising its outputs to solve the underlying hard problem. The analysis, as well, proceeds in a different manner: it is similar to that of a randomised algorithm, with the adversary (the wrapper, to be precise) modelled as a random variable.[4] The cost of forking, though, is still $\mathrm{O}(q)$.

Multiple forking. The concept of general forking was further generalised by Boldyreva *et al.* [BPW12] leading to the Multiple Forking (MF) Lemma. The immediate motivation behind this new abstraction was to argue the security of a proxy signature scheme that uses more than one hash functions. The hash functions are modelled as random oracles and the MF Algorithm allows one to mount the so-called *nested* replay attacks by rewinding the adversary several times on related inputs. In particular, a nested oracle replay attack involves two random oracles and multiple forkings on the two oracles (see **Figure 2**). The more abstract notion of multiple forking still retains the modularity advantage of general forking and has been applied in several other security arguments [GG09, CMW12, CKK13] in a more-or-less black-box fashion.

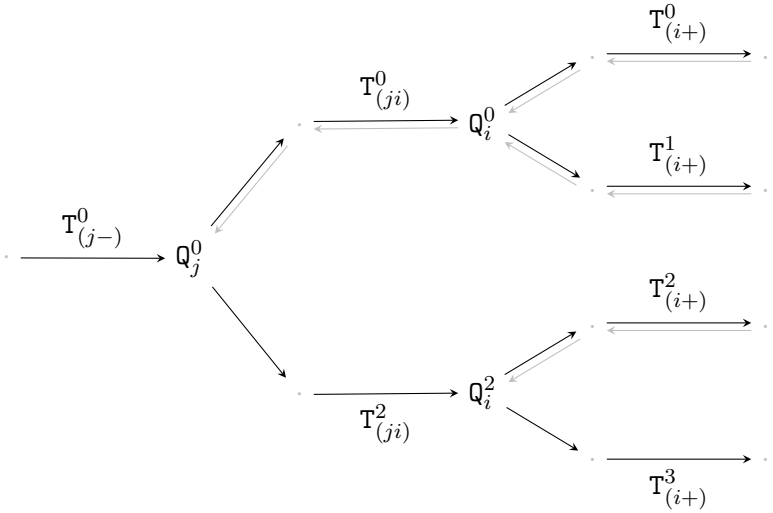

Figure 2: Nested replay attack with three forkings: the adversary is "forked" at the points $\mathtt{Q}^0_i$, $\mathtt{Q}^0_j$ and $\mathtt{Q}^2_i$, in that order. Again, $\mathtt{T}$ denotes the tape involved in the simulation.

The cost of multiple forking is determined by the number of times the adversary is forked: if it is denoted by n ($\mathcal{M}_{\mathcal{W},n}$), the cost is roughly $\mathrm{O}\left(q^{2n}\right)$. By implication, it costs the challenger $\mathrm{O}\left(q^2\right)$ per forking.

1.2 Our Contribution

Even though Chatterjee *et al.* had managed to fix the security argument for GG-IBS, the security bound that they ended up with (in [CKK13, **Theorem 1**]) is still quite loose. The loss of tightness is, in fact, *inherited* from the MF Algorithm (see **Algorithm 2**) which is used to launch nested replay attack on the GG-IBS adversary. To be accurate, the use of MF

[4] We will see how the two analyses are connected by the characteristic expression in **§3.4**.

Algorithm with $n = 3$ ($\mathcal{M}_{\mathcal{W},3}$) in reduction $\mathcal{R}_3$ results in an effective degradation of $\mathrm{O}\left(q^6\right)$ (where q denotes $q_{\mathrm{H}} + q_{\mathrm{G}}$).

Our Contribution. In this paper, we contemplate a better security bound for GG-IBS and in particular for the reduction $\mathcal{R}_3$. We pin point the source of degradation for $\mathcal{R}_3$ and identity ways to reduce it. To this end, we introduce two notions pertaining to random oracles: "dependency" and "independency". Informally speaking, random-oracle dependency involves designing the hash functions (which are modelled as random oracles) in such a way that the cost of forking involving two random oracles (as in the MF Algorithm) is reduced to that with single random oracle (as in the GF Algorithm). Independency, on the other hand, involves the relaxing of an unnecessary condition that is imposed on the adversary involved in the (nested) replay attack. It turns out that the notion of independency follows naturally for GG-IBS; dependency, on the other hand, has to be induced by modifying the construction (to be precise, the hash functions H and G) of the protocol in a clever manner. Plus, the two notions can be applied in conjunction and this leads to the nested replay attack being launched far more *effectively* than specified in the MF Algorithm: the cost of forking involving two oracles (*i.e.*, $\mathrm{O}\left(q^2\right)$) is reduced to that involving a single oracle (*i.e.*, $\mathrm{O}\left(q\right)$). As a result, the effective degradation is reduced to $\mathrm{O}\left(q^3\right)$. The non-trivial aspect is to leverage these two notions in the analysis of the security argument. To achieve this, we *return* to the Pointcheval-Stern approach of analysis: *i.e.*, using the Splitting Lemma. A summary follows.

Outline of our analysis. As already mentioned, the novel contribution of the paper is the analysis technique. We re-introduce the Splitting Lemma which is used in analysing the security argument of Schnorr Signature in [PS00]. Recall that the MF Algorithm doesn't, directly, involve the Splitting Lemma. However, its analysis (by modelling the adversary as a random variable) is tantamount to a straightforward extension of the Pointcheval-Stern one, but, using a *Three*-Splitting Lemma[5] in place of the normal *Two*-Splitting Lemma. A close inspection reveals that the degradation incurred by the analysis is inherent to the *Three*-Splitting Lemma and, that, it's unlikely one can come up with a better bound. Our idea is to bypass the *Three*-Splitting Lemma altogether. In order to achieve this, we have to impart a structure to the underlying universe of random tapes which is, otherwise, unstructured. This is exactly where the notion of (in)dependency comes into the picture: it imparts a beautiful structure to the underlying tape-set which allows us to apply the *Two*-Splitting Lemma twice, in a self-similar fashion (see **§3.3** for details).

1.3 Notations

We adopt the notations commonly used in the literature. $s \in_R \mathbb{S}$ denotes picking an element s uniformly at random from the set $\mathbb{S}$. In general, $\{s_1, \ldots, s_n\} \in_R \mathbb{S}$ denotes picking elements $s_1, \ldots, s_n$ independently and uniformly at random from the set $\mathbb{S}$. In a similar manner, $s \xleftarrow{\$} \mathbb{S}$ and $\{s_1, \ldots, s_n\} \xleftarrow{\$} \mathbb{S}$ denote random sampling, but with some underlying probability distribution on $\mathbb{S}$. $(y_1, \ldots, y_n) \xleftarrow{\$} \mathcal{A}(x_1, \ldots, x_m)$ denotes a probabilistic algorithm $\mathcal{A}$ which takes as input $(x_1, \ldots, x_m)$ to produce output $(y_1, \ldots, y_n)$. Sometimes the internal coins ρ of this algorithm is given explicitly as an input. This is distinguished from the normal input using a semi-colon, *e.g.*, $y \leftarrow \mathcal{A}(x; \rho)$.

In reductionist security arguments, $\Pi \leq_\gamma \mathsf{M}[\mathfrak{P}]$ denotes that Π is γ-reducible to $\mathfrak{P}$: given an adversary against $\mathfrak{P}$ in the security model M, one can construct an algorithm that breaks Π with a degradation of γ. $\Pi \overset{\gamma}{\Leftarrow} \mathsf{M}[\mathfrak{P}]$ conveys the same message. A straightforward generalisation is

[5] A generalisation of the Splitting Lemma that accommodates splitting the universe into *three* sets instead of just two (see **§2.3**.)

$\mathsf{M}'[\mathfrak{P}'] \leq_\gamma \mathsf{M}[\mathfrak{P}]$: given an adversary against $\mathfrak{P}$ in the security model M, one can construct an algorithm that breaks $\mathfrak{P}'$ in the security model M' with a degradation of γ.

Next, we introduce some notations pertaining to random oracles. The symbol $<$ is used to order the random oracle calls; *e.g.*, $\mathrm{H}(x) < \mathrm{G}(y)$ indicates that the random oracle call $\mathrm{H}(x)$ precedes the random oracle call $\mathrm{G}(y)$. More generally, $\mathrm{H} < \mathrm{G}$ indicates that the target H-oracle call precedes the target G-oracle call. The convention applies to hash functions as well. The symbol, on the other hand, $\prec$ is used to indicate random oracle dependency; *e.g.* $\mathrm{H} \prec \mathrm{G}$ indicates that the random oracle G is dependent on the random oracle H. In the discussion involving the forking algorithms, $\mathtt{Q}_j^i$ denotes the j^{th} random oracle query in round i of simulation.

Organisation of the paper. We provide the necessary background in **§2**. In **§3**, we describe the improved version of GG-IBS and its security argument. The major contribution in this section is the security analysis given in **§3.3**. Finally, we end with the concluding remarks in **§4**. As for the appendix, we begin with relevant forking algorithms and their associated lemmas in **Appendix A**. The reduction $\mathcal{R}_1'$ is given in **Appendix B**.

2 Background

We begin the section with the formal definition of IBS along with its security model. We also define the discrete-logarithm assumption–the hardness assumption on which GG-IBS is based on. Then, we describe the Splitting Lemma [PS00] which plays a crucial role in the new security argument for GG-IBS. Finally, for the sake of completeness, we describe the construction of GG-IBS and a sketch of the fixed security argument due to Chatterjee *et al.*.

2.1 Identity-Based Signatures

Definition 1 (Identity-Based Signature)**.** An IBS scheme consists of four polynomial-time, probabilistic algorithms $\{\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V}\}$ described below.

> Set-up, $\mathcal{G}(1^\kappa)$: It takes as input the security parameter κ. It outputs the master secret key $\mathtt{msk}$ and the master public key $\mathtt{mpk}$.
>
> Key Extraction, $\mathcal{E}(\mathtt{id}, \mathtt{msk})$: It takes as input the user's identity $\mathtt{id}$, the master secret key $\mathtt{msk}$ to generate a secret key $\mathtt{usk}$ for the user.
>
> Signing, $\mathcal{S}(\mathtt{id}, m, \mathtt{usk})$: It takes as input the user's identity $\mathtt{id}$, a message m and the user's secret key $\mathtt{usk}$ to generate a signature σ.
>
> Verification, $\mathcal{V}(\sigma, \mathtt{id}, m, \mathtt{mpk})$: It takes as input a signature σ, a message m, an identity $\mathtt{id}$ and the master public key $\mathtt{mpk}$. If outputs a bit b which is 1 if σ is a valid signature on $(\mathtt{id}, m)$ or 0 if the signature is invalid.

The standard *correctness* condition: $1 \leftarrow \mathcal{V}(\mathcal{S}(\mathtt{id}, m, \mathtt{usk}), \mathtt{id}, m, \mathtt{mpk})$, should be satisfied for all $\mathtt{id}$, m, $(\mathtt{msk}, \mathtt{mpk}) \xleftarrow{\$} \mathcal{G}(1^\kappa)$ and $\mathtt{usk} \xleftarrow{\$} \mathcal{E}(\mathtt{id}, \mathtt{msk})$. As for security, we use the detailed EU-ID-CMA model given in [BNN04].

Definition 2 (EU-ID-CMA Game[6])**.** The security of an IBS scheme in the EU-ID-CMA model is argued in terms of the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

[6]The security game in [BNN04], *i.e.* $\mathrm{Exp}^{\text{uf-cma}}_{\text{IBS},\mathrm{F}}$, is explained in terms of the three oracles: INIT, CORR and SIGN. Here we use an *equivalent* formulation in terms of Extract and Signature queries.

Set-up: $\mathcal{C}$ runs $\mathcal{G}$ to obtain the master public key $\mathtt{mpk}$ and the master secret key $\mathtt{msk}$. It passes $\mathtt{mpk}$ as the challenge master public key to $\mathcal{A}$.

Queries: $\mathcal{A}$ can adaptively make extract queries to an oracle $\mathcal{O}_{\mathcal{E}}$ and signature queries to an oracle $\mathcal{O}_{s}$. These queries are handled as follows.

- **Extract query**, $\mathcal{O}_{\mathcal{E}}(\mathtt{id})$: $\mathcal{A}$ asks for the secret key of a user with identity $\mathtt{id}$. If there has already been an extract query on $\mathtt{id}$, $\mathcal{C}$ returns the user secret key that was generated during the earlier query. Otherwise, $\mathcal{C}$ uses the knowledge of $\mathtt{msk}$ to run $\mathcal{E}$ and generate the user secret key $\mathtt{usk}$, which is then passed on to $\mathcal{A}$.
- **Signature query**, $\mathcal{O}_{s}(\mathtt{id}, m)$: $\mathcal{A}$ asks for the signature of a user with identity $\mathtt{id}$ on a message m. $\mathcal{C}$ first obtains, as specified the extract query, a user secret key $\mathtt{usk}$ corresponding to $\mathtt{id}$. Next, it uses the knowledge of $\mathtt{usk}$ to run $\mathcal{S}$ and generate a signature σ, which is passed to $\mathcal{A}$.

Forgery: $\mathcal{A}$ outputs a signature $\hat{\sigma}$ on an identity $\hat{\mathtt{id}}$ and a message $\hat{m}$, and *wins* the game if the forgery is *i) valid:* $\hat{\sigma}$ passes the verification on $(\hat{\mathtt{id}}, \hat{m})$; and *ii) non-trivial:* $\mathcal{A}$ has not queried the extract oracle with $\hat{\mathtt{id}}$, *nor* has it queried the signature oracle with $(\hat{\mathtt{id}}, \hat{m})$.

The advantage that $\mathcal{A}$ has in the above game, denoted by $\mathrm{Adv}_{\mathcal{A}}^{\mathtt{EU-ID-CMA}}(\kappa)$, is defined as the probability with which it wins the above game, *i.e.*

$$\Pr\left[1 \leftarrow \mathcal{V}(\hat{\sigma}, \hat{\mathtt{id}}, \hat{m}, \mathtt{mpk}) \mid (\mathtt{msk}, \mathtt{mpk}) \xleftarrow{\$} \mathcal{G}(1^{\kappa}); (\hat{\sigma}, \hat{\mathtt{id}}, \hat{m}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathcal{E}}(\cdot), \mathcal{O}_{s}(\cdot,\cdot)}(\mathtt{mpk})\right]$$

provided $\hat{\sigma}$ is a non-trivial forgery on $(\hat{\mathtt{id}}, \hat{m})$. An adversary $\mathcal{A}$ is said to be an $(\epsilon, t, q_{\mathcal{E}}, q_s)$-forger of an IBS scheme if it has advantage of at least ϵ in the above game, runs in time at most t and makes at most $q_{\mathcal{E}}$ [resp. q_s] extract [resp. signature] queries.

Definition 3 (ID-Security of IBS)**.** We say that an IBS is EU-ID-CMA-secure (ID-secure, in short) if for any polynomial-time adversary $\mathcal{A}$ the function $\mathrm{Adv}_{\mathcal{A}}^{\mathtt{EU-ID-CMA}}(\kappa)$ is negligible in κ.

Remark 1. If the security argument models the hash functions as random oracles, the security games are to be modified, appropriately, to accommodate queries to the random oracles.

2.2 Discrete-Logarithm Assumption

Let $\mathcal{G}_{\mathrm{DL}}$ be a (randomised) group generator: it takes as input a security parameter κ (in unary) and outputs $(\mathbb{G}, g, p)$, where $\mathbb{G}$ is a cyclic group of prime order p generated by g.[7] The discrete-logarithm problem (DLP) is defined through the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ (see **Figure 3**).

Definition 4 (DLP Game)**.** $\mathcal{C}$ invokes the group generator to generate a cyclic group: $(\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}_{\mathrm{DL}}(\kappa)$. Next, it sends to $\mathcal{A}$ the challenge DLP instance $(\mathbb{G}, g, p, h)$, where $h := g^{\alpha}$ with $\alpha \in_R \mathbb{Z}_p$. $\mathcal{A}$ wins the game if it correctly finds the discrete logarithm of h (with respect to the generator g), *i.e.*, $\alpha' = \alpha$. The advantage $\mathcal{A}$ has solving the DLP is the probability with which it wins the game, *i.e.*,

$$\mathrm{Adv}_{\mathcal{A}}^{\mathtt{DLP}}(\kappa) \stackrel{\mathrm{def}}{=} \Pr\left[\alpha' = \alpha : (\mathbb{G}, g, p) \xleftarrow{\$} \mathcal{G}_{\mathrm{DL}}(\kappa); \alpha \in_R \mathbb{Z}_p; \alpha' \leftarrow \mathcal{A}(\mathbb{G}, p, g, g^{\alpha})\right].$$

Definition 5 (Discrete-Logarithm Assumption)**.** The (ϵ, t)-discrete-log assumption holds if no adversary that takes takes time at most t has advantage at least ϵ in the DLP game. In general, the discrete-log assumption holds if for all PPT adversaries $\mathcal{A}$ the function $\mathrm{Adv}_{\mathcal{A}}^{\mathtt{DLP}}(\kappa)$ is negligible in κ; we say that $\mathcal{G}_{\mathrm{DL}}$ generates DL-secure groups [GG09].

[7] We sometimes omit the description of the group generator for sake of convenience and, also, consistency.

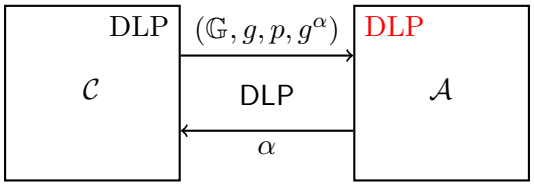

Figure 3: The DLP Game.

2.3 The Splitting Lemma

Let $\mathbb{X}$ and $\mathbb{Y}$ be two finite sets (with some underlying probability distribution) with $|\mathbb{X}| = m$ and $|\mathbb{Y}| = n$. Also, let $\mathbb{T}$ denote the "universal" set $\mathbb{X} \times \mathbb{Y}$. A pair $(x, y) \in \mathbb{X} \times \mathbb{Y}$ is deemed to be good if it satisfies a certain property. Let $\mathbb{V}$ denote the set of all such good pairs. Suppose that at least γ fraction of the pairs in $\mathbb{X} \times \mathbb{Y}$ are good, *i.e.*, $\Pr[\mathbb{V}] \geq \gamma$. We define a function $\mathrm{k} : \mathbb{X} \mapsto \mathfrak{P}(\mathbb{Y})$ as

$$\mathrm{k}(x) = \{y \in \mathbb{Y} \mid (x, y) \in \mathbb{V}\}.$$

k can be used to count, for a particular x, the number of elements $y \in \mathbb{Y}$ such that (x, y) forms a good pair. For some $\beta < \gamma$, we define the (sub)set of better pairs of $\mathbb{V}$ as

$$\mathbb{V}^* = \{(x, y) \in \mathbb{V} \mid |\mathrm{k}(x)| \geq (\gamma - \beta)n\}.$$

Thus, a good pair (x, y) is better if x forms good pairs with at least $(\gamma - \beta)$ fraction of the elements of $\mathbb{Y}$.

Lemma 1 (The Splitting Lemma, Pointcheval-Stern [PS00][8])**.** *The following propositions hold on the aforementioned assumptions:*

1. $\Pr[\mathbb{V}^*] \geq \beta$, *i.e., at least* β *fraction of pairs in* $\mathbb{T}$ *are* better.
2. $\Pr\left[(x, y') \in \mathbb{V} \mid (x, y) \in \mathbb{V}^*; y' \xleftarrow{\$} \mathbb{Y}\right] \geq \gamma - \beta$, *i.e., given a* better *pair* (x, y), *the probability with which the pair* (x, y') *(with* y' *sampled randomly from* $\mathbb{Y}$*) is* good *is at least* $\gamma - \beta$.
3. $\Pr[\mathbb{V}^* \mid \mathbb{V}] \geq \beta/\gamma$, *i.e., a* good *pair is* better *with a probability of at least* β/γ.

2.4 Galindo-Garcia IBS

We discuss, in brief, the construction and security argument of GG-IBS. We refer the reader to [CKK13] for a more detailed take on GG-IBS.

2.4.1 Construction

The scheme is based on the Schnorr signature scheme [Sch91]. The user secret key can be considered as the Schnorr signature by the PKG on the identity of the user, using the master secret key as the signing key. Analogously, the signature on a message by a user is the Schnorr signature, by that user, on the message using her user secret key as the signing key. The construction is given below.

[8] Also refer to [Kia07].

Set-up, $\mathcal{G}(1^\kappa)$: Generate a group $\mathbb{G} = \langle g \rangle$ of prime order p. Select $z \in_R \mathbb{Z}_p$ and set $Z = g^z$. Return z as the master secret key $\mathtt{msk}$ and $(\mathbb{G}, p, g, Z, \mathrm{H}, \mathrm{G})$ as the master public key $\mathtt{mpk}$, where H and G are hash functions

$$\mathrm{H} : \{0,1\}^* \mapsto \mathbb{Z}_p \quad \text{and} \quad \mathrm{G} : \{0,1\}^* \mapsto \mathbb{Z}_p.$$

Key Extraction, $\mathcal{E}(\mathtt{id}, \mathtt{msk})$: Select $r \in_R \mathbb{Z}_p$ and set $R := g^r$. Return $\mathtt{usk} := (y, R) \in \mathbb{Z}_p \times \mathbb{G}$ as the user secret key, where

$$y := r + zc \quad \text{and} \quad c := \mathrm{H}(R, \mathtt{id}).$$

Signing, $\mathcal{S}(\mathtt{id}, m, \mathtt{usk})$: Let $\mathtt{usk} = (y, R)$ and $c = \mathrm{H}(R, \mathtt{id})$. Select $a \in_R \mathbb{Z}_p$ and set $A := g^a$. Return $\sigma := (b, R, A) \in \mathbb{G} \times \mathbb{Z}_p \times \mathbb{G}$ as the signature, where

$$b := a + yd \quad \text{and} \quad d := \mathrm{G}(\mathtt{id}, A, m).$$

Verification, $\mathcal{V}(\sigma, \mathtt{id}, m, \mathtt{mpk})$: Let $\sigma = (b, R, A)$, $c := \mathrm{H}(R, \mathtt{id})$ and $d := \mathrm{G}(\mathtt{id}, A, m)$. The signature is valid if

$$g^b = A(R \cdot Z^c)^d.$$

Figure 4: The (Original) Galindo-Garcia IBS.

Remark 2. There are two hash functions in consideration: H and G. H is used to generate the user secret key which, in turn, is required to sign on a message (using G). Hence H < G (recall that < denotes 'followed by') constitutes the *logical* order for calling the hash functions.

2.4.2 Security Argument.

Henceforth, we refer to the security argument given in [CKK13]. The security is argued using three reductions $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ depending on two events E and F. Briefly, E partitions the type of adversaries into two, depending on whether it produces a forgery with the same randomiser; F, on the other hand, further partitions the type of adversaries into two depending on the order of the target[9] G and H indices: G < H or H < G. In all the three cases: DLP $\leq$ EU-ID-CMA [GG-IBS]. The differences are highlighted below.

Event	Reduction	Degradation	Forking Algorithm
E	$\mathcal{R}_1$	$\mathrm{O}\left(q_\varepsilon q_\mathrm{H}\right)$	$\mathcal{F}_\mathcal{W}$
$\neg\mathsf{E} \wedge \mathsf{F}$	$\mathcal{R}_2$	$\mathrm{O}\left((q_\mathrm{H} + q_\mathrm{G})^2\right)$	$\mathcal{M}_{\mathcal{W},1}$
$\neg\mathsf{E} \wedge \neg\mathsf{F}$	$\mathcal{R}_3$	$\mathrm{O}\left((q_\mathrm{H} + q_\mathrm{G})^6\right)$	$\mathcal{M}_{\mathcal{W},3}$

Table 2: Security Argument from [CKK13].

[9]The random oracle query that is used by $\mathcal{A}$ to produce its desired output is termed the target query and the index of this query is the target index.

3 Our Result

In this section, we describe means by which one can give a better bound for the reduction $\mathcal{R}_3$. But first, we try to pin point the source of degradation for $\mathcal{R}_3$ and in general for the nested replay attack with three forking.

3.1 Degradation: A Closer Look

Let's return to the primary source of degradation for GG-IBS: reduction $\mathcal{R}_3$. For $i = \{0, \ldots, 3\}$, let J_i [resp. I_i] denote the target H-index [resp. G-index] for round i of simulation. (see **Figure 5**). The condition for the success of the forkings can be derived from (14) to be

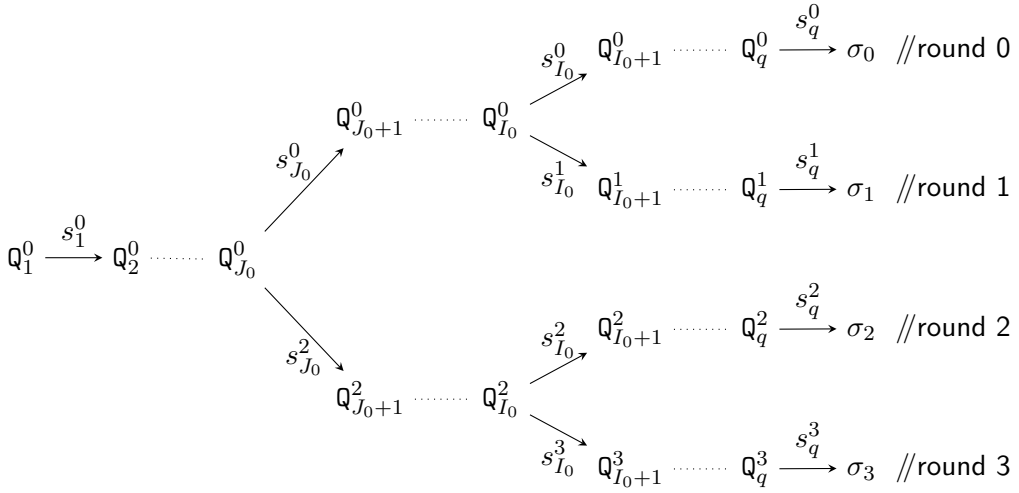

Figure 5: Successful Multiple-Forking for $\mathcal{R}_3$. $\mathtt{Q}_{J_0}^0$ denotes the target H-query $\mathrm{H}(\hat{R}, \hat{\mathtt{id}})$ for round 0; $\mathtt{Q}_{I_0}^0$ [resp. $\mathtt{Q}_{I_0}^2$] denotes the target G-query $\mathrm{G}(\hat{\mathtt{id}}, \hat{A}_0, \hat{m})$ [resp. $\mathrm{G}(\hat{\mathtt{id}}, \hat{A}_1, \hat{m})$] for round 0 [resp. round 2].

$\mathsf{E} : \mathsf{B} \wedge \mathsf{C}_0 \wedge \mathsf{C}_2 \wedge \mathsf{D}_2$, with the *core* condition of

$$(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0). \tag{1}$$

Each of the equality checks, loosely speaking, contributes to a factor of $\mathrm{O}\left(q^2\right)$ leading to the overall degradation of $\mathrm{O}\left(q^6\right)$. Our objective of bringing down the overall degradation relies on relaxing the core condition. This, in turn, is owed to the following observations.

Observation 1 (Independency of I_2 and I_0)**.** *It is not necessary for the target indices of the H oracle to be the same in* round 0 *and* round 2 *(i.e., I_2 needn't match I_0).*

In order to see this, let's return to the system of congruences involved in the simulation, *i.e.*,

$$\{\hat{b}_0 = \hat{a}_0 + (\hat{r} + \alpha c_0)d_0, \hat{b}_1 = \hat{a}_0 + (\hat{r} + \alpha c_0)d_1, \hat{b}_2 = \hat{a}_2 + (\hat{r} + \alpha c_2)d_2, \hat{b}_3 = \hat{a}_2 + (\hat{r} + \alpha c_2)d_3\} \tag{2}$$

What we have is a system of four congruences in the four unknowns $\{\hat{r}, \hat{a}_0, \hat{a}_2, \alpha\}$ with α being the solution to the DLP. From the structure of the hash function H, it follows that the index I corresponds to the unknown $\hat{a}$. The process of solving for α starts by eliminating the unknown $\hat{a}_0$ [resp. $\hat{a}_2$] from $\{\hat{b}_0, \hat{b}_1\}$ [resp. $\{\hat{b}_2, \hat{b}_3\}$]. What is *necessary* at this point is that the I indices must match for round 0 and round 1 [resp. round 2 and round 3]. However, eliminating $\hat{a}_0$ from $\{\hat{b}_0, \hat{b}_1\}$ is not affected by the pair $\{\hat{b}_2, \hat{b}_3\}$ and vice versa. Hence, from the point of view of the reduction, it doesn't make any difference whether we relax the condition to accommodate independency–the system of congruences one ends up with is *exactly* the same as in (2). In fact,

the reduction is unlikely to achieve anything by restricting the indices. Thus, incorporating independency simplifies the condition in (1) to

$$(I_3, J_3) = (I_2, J_2) \wedge (I_1, J_1) = (I_0, J_0) \wedge (J_2 = J_0). \tag{3}$$

Remark 3. The notion of independency can be better appreciated if we visualise the process of multiple-forking in terms of congruences and unknowns. At a high level, what the reduction algorithm $\mathcal{R}_3$ secures from the MF Algorithm is a set of four congruences in four unknowns. One of these unknowns is the solution to the DLP. The MF Algorithm needs to ensure that the congruences are linearly independent of each other with a certain non-negligible probability. **Observation 1** can then be restated as: even if the condition on the I indices is relaxed, we *still* end up with a system of four congruences in four unknowns.

Observation 2 (Dependency between H and G)**.** *It is possible to modify GG-IBS, the structure of the hash function G to be precise, such that $I_1 = I_0$ implies $J_1 = J_0$ (and similarly $I_3 = I_2$ implies $J_3 = J_2$) with a non-negligible probability.*

Recall that the hash functions in the construction of GG-IBS are of the form $\mathrm{H}(\mathtt{id}, R)$ and $\mathrm{G}(\mathtt{id}, A, m)$. Suppose that we modify the structure of G to $\mathrm{G}(\mathtt{id}, A, m, c)$ where $c := \mathrm{H}(\mathtt{id}, R)$. Let's consider round 1 of simulation for the "modified" GG-IBS initiated by a forking at I_0 as in reduction $\mathcal{R}_3$. Suppose that the adversary is successful and, in addition, the target H-index for

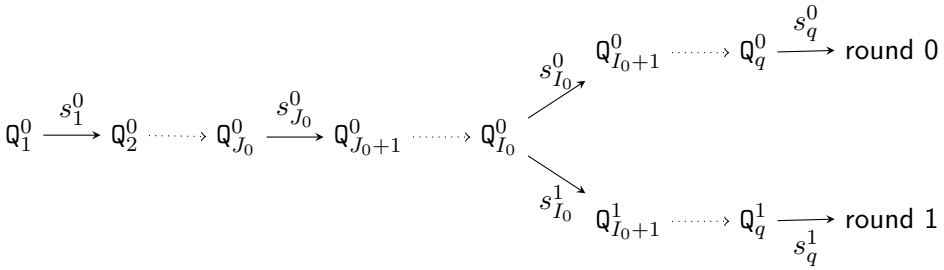

Figure 6: Oracle replay attack after setting up dependency between the hash functions. $\mathtt{Q}_{J_0}^0$ denotes the target H-query $\mathrm{H}(\hat{x})$ while $\mathtt{Q}_{I_0}^0$ corresponds to the target $\mathrm{G}(\hat{y}, \hat{c})$ where $\hat{c} = \mathrm{H}(\hat{x})$.

the round 1 matches with that in round 0 (*i.e.*, $I_1 = I_0$). We claim: due to the *binding* between the hash functions that we have introduced, the target G-indices for the two rounds also *have to* match. The advantage with which an adversary can forge a signature having violated this condition is, in fact, negligible (see **Claim 1**). Hence, $(I_1 = I_0)$ *implies* $(J_1 = J_0)$. We say that the random oracle G is "dependent" on the random oracle H (denoted by $\mathrm{H} \prec \mathrm{G}$) over these two rounds[10]. The property holds for round 2 and round 3 as well (*i.e.* $(I_3 = I_2)$ implies $(J_3 = J_2)$). As a result, the condition in (3) is further simplified to

$$(I_3 = I_2) \wedge (I_1 = I_0) \wedge (J_2 = J_0). \tag{4}$$

The introduction of binding, in fact, brings more to the table. In a particular round of simulation, to forge a signature, an adversary (except with a negligible probability of guessing) *has* to make the target random oracle queries in the order $\mathrm{H} < \mathrm{G}$ (see **Claim 1**). For example, it follows in round 0 that $(1 \leq J_0 < I_0 \leq q)$. Thus, as a result of the binding, the logical order is *imposed* on the adversary. That brings us to the formal definition of the notion of random-oracle dependency.

[10]Intuitively, if a protocol uses two hash functions H_1 and H_2 in such a way that the input to H_2 is a *function* of the output of H_1, then we say that the H_2-call is *dependent* on the H_1-call. This is possible only when there is an inherent order, a *hierarchy*, among the hash functions used in the construction of the protocol.

Definition 6 (Random-Oracle Dependency)**.** Consider the oracle replay attack in the context of a cryptographic protocol that employs two hash functions H_1 and H_2 modelled as random oracles. Let J [resp. I] denote the target H_1-index [resp. H_2-index] for the first round of simulation of the protocol. Also, let J' [resp. I'] denote the target H_1-index [resp. H_2-index] for the second round of simulation that was initiated by a forking at I. Suppose that the adversary was successful in both the rounds. The random oracle H_2 is defined to be η-dependent on the random oracle H_1 on the target query (denoted by $\mathrm{H}_1 \prec \mathrm{H}_2$) if the following criteria are satisfied: *i*) $(1 \leq J < I \leq q)$ or, in other words $\mathrm{H}_1 < \mathrm{H}_2$; and *ii*) $\Pr[(J' \neq J) \mid (I' = I)] \leq \eta$.

The second criterion, in other words, requires $I' = I \implies J' = J$ with probability at least $(1 - \eta)$; for the criterion to hold with overwhelming probability, η should be negligible in κ. A little more specifically, H_2 is said to be *fully-dependent* on H_1 if $\eta = 0$, *i.e.* $(I' = I) \implies (J' = J)$. But for most applications, it suffices that J be η-dependent on I for some η which is negligible.

Although dependency forces an order among the hash functions, the reverse may not be true–a logical order among the hash functions in the protocol does not necessarily translate into dependency between them. Hence, one may need to impose explicit dependency among the hash functions. A natural way to induce the dependency $\mathrm{H}_1 \prec \mathrm{H}_2$ is by making the input to H_2 a function of H_1's output (like the binding we introduced for GG-IBS).

Next, we formally argue that the binding in GG-IBS does indeed translate into a dependency between the random oracles (see **Figure 7** for a detailed construction).

Claim 1 (Dependency for modified GG-IBS)**.** *The random-oracle dependency of $H \prec G$ with $\eta := q(q-1)/p$ applies to the modified GG-IBS.*

Argument. Consider an adversary $\mathcal{A}$ against the modified GG-IBS. Let's suppose that it produces a valid, non-trivial forgery $\hat{\sigma} = (\hat{b}, \hat{R}, \hat{A})$ on $(\hat{\mathtt{id}}, \hat{m})$ at the end of one round of simulation. Let J [resp. I] denote the index of the target H-query $\mathrm{H}(\hat{\mathtt{id}}, \hat{R})$ [resp. G-query $\mathrm{G}(\hat{m}, \hat{A}, c)$, where $c := \mathrm{H}(\hat{\mathtt{id}}, \hat{R})$] for the round. It can be established[11] that $0 < I, J \leq q$. Now, there are two possibilities: $\mathrm{H} < \mathrm{G}$ and $\mathrm{G} < \mathrm{H}$. Let's look at the second of the cases. Since G takes c as an input, $\mathrm{G} < \mathrm{H}$ is plausible only if $\mathcal{A}$ correctly *anticipates* the value of c. Thus, for $\mathcal{A}$ to have a non-negligible advantage, it has to make the two target queries *and* in the order $\mathrm{H} < \mathrm{G}$. Simply put, the random oracles satisfy the first criterion.

Next, we simulate $\mathcal{A}$ again after forking at I. Let (I', J') be the target indices for this round. In order to establish that the random oracles satisfy the second criterion, let's assume the contrary: $I' = I$ *but* $J' \neq J$. Since $I' = I$, $\mathcal{A}$ has to forge a signature $\hat{\sigma}' = (\hat{b}', \hat{R}', \hat{A})$ on $(\hat{\mathtt{id}}', \hat{m})$ such that $\mathrm{H}(\hat{\mathtt{id}}', \hat{R}') = c$. This, in turn, is tantamount to a collision of the random function corresponding to H and, as per the birthday bound, the probability of this event is $q(q-1)/p$. Moreover, provided that $q \ll p$ the value is negligible[12]. By implication, the second criterion is also satisfied. □

More generally:

[11] Let S denote the event that $\mathcal{A}$ successfully forges $\hat{\sigma} = (y, R)$ on some message $\hat{m}$. In addition, let H denote the event that $\mathcal{A}$ queries the random oracle with $\mathrm{H}(\hat{m}, R)$–the "target" H-oracle query. Thus, we have

$$\begin{aligned}\Pr[\mathsf{S}] &= \Pr[\mathsf{S} \wedge \mathsf{H}] + \Pr[\mathsf{S} \wedge \neg\mathsf{H}] \\ &= \Pr[\mathsf{S} \wedge \mathsf{H}] + \Pr[\mathsf{S} \mid \neg\mathsf{H}] \Pr[\neg\mathsf{H}].\end{aligned}$$

The term $\mathsf{S}|\neg\mathsf{H}$ indicates that $\mathcal{A}$ has forged successfully, but, without actually having made the target H-oracle query $\mathrm{H}(\hat{m}, R)$. However, the value of $\mathrm{H}(\hat{m}, R)$ is necessary in order to forge. Therefore, in this situation, the best that the adversary can do to conjure up with the value of $\mathrm{H}(\hat{m}, R)$ is to guess it. However, the probability with which the adversary could have *correctly* anticipated the output is negligible ($1/p$ to be precise). Thus the $\Pr[\mathsf{S} \mid \neg\mathsf{H}]$ term (weighed down by the $1/p$ factor) is negligible and as a result $\Pr[\mathsf{S}] \approx \Pr[\mathsf{S} \wedge \mathsf{H}]$. Simply put, $\mathcal{A}$ *has* to make the target random oracle query to be able to forge with a non-negligible probability.

[12] To be precise, q is polynomial in κ whereas $1/p$ is a negligible function in κ. Plus, the product of a polynomial and a non-negligible function is still non-negligible.

Claim 2 (Binding induces dependency)**.** *Consider the hash functions (and the corresponding random oracles) described in* **Definition 1**. *Let q_1 denote the upper bound on the number of queries to the random oracle H_1. In addition, let $\mathbb{R}_1$ denote the range of H_1. Binding H_2 to H_1 (by making the input to H_2 a function of H_1's output) induces a random-oracle dependency $H_1 \prec H_2$ with $\eta_b := q_1(q_1 - 1)/|\mathbb{R}_1|$.*

Argument. The line of argument is the same as in **Claim 1**. □

Remark 4. Assuming q_1 to be polynomial and $|\mathbb{R}_1|$ to be exponential in κ, the value of η_b is asymptotically negligible. Now, let's consider a concrete setting, say 80-bit security level. Assuming typical values of $q_1 := 2^{60}$ and $|\mathbb{R}_1| := 2^{80}$, the value can no longer be ignored. However, if we take into consideration the degradation in **Theorem 1**, we end up choosing $|\mathbb{R}_1| := 2^{260}$ rendering η_b negligible. Hence, we can safely assume full-dependency in subsequent discussions.

The consequences of (in)dependency. In a nutshell, our observations affect the security of GG-IBS in the following ways.

(i) We saw that, once (in)dependency is taken into consideration, the condition for success of the multiple forkings is simplified to (4). Intuitively, this would bring down the degradation to $\mathrm{O}\left(q^3\right)$.

(ii) The incompleteness of $\mathcal{B}_2$ stems from the fact that the simulator cannot really restrict the order in which the adversary makes the two target random oracle queries. In [CKK13], the issue was fixed using the *two-reduction* strategy. By imposing the logical order $\mathrm{H} < \mathrm{G}$ through dependency, we can do away with the events F and $\neg\mathsf{F}$ and hence the reduction $\mathcal{R}_2$. This leads to a simpler event structure.

As we will see in the next section, we end up with a cleaner, tighter security argument for GG-IBS.

3.2 Galindo-Garcia IBS, Improved

We begin by describing, in detail, the GG-IBS scheme in which the binding required to induce the dependency $\mathrm{H} \prec \mathrm{G}$ has been set up.[13] The construction is same as in **Figure 4** *save* for the structure of the hash function G, which is now a function of $c := \mathrm{H}(\mathtt{id}, R)$ (boxed in **Figure 7**). The binding that we have introduced is more refined[14] than the one suggested in **Observation 2**.

Set-up, $\mathcal{G}(1^\kappa)$: Generate a group $\mathbb{G} = \langle g \rangle$ of prime order p. Select $z \in_R \mathbb{Z}_p$ and set $Z = g^z$. Return z as the master secret key `msk` and $(\mathbb{G}, p, g, Z, \mathrm{H}, \mathrm{G})$ as the master public key `mpk`, where H and G are hash functions

$$\mathrm{H} : \{0,1\}^* \times \mathbb{G} \mapsto \mathbb{Z}_p \quad \text{and} \quad \mathrm{G} : \{0,1\}^* \times \mathbb{G} \times \mathbb{Z}_p \mapsto \mathbb{Z}_p.$$

[13] A noteworthy observation is that, setting up the random-oracle dependency $\mathrm{G} \prec \mathrm{H}$ in GG-IBS allows more efficient reductions to DLP. However, this disturbs the "logical" order of the random oracles from the protocol's perspective. In such a protocol, the PKG will have to issue private keys for each message to be signed, rendering it impractical.

[14] Recall that the suggestion in **Observation 2** was to set $d := \mathrm{G}(\mathtt{id}, A, m, c)$ where $c := \mathrm{H}(R, \mathtt{id})$. However, `id` is redundant here as it is anyways captured indirectly by c.

Key Extraction, $\mathcal{E}(\mathtt{id}, \mathtt{msk})$: Select $r \in_R \mathbb{Z}_p$ and set $R := g^r$. Return $\mathtt{usk} := (y, R) \in \mathbb{Z}_p \times \mathbb{G}$ as the user secret key, where

$$y := r + zc \quad \text{and} \quad c := \mathrm{H}(\mathtt{id}, R).$$

Signing, $\mathcal{S}(\mathtt{id}, m, \mathtt{usk})$: Let $\mathtt{usk} = (y, R)$ and $c = \mathrm{H}(\mathtt{id}, R)$. Select $a \in_R \mathbb{Z}_p$ and set $A := g^a$. Return $\sigma := (b, R, A) \in \mathbb{G} \times \mathbb{Z}_p \times \mathbb{G}$ as the signature, where

$$b := a + yd \quad \text{and} \quad \boxed{d := \mathrm{G}(m, A, c)}.$$

Verification, $\mathcal{V}(\sigma, \mathtt{id}, m, \mathtt{mpk})$: Let $\sigma = (b, R, A)$, $c := \mathrm{H}(\mathtt{id}, R)$ and $\boxed{d := \mathrm{G}(m, A, c)}$. The signature is valid if

$$g^b = A(R \cdot Z^c)^d.$$

Figure 7: Galindo-Garcia IBS with binding.

3.2.1 Security Argument

Once the binding is in place, the adversary is bound to make the target queries in the logical order (except with a negligible probability, see **Claim 1**). This simplifies the event structure of the security argument to quite an extent and, accordingly, it consists of only two reductions $\mathcal{R}_1'$ and $\mathcal{R}_3'$.

Theorem 1. *Let $\mathcal{A}$ be an $(\epsilon, q_\varepsilon, q_H, q_G)$-adversary against GG-IBS in the* EU-ID-CMA *model. If the hash functions H and G are modelled as random oracles, we can construct either*

(i) Algorithm $\mathcal{R}_1'$ that $\epsilon^2/(2\exp(1)q_G q_\varepsilon)$-breaks the DLP, or

(ii) Algorithm $\mathcal{R}_3'$ that $\epsilon^4/64(q_H + q_G)^3$-breaks the DLP.

Here q_ε denotes the upper bound on the number of extract queries that $\mathcal{A}$ can make; q_H [resp. q_G] denotes the upper bound on the number of queries to the H-oracle [resp. G-oracle]. exp *is the base of natural logarithm. (Plus, for simplicity, we have assumed both $1/p$ and $q(q-1)/p$ to be negligible).*

Argument. Recall the events E and $\neg\mathsf{E}$ that were defined in the security argument of GG-IBS. In the case of the event E [resp. $\neg\mathsf{E}$] we give a reduction $\mathcal{R}_1'$ [resp. $\mathcal{R}_3'$] to the DLP. $\mathcal{R}_1'$ can be regarded to be a simplified version of the reduction $\mathcal{R}_1$ confined to handling the order $\mathrm{H} < \mathrm{G}$ (recall that $\mathcal{R}_1$ is equipped to handle both the orders). As the gist of the reduction $\mathcal{R}_1'$ is the same as in $\mathcal{R}_1$ given in [CKK13, §3], we relegate it to **Appendix B**. As for $\mathcal{R}_3'$, the simulation of one round of the adversary, as well as the manner in which it is forked is the same as plotted in $\mathcal{R}_3$. However, we use the Rewinding Technique ([PS00]) instead of the MF Algorithm, as shown in **Figure 8**. The novelty lies in exploiting (in)dependency for the fruition of a better security bound. Hence, we focus on the probability analysis. □

3.3 Analysis

On a high level, the analysis involves two iterations of the Splitting Lemma (**Lemma 1**) on different, but correlated, underlying sets. Consequently, we have two base notions of "good": good and $\mathsf{good}_{(1)}$. In addition, we have a higher notion of $\mathsf{good}_{(3)}$ which we, ultimately, intend to bound.

Conventions. $\mathbb{T}$ denotes the universe of random tapes participating in a single round of simulation of the adversary. This includes the internal coins of the adversary as well as the randomness from the random functions associated to the two random oracles, the signature oracle and the extract oracle. $\mathbb{T}_{(1)}$ denotes the universe of random tapes involved in two rounds of simulation resulting from the forking of the G-oracle. Thus, $\mathbb{T}_{(1)}$ is defined as

$$\mathbb{T}_{(1)} = \bigcup_{i=2}^{q} \mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)}. \tag{5}$$

Here, $\mathbb{T}_{(i-)}$ denotes the set of all random tapes involved in the simulation before the adversary makes the query $\mathtt{Q}^0_i$, whereas $\mathbb{T}_{(i+)}$, those after the query $\mathtt{Q}^0_i$.[15] Note that $\mathbb{T}_{(1)}$ consists of the partitions $(\mathbb{T}_{(2-)} \times \mathbb{T}^2_{(2+)})$, $(\mathbb{T}_{(3-)} \times \mathbb{T}^2_{(3+)})$ and $(\mathbb{T}_{(q-)} \times \mathbb{T}^2_{(q+)})$. We use the shorthand $\mathbb{T}_{(1,j<)}$ [resp. $\mathbb{T}_{(1,j>)}$] to denote the union $\cup^{j}_{i=2}(\mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)})$ [resp. $\cup^{q}_{i=j+1}(\mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)})$]. Thus, $\mathbb{T}_{(1)} = \mathbb{T}_{(1,j<)} \cup \mathbb{T}_{(1,j>)}$. Finally, $\mathbb{T}_{(3)}$ denotes the universe of random tapes for all four rounds of simulation[16] with the adversary forked as per the illustration in **Figure 8**. We follow the same subscripting convention for individual tapes as well with the round is indicated in the superscript; *e.g.*, $\mathtt{T}^0_{(i-)}$ denotes the random tape involved in the first round of simulation up till the adversary makes the query $\mathtt{Q}^0_i$ (see **Figure 8**).

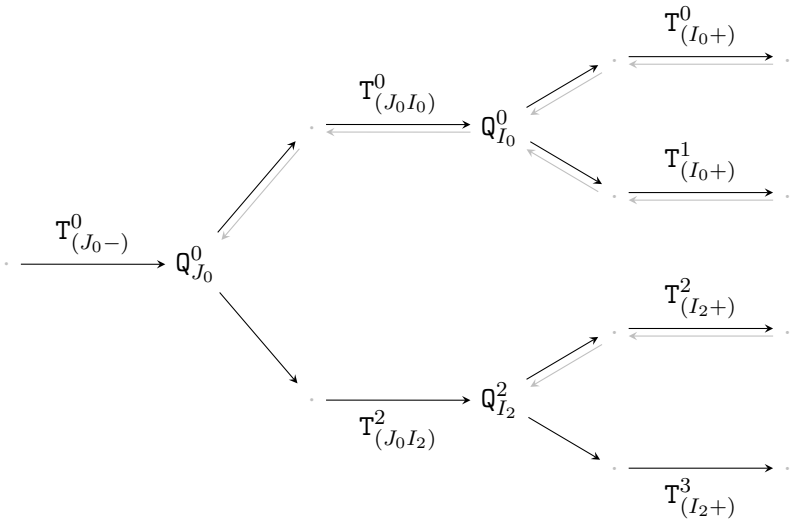

Figure 8: The tapes involved in the nested replay attack involving *three* forkings (carried out using the Rewinding Technique).

Defining the notions of "good". A tape in set $\mathbb{T}$ is deemed to be good if it leads to the adversary successfully forging. Thus, by definition, at least ϵ fraction of the tapes in $\mathbb{T}$ are

[15] Keep in mind that the set $\mathbb{T}_{(i-)}$ can be further represented by the Cartesian product of the two sets $\mathbb{T}_{(j-)}$ and $\mathbb{T}_{(ji)}$ as shown below.

$$\mathbb{T}_{(1)} = \bigcup_{i=2}^{q} \left(\mathbb{T}_{(j-)} \times \mathbb{T}_{(ji)}\right) \times \mathbb{T}^2_{(i+)}$$

Here $\mathbb{T}_{(j-)}$ denotes the set of all random tapes involved in the simulation before the adversary makes the query $\mathtt{Q}^0_j$, whereas $\mathbb{T}_{(ji)}$ denotes those from the query $\mathtt{Q}^0_j$ to the query $\mathtt{Q}^0_i$. In fact, this representation is in agreement with the **Figure 8** more than the one in (5).

[16] It can be worked out that the universe of tapes for three forkings is

$$\mathbb{T}_{(3)} = \bigcup_{j=1}^{q-1} \left(\mathbb{T}_{(j-)} \times \left(\bigcup_{i=j+1}^{q} \left(\mathbb{T}_{(ji)} \times \mathbb{T}^2_{(i+)}\right) \right)^2 \right).$$

.

good. However, the refined notion for good is a bit different as there are *two* random oracles in consideration. A tape in set $\mathbb{T}$ is deemed to be $\mathsf{good}_{(j,i)}$ if it leads to the adversary forging successfully and with a target H-index of j and a target G-index of i. In addition, $\mathsf{good}_{(j,*)}$ denotes a tape which leads to the adversary forging successfully with a target H-index i and *any* target G-index. In a analogous manner, we define the notion of $\mathsf{good}_{(*,i)}$.[17]

In the case of GG-IBS, there are two random oracles in consideration. Thus, the "higher" notion of $\mathsf{good}_{(1)}$ could be applied to tapes participating in replay of either of these oracles. But, as we are concerned with the replay of the G-oracle in the first two rounds of the nested replay attack (see **Figure 8**), the underlying set for which the notion of $\mathsf{good}_{(1)}$ applies is defined to be $\mathbb{T}_{(1)}$. A triplet of tape segments $(\mathtt{T}_{(I-)}, (\mathtt{T}_{(I+)}, \mathtt{T}'_{(I+)})) \in \mathbb{T}_{(1)}$ is considered to be $\mathsf{good}_{(1)}$ if the tapes $\mathtt{T} = (\mathtt{T}_{(I-)}, \mathtt{T}_{(I+)})$ and $\mathtt{T}' = (\mathtt{T}_{(I-)}, \mathtt{T}'_{(I+)})$ are both $\mathsf{good}_{(*,I)}$. To paraphrase, the tape is $\mathsf{good}_{(1)}$ if the replay of the G-oracle using tapes $\mathtt{T}$ and $\mathtt{T}'$ is successful. We define the refined notions of $\mathsf{good}_{(1,j,i)}$, $\mathsf{good}_{(1,*,i)}$ and $\mathsf{good}_{(1,j,*)}$ the same way we had defined it for the notion of good. Finally, a tape in $\mathbb{T}_{(3)}$ is $\mathsf{good}_{(3)}$ if the nested replay attack launched as in **Figure 8** using the tape is successful. Keeping these notions in mind, we proceed to the probability analysis of the nested replay attack.

Probability Analysis. Our objective is to lower bound the fraction of $\mathsf{good}_{(3)}$ tapes in $\mathbb{T}_{(3)}$ in terms of the fraction of good tapes in $\mathbb{T}$. This is accomplished through three claims: **Claim 3** through **Claim 5**, which, in turn, requires two iterations of the Splitting Lemma (**Lemma 1**). The objective of the first iteration of the Splitting Lemma (in **Claim 3**) is to establish a lower bound on the fraction of $\mathsf{good}_{(1)}$ tapes in $\mathbb{T}_{(1)}$, in terms of the fraction of good tapes in $\mathbb{T}$. This bound is, in turn, to be used in the second iteration (in **Claim 5**) to lower bound the fraction of $\mathsf{good}_{(3)}$ tapes in $\mathbb{T}_{(3)}$. However, the second iteration is far more involved than the first iteration–we need an intermediate **Claim 4**. Nevertheless, the usage of the Splitting Lemma, on a high level, is (self) similar to that in the first iteration; it's the underlying set, the associated probabilities and the notion of "good" that differs. Let's focus on the first two rounds of simulation of the adversary (see **Figure 8**).

Claim 3. *At least $\epsilon^2/4q$ fraction of the tapes in $\mathbb{T}_{(1)}$ are $\mathsf{good}_{(1)}$.*

Argument. The analysis, because of dependency, turns out to be similar to analysing the two rounds of the elementary replay attack *bar* the definition of the notion of "good"–we use the notion of $\mathsf{good}_{(*,i)}$ in place of $\mathsf{good}_{(i)}$. Let's presume that ϵ_i fraction of the tapes in $\mathbb{T}$ are $\mathsf{good}_{(*,i)}$. Thus, we have

$$\begin{aligned}\sum_{i=1}^{q} \epsilon_i &= \sum_{i=2}^{q} \Pr\left[I = i \wedge J > 0\right] \\ &= \Pr\left[1 \leq J < I \leq q\right] = \epsilon \ \text{ (by definition).}\end{aligned} \tag{6}$$

We assume that $\mathbb{T}$ is bijective to $\mathbb{T}_{(i-)} \times \mathbb{T}_{(i+)}$ through a "join" function (denoted by $\|$) and a "split" function.[18] The notion of $\mathsf{good}_{(*,i)}$, which we defined for the set $\mathbb{T}$, is adapted for the set $\mathbb{T}_{(i-)} \times \mathbb{T}_{(i+)}$ through the split and join functions . A tape $(\mathtt{T}_{(i-)}, \mathtt{T}_{(i+)}) \in \mathbb{T}_{(i-)} \times \mathbb{T}_{(i+)}$ is

[17]The notion of $\mathsf{good}_{(j,i)}$, in conjunction with the Three-Splitting Lemma, yields a (more) direct analysis. However, the resulting bound is alike the MF Lemma (**Lemma 3**). The key to an improved analysis requires circumventing the notion of $\mathsf{good}_{(j,i)}$ through the notion of $\mathsf{good}_{(1)}$ and, its refinements, $\mathsf{good}_{(1,j,*)}$ and $\mathsf{good}_{(1,*,i)}$. This, in turn, is made possible by our observations in **§3.1**–*i.e.*, through (in)dependency.

[18]The "join" function f_i concatenates the two constituent tape segments from the set $\mathbb{T}_{(i-)} \times \mathbb{T}_{(i+)}$ into a single tape in set $\mathbb{T}$, *i.e.*, $\mathrm{f}_i(\mathtt{T}_{(i-)}, \mathtt{T}_{(i+)}) := \mathtt{T}_{(i-)} \| \mathtt{T}_{(i+)}$. Provided that f_i is bijective, the inverse of f_i would be the function which "splits" a tape $\mathtt{T}$ into the two constituent tape segments $\mathtt{T}_{(i-)}$ and $\mathtt{T}_{(i+)}$ depending upon the random oracle query $\mathtt{Q}_i$, *i.e.*, $\mathrm{f}_i^{-1}(\mathtt{T}) := (\mathtt{T}_{(i-)}, \mathtt{T}_{(i+)})$. Note that the definition of f_i does not dependent of i and hence we consider a single join function f.

deemed to be $\mathsf{good}_{(*,i)}$ if its counterpart in $\mathbb{T}$ is $\mathsf{good}_{(*,i)}$. We denote the (sub)set of all such tapes in $\mathbb{T}_{(i-)} \times \mathbb{T}_{(i+)}$ by $\mathbb{V}_i$. Also, suppose that at least β_i fraction of tapes in $\mathbb{T}_{(i-)} \times \mathbb{T}_{(i+)}$ are $\mathsf{better}_{(*,i)}$. On these underlying assumptions, the analysis proceeds in a manner (almost) similar to that of the elementary replay attack (using the Rewinding Technique). Thus, we may conclude that the first two rounds of the nested replay attack are successful with a probability of at least $\epsilon^2/4q$. Moreover, it follows from the definition (of $\mathsf{good}_{(1)}$) that at least $\epsilon^2/4q$ fraction of tape segments in $\mathbb{T}_{(1)}$ are $\mathsf{good}_{(1)}$. □

The second iteration of the Splitting Lemma is a bit trickier than the first one. We have to prime the set $\mathbb{T}_{(1)}$ before actually applying the Splitting Lemma. The partitions of the set $\mathbb{T}_{(1)}$ come into play. Let's focus on a particular partition $(\mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)})$. It is not difficult to infer that the set of $\mathsf{good}_{(1,*,i)}$ tapes in $\mathbb{T}_{(1)}$, in fact, all belong[19] to the partition $(\mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)})$. These $\mathsf{good}_{(1,*,i)}$ elements can be further partitioned, depending on their target H-index, into subsets containing the $\mathsf{good}_{(1,j,i)}$ elements. The set of all such $\mathsf{good}_{(1,j,i)}$ elements, across the partitions (of $\mathbb{T}_{(1)}$), form the $\mathsf{good}_{(1,j,*)}$ elements of $\mathbb{T}_{(1)}$. Let δ_j denote the fraction of all such tapes in $\mathbb{T}_{(1)}$. By implication,

$$\begin{aligned}\sum_{j=1}^{q-1} \delta_j &= \sum_{j=1}^{q-1} \Pr\left[J_1 = J_0 = j \wedge I_1 = I_0 \wedge 1 < I_0 \leq q\right] \\ &= \Pr\left[J_1 = J_0 \wedge I_1 = I_0 \wedge 1 \leq J_0 < I_0 \leq q\right] \\ &\geq \epsilon^2/4q \ \ \text{(using } \textbf{Claim 3}\text{)} \end{aligned} \tag{7}$$

That brings us to our second claim.

Claim 4. *At least δ_j fraction of tapes in $\mathbb{T}_{(1,j>)}$ are $\mathsf{good}_{(1,j,*)}$.*

Argument. Let's presume that the tape involved in the first two rounds of simulation (the replay of G-oracle) was $\mathsf{good}_{(1,j,i)}$. Our argument relies on two observations: *i*) the candidate tapes for the replay of the H-oracle belong to $\mathbb{T}_{(1,j>)}$; and *ii*) a $\mathsf{good}_{(1,j,*)}$ element *cannot* belong to $\mathbb{T}_{(1,j<)}$. The first of the observations follows by definition, whereas the second–a consequence of dependency between the two random oracles–requires some explanation. Let's assume the contrary: a tape is $\mathsf{good}_{(1,j,*)}$ and it belongs to $\mathbb{T}_{(1,j<)}$. Consequently, it is $\mathsf{good}_{(1,j,i)}$ for some $i < j$. However, the dependency H $\prec$ G means that an adversary *has to* make the target queries in the order H < G which, in turn, implies $j < i$ (leading to a contradiction). Thus, the $\mathsf{good}_{(1,j,*)}$ elements all belong to $\mathbb{T}_{(1,j>)}$.

The two observations, in conjunction with the fact that $\mathbb{T}_{(1,j>)}$ is a subset of the set $\mathbb{T}_{(1)}$ (of which at least δ_j fraction are $\mathsf{good}_{(1,j,*)}$ elements) completes the proof. □

Claim 5. *The fraction of $\mathsf{good}_{(3)}$ tapes in $\mathbb{T}_{(3)}$ is at least $\epsilon^4/64q^3$.*

Argument. Let's assume that the tape involved in the first two rounds of simulation $\mathtt{T}^0_{(1)} := (\mathtt{T}^0_{(I_0-)}, (\mathtt{T}^0_{(I_0+)}, \mathtt{T}^1_{(I_0+)}))$ is $\mathsf{good}_{(1,J_0,I_0)}$. We further split the tape segment $\mathtt{T}^0_{(I_0-)}$ into two: $\mathtt{T}^0_{(J_0-)}$ and $\mathtt{T}^0_{(J_0I_0)}$. Here, $\mathtt{T}^0_{(J_0-)}$ is the tape involved in the simulation before the adversary makes the target query $\mathtt{Q}^0_{J_0}$, whereas $\mathtt{T}^0_{(J_0I_0)}$, that from the query $\mathtt{Q}^0_{J_0}$ up to the query $\mathtt{Q}^0_{I_0}$ (see **Figure 8**). Thus, an alternative representation of the tape $\mathtt{T}^0_{(1)}$ is

$$\begin{aligned}\mathtt{T}^0_{(1)} &:= \left(\mathtt{T}^0_{(J_0-)}, \mathtt{T}^0_{(J_0I_0)}, \left(\mathtt{T}^0_{(I_0+)}, \mathtt{T}^1_{(I_0+)}\right)\right) \\ &:= \left(\mathtt{T}^0_{(J_0-)}, \mathtt{T}^0_{(1,J_0I_0+)}\right) \ \ \text{(using shorthand notation)}\end{aligned}$$

[19] Strictly speaking, we have to take into consideration the non-core conditions.

The simulator, next, proceeds with the remaining two rounds of simulations by forking at $\mathtt{Q}^0_{J_0}$ and $\mathtt{Q}^2_{I_2}$ (in that order). Let $\mathtt{T}^2_{(1,J_0I_2+)} := (\mathtt{T}^2_{(J_0I_2)}, (\mathtt{T}^2_{(I_2+)}, \mathtt{T}^3_{(I_2+)})) \in \mathbb{T}_{(1,J_0I_2+)}$ denote the tapes involved in the aforementioned simulation. Thus, we end up with the tapes given in **Figure 8**. The nested replay attack, all four rounds of it, is successful if the tape

$$\mathtt{T}^2_{(1)} := \left(\mathtt{T}^0_{(J_0-)}, \left(\mathtt{T}^2_{(J_0I_2)}, \left(\mathtt{T}^2_{(I_2+)}, \mathtt{T}^3_{(I_2+)}\right)\right)\right)$$

turns out to be $\mathsf{good}_{(1,J_0,I_2)}$ (note that I_2 need not be the same as I_0). Our objective is to find the probability of this particular event using the Splitting Lemma. This requires $\mathbb{T}_{(1)}$ to be split into $\mathbb{T}_{(J_0-)}$ and $\cup^q_{i=J_0}(\mathbb{T}_{(J_0i)} \times \mathbb{T}^2_{(i+)})$. This, in turn, can be achieved by splitting the individual partitions of $\mathbb{T}_{(1)}$. But note that *not all* of the partitions in $\mathbb{T}_{(1)}$ can be split in such a way. In fact, by definition, it is viable only to the partitions $(\mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)})$ for $i > J_0$, *i.e.* the set $\mathbb{T}_{(1,J_0>)}$. At this point **Claim 4** comes into play. It allows us to apply the Splitting Lemma to the "reduced" universe of $\mathbb{T}_{(1,J_0>)}$ (by ensuring that at least δ_{J_0} fraction of the tapes in $\mathbb{T}_{(1,J_0>)}$ *too* are $\mathsf{good}_{(1,J_0,*)}$). We are now ready to apply the Splitting Lemma.

Let's assume that an arbitrary λ_j fraction of the tapes in $\mathbb{T}_{(1)}$ (and hence $\mathbb{T}_{(1,J_0>)}$) are $\mathsf{better}_{(1,j,*)}$. We denote the (sub)set of $\mathsf{good}_{(1,J_0,*)}$ elements and the $\mathsf{better}_{(1,J_0,*)}$ elements in $\mathbb{T}_{(1,J_0>)}$ by $\mathbb{U}_{J_0}$ and $\mathbb{U}^*_{J_0}$ respectively. For ease of analysis, we use the sufficient condition that the tape $\mathtt{T}^0_{(1)}$ be $\mathsf{good}_{(1,J_0,*)}$ and the $\mathtt{T}^2_{(1)}$ be $\mathsf{better}_{(1,J_0,*)}$. Let's denote the probability of this event (with the tapes sampled as described above) by δ'_{J_0}, *i.e.*,

$$\begin{aligned}
\delta'_{J_0} &= \Pr\left[\left(\mathtt{T}^0_{(J_0-)}, \mathtt{T}^0_{(1,J_0I_0+)}\right) \in \mathbb{U}^*_{J_0} \wedge \left(\mathtt{T}^0_{(J_0-)}, \mathtt{T}^2_{(1,J_0I_2+)}\right) \in \mathbb{U}_{J_0}\right] \\
&= \Pr\left[\left(\mathtt{T}^0_{(J_0-)}, \mathtt{T}^2_{(1,J_0I_2+)}\right) \in \mathbb{U}_{J_0} \mid \left(\mathtt{T}^0_{(J_0-)}, \mathtt{T}^0_{(1,J_0I_0+)}\right) \in \mathbb{U}^*_{J_0}\right] \cdot \Pr\left[\left(\mathtt{T}^0_{(J_0-)}, \mathtt{T}^0_{(1,J_0I_0+)}\right) \in \mathbb{U}^*_{J_0}\right] \\
&= (\delta_{J_0} - \lambda_{J_0})\lambda_{J_0} \quad \text{(using propositions 1 and 2 of \textbf{Lemma 1})} \qquad (8)
\end{aligned}$$

The above expression attains a maxima of $\delta^2_{J_0}/4$ at the point $\lambda_{J_0} = \delta_{J_0}/2$. Now, the probability that the nested oracle replay attack is successful for any J_0 is given by

$$\begin{aligned}
\delta' &= \sum_{J_0=1}^{q-1} \delta'_{J_0} \geq \sum_{J_0=1}^{q-1} \frac{\delta^2_{J_0}}{4} \\
&\geq \frac{\epsilon^4}{64q^3} \quad \text{(using Hölder's inequality and (7))} \qquad (9)
\end{aligned}$$

□

Significance of (in)dependency. Dependency plays a crucial role in the first two claims. It is the first criterion of **Definition 6** which allows us to apply the Splitting Lemma to a reduced universe of tapes in **Claim 4**. The second criterion of dependency comes into play in **Claim 3**. It ensures: if, indeed, the two tapes $(\mathtt{T}_{(I-)}, \mathtt{T}_{(I+)})$ and $(\mathtt{T}_{(I-)}, \mathtt{T}'_{(I+)})$ are both $\mathsf{good}_{(1,*,I)}$, then they have the same target H-index as well. In fact, this is the *sole* reason why the analysis proceeds as for the elementary replay attack. These two handles, together, reduce the degradation by $\mathrm{O}\left(q^2\right)$. Independency, on the other hand, is used in **Claim 5**. Recall that a $\mathsf{good}_{(1,*,i)}$ can only belong to the partition $(\mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)})$. Without independency, the set of candidate tapes (for replay of the H-oracle at a particular index j) is restricted to $(\mathbb{T}_{(i-)} \times \mathbb{T}^2_{(i+)})$. But, relaxing this condition (*i.e.*, incorporating independency) "expands" the set of candidate tapes to $\mathbb{T}_{(1,j>)}$. On a high level, this reduces the degradation by $\mathrm{O}\left(q\right)$. Apply the two observations together, the degradation is reduced by $\mathrm{O}\left(q^3\right)$.

3.4 Taking Stock

It is evident that the modified GG-IBS allows far tighter reductions. The effective degradation is reduced from $\mathrm{O}\left((q_{\mathrm{H}}+q_{\mathrm{G}})^6\right)$ to $\mathrm{O}\left((q_{\mathrm{H}}+q_{\mathrm{G}})^3\right)$ (see **Table 3**). The "gain", which is substantial, is by the virtue of the multiple forkings being replaced, on a high level, by (two) general forkings. This, in turn, is endowed by our observations in **§3.1**.

Scheme	Security Argument			
GG-IBS [CKK13]	Reduction	$\mathcal{R}_1$	$\mathcal{R}_2$	$\mathcal{R}_3$
	Forking	$\mathcal{F}_{\mathcal{W}}$	$\mathcal{M}_{\mathcal{W},1}$	$\mathcal{M}_{\mathcal{W},3}$
	Degradation	$\mathrm{O}\left(q_{\mathrm{G}}q_{\varepsilon}\right)$	$\mathrm{O}\left((q_{\mathrm{H}}+q_{\mathrm{G}})^2\right)$	$\mathrm{O}\left((q_{\mathrm{H}}+q_{\mathrm{G}})^6\right)$
Modified GG-IBS (**Figure 7**)	Reduction	$\mathcal{R}'_1$	$\mathcal{R}'_3$	
	Forking	$\mathcal{F}_{\mathcal{W}}$	Nested rewinding with (in)dependency	
	Degradation	$\mathrm{O}\left(q_{\mathrm{G}}q_{\varepsilon}\right)$	$\mathrm{O}\left((q_{\mathrm{H}}+q_{\mathrm{G}})^3\right)$	

Table 3: Degradation for the original and modified GG-IBS. q_{G} [resp. q_{H}] denotes the upper bound on the H-oracle [resp. G-oracle] queries, whereas, q_{ε} denotes upper bound on the extract queries.

The characteristic expression. The replay attack (involving one oracle) launched using Rewinding Technique is captured, essentially, by the expression:

$$\epsilon' \geq \frac{1}{4}\sum_{i=1}^{q}\epsilon_i^2 \tag{10}$$

under the set of constraints: *i)* $\sum_{i=1}^{q}\epsilon_i = \epsilon$, and *ii)* $(0 \leq \epsilon_i \leq 1)_{i\in\{1,\ldots,q\}}$. We say that (10) is the *characteristic* expression for the Rewinding Technique. Although the analysis of the GF Algorithm is handled in a different manner, the analyses are *au fond* simliar: the two approach are connected through the *characteristic* expression (10). On maximising (10) under the set of constraints given above, we end up with the bound given in Forking/GF Lemma.

The analysis in [PS00] can be, straightforwardly, extended to analyse the nested replay attack via rewinding (but) using a Three-Splitting Lemma[20]. For a general nested replay with n forkings, the resulting characteristic expression is:

$$\epsilon' \geq \sum_{j=1}^{q}\sum_{i=1}^{q}\epsilon_{ij}^{n+1} \tag{11}$$

under the set of constraints

$$\sum_{j=1}^{q}\sum_{i=1}^{q}\epsilon_{ij} = \epsilon \quad \text{and} \quad (0 \leq \epsilon_{ij} \leq 1)_{(i,j)\in\{1,\ldots,q\}^2}. \tag{12}$$

The expression attains a minima of ϵ^{n+1}/q^{2n} at the point $\epsilon_{ij} = \epsilon/q^2$ for $1 \leq i,j \leq q$, and hence the success probability of $\Omega(\epsilon^4/q^6)$. The characteristic expression for the analysis of MF Algorithm (using the random variable approach), again, agrees with (11)

[20] A generalisation of the Splitting Lemma that accommodates splitting the universe into *three* sets (see **§2.3.**). To be precise, one has to use an $(\epsilon_i, \epsilon_i/2, \epsilon_i/4)$-Three-Splitting Lemma.

By deploying (in)dependency, we have managed to come up with an optimal characteristic expression for nested replay attack (with three forkings). The second iteration of the Splitting Lemma, on a high level, seems to be similar to the first one. This is reflected in the characteristic expressions[21]

$$\delta' \geq \sum_{j=1}^{q-1} \delta_j^2 \quad \text{and} \quad \epsilon' \geq \sum_{i=2}^{q} \epsilon_i^2$$

under the set of constraints

$$\sum_{j=1}^{q-1} \delta_j = \epsilon' \ , \ \sum_{i=2}^{q} \epsilon_i = \epsilon \quad \text{and} \quad (0 \leq \epsilon_i, \delta_j \leq 1)_{i,j \in \{1,\ldots,q\}}.$$

Clearly, there is a degree of *self-similarity* in the two expressions and, in addition, the expressions themselves are quite disparate from that of the nested replay attack (and the MF algorithm for $n = 3$) in (11).

4 Conclusion

In this paper, we used the notion of (in)dependency to come up with a cleaner, tighter security argument for GG-IBS: the effective degradation is down from $\mathrm{O}\left(q^6\right)$ to $\mathrm{O}\left(q^3\right)$. The bound on security reductions for the Schnorr signature has been well-studied [PV05, GBL08, Seu12]. In the same vein, giving a bound on the reductions for GG-IBS scheme would be an interesting problem to pursue. In addition, studying the effect of (in)dependency on the MF Algorithm, and finding other applications for random-oracle dependency should also be worthwhile.

References

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 390–399, New York, NY, USA, 2006. ACM. (Cited on pages 2, 3, 22 and 23.)

[BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer Berlin / Heidelberg, 2004. (Cited on pages 2 and 6.)

[BPW12] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology*, 25:57–115, 2012. (Cited on pages 2, 4, 23 and 24.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, CCS '93, pages 62–73, New York, NY, USA, 1993. ACM. (Cited on page 4.)

[CC02] Jae Choon and Jung-Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In Yvo Desmedt, editor, *Public Key Cryptography — PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer Berlin / Heidelberg, 2002. (Cited on page 2.)

[21] In fact, it was the characteristic expression that guided us to the analysis (and not the other way around).

[CK13] Sanjit Chatterjee and Chethan Kamath. A closer look at multiple-forking: Leveraging (in)dependence for a tighter bound. Cryptology ePrint Archive, Report 2013/651, 2013. `http://eprint.iacr.org/`. (Cited on page 2.)

[CKK13] Sanjit Chatterjee, Chethan Kamath, and Vikas Kumar. Galindo-Garcia identity-based signature revisited. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology - ICISC 2012*, volume 7839 of *Lecture Notes in Computer Science*, pages 456–471. Springer Berlin / Heidelberg, 2013. Full version available in Cryptology ePrint Archive, Report 2012/646, http://eprint.iacr.org/2012/646. (Cited on pages 2, 3, 4, 8, 9, 13, 14 and 19.)

[CMW12] Sherman S. M. Chow, Changshe Ma, and Jian Weng. Zero-knowledge argument for simultaneous discrete logarithms. *Algorithmica*, 64(2):246–266, 2012. (Cited on page 4.)

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Berlin Heidelberg, 1985. (Cited on page 2.)

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Berlin / Heidelberg, 1987. (Cited on page 2.)

[GBL08] Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures. In *Proceedings of the 28th Annual conference on Cryptology: Advances in Cryptology*, CRYPTO 2008, pages 93–107, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on page 20.)

[GG09] David Galindo and Flavio Garcia. A Schnorr-like lightweight identity-based signature scheme. In Bart Preneel, editor, *Progress in Cryptology – AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 135–148. Springer Berlin / Heidelberg, 2009. (Cited on pages 2, 3, 4 and 7.)

[GQ90] Louis Guillou and Jean-Jacques Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO' 88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer Berlin / Heidelberg, 1990. (Cited on page 2.)

[Her05] Javier Herranz. Deterministic identity-based signatures for partial aggregation. *The Computer Journal*, 49(3):322–330, 2005. (Cited on page 2.)

[Hes03] Florian Hess. Efficient identity based signature schemes based on pairings. In Kaisa Nyberg and Howard Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer Berlin / Heidelberg, 2003. (Cited on page 2.)

[Kia07] Aggelos Kiayias. Cryptography: Primitives and protocols, 2007. (Cited on page 8.)

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In ErnestF. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer Berlin Heidelberg, 1993. (Cited on page 2.)

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000. (Cited on pages 2, 3, 5, 6, 8, 14 and 19.)

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin Heidelberg, 2005. (Cited on page 20.)

[RS11] V. Radhakishan and S. Selvakumar. Prevention of man-in-the-middle attacks using id-based signatures. In *Second International Conference on Networking and Distributed Computing - ICNDC, 2011*, pages 165 –169. 2011. (Cited on page 2.)

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991. 10.1007/BF00196725. (Cited on pages 2 and 8.)

[Seu12] Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 554–571. Springer Berlin / Heidelberg, 2012. (Cited on page 20.)

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin / Heidelberg, 1985. (Cited on page 2.)

[XW12] Min Xie and Libin Wang. One-round identity-based key exchange with perfect forward security. *Information Processing Letters*, 112(14–15):587 – 591, 2012. (Cited on page 2.)

A Forking Algorithms

A.1 General Forking

We reproduce the GF Algorithm from [BN06] followed by the statement of the GF lemma. We use slightly different notations to maintain uniformity.

Forking Algorithm. Fix $q \in \mathbb{Z}^+$ and a set $\mathbb{S}$ such that $|\mathbb{S}| \geq 2$. Let $\mathcal{W}$ be a randomised algorithm that on input a string x and elements $s_1, \ldots, s_q \in \mathbb{S}$ returns a pair (I, σ) consisting of an integer $0 \leq I \leq q$ and a string σ. The forking algorithm $\mathcal{F}_{\mathcal{W}}$ associated to $\mathcal{W}$ is defined as **Algorithm 1** below.

Algorithm 1 $\mathcal{F}_{\mathcal{W}}(x)$

Pick coins ρ for $\mathcal{W}$ at random

$\{s_1^0, \ldots, s_q^0\} \in_R \mathbb{S}$; $(I_0, \sigma_0) \leftarrow \mathcal{W}(x, s_1^0, \ldots, s_q^0; \rho)$ //round 0
if $(I_0 = 0)$ **then return** $(0, \perp, \perp)$

$\{s_{I_0}^1, \ldots, s_q^1\} \in_R \mathbb{S}$; $(I_1, \sigma_1) \leftarrow \mathcal{W}(x, s_1^0, \ldots, s_{I_0-1}^0, s_{I_0}^1, \ldots, s_q^1; \rho)$ //round 1
if $(I_1 = I_0 \wedge s_{I_0}^1 \neq s_{I_0}^0)$ **then return** $(1, \sigma_0, \sigma_1)$
else return $(0, \perp, \perp)$

Lemma 2 (General Forking Lemma [BN06]). *Let $\mathcal{G}_I$ be a randomised algorithm that takes no input and returns a string. Let*

$$gfrk := \Pr\left[(\mathsf{b}=1) \mid x \xleftarrow{\$} \mathcal{G}_I; (\mathsf{b},\sigma,\sigma) \xleftarrow{\$} \mathcal{F}_{\mathcal{W}}(x)\right] \quad \textit{and}$$
$$acc := \Pr\left[I \geq 1 \mid x \xleftarrow{\$} \mathcal{G}_I; \{s_1,\ldots,s_q\} \in_R \mathbb{S}; (I,\sigma) \xleftarrow{\$} \mathcal{W}(x,s_1,\ldots,s_q)\right],$$

then

$$gfrk \geq acc \cdot \left(\frac{acc}{q} - \frac{1}{|\mathbb{S}|}\right). \tag{13}$$

A.2 Multiple-Forking Algorithm

We describe the Multiple-Forking Algorithm [BPW12] with some notational changes. The success condition and the lemma that governs the success probability follows the algorithm.

The Multiple-Forking Algorithm Fix $q \in \mathbb{Z}^+$ and a set $\mathbb{S}$ such that $|\mathbb{S}| \geq 2$. Let $\mathcal{W}$ be a randomised algorithm that on input a string x and elements $s_1,\ldots,s_q \in \mathbb{S}$ returns a triple (I,J,σ) consisting of two integers $0 \leq J < I \leq q$ and a string σ. Let $n \geq 1$ be an odd integer. The MF Algorithm $\mathcal{M}_{\mathcal{W},n}$ associated to $\mathcal{W}$ and n is defined as **Algorithm 2** below.

Algorithm 2 $\mathcal{M}_{\mathcal{W},n}(x)$

Pick coins ρ for $\mathcal{W}$ at random

$\{s_1^0,\ldots,s_q^0\} \in_R \mathbb{S}$;
$(I_0,J_0,\sigma_0) \leftarrow \mathcal{W}(x,s_1^0,\ldots,s_q^0;\rho)$ //round 0
if $((I_0=0) \vee (J_0=0))$ **then return** $(0,\perp)$ //Condition $\neg\mathsf{B}$

$\{s_{I_0}^1,\ldots,s_q^1\} \in_R \mathbb{S}$;
$(I_1,J_1,\sigma_1) \leftarrow \mathcal{W}(x,s_1^0,\ldots,s_{I_0-1}^0,s_{I_0}^1,\ldots,s_q^1;\rho)$ //round 1
if $\left((I_1,J_1) \neq (I_0,J_0) \vee (s_{I_0}^1 = s_{I_0}^0)\right)$ **then return** $(0,\perp)$ //Condition $\neg\mathsf{C}_0$

$k := 2$
while $(k < n)$ **do**
  $\{s_{J_0}^k,\ldots,s_q^k\} \in_R \mathbb{S}$;
  $(I_k,J_k,\sigma_k) \leftarrow \mathcal{W}(x,s_1^0,\ldots,s_{J_0-1}^0,s_{J_0}^k,\ldots,s_q^k;\rho)$ //round k
  if $\left((I_k,J_k) \neq (I_0,J_0) \vee (s_{J_0}^k = s_{J_0}^{k-1})\right)$ **then return** $(0,\perp)$ //Condition $\neg\mathsf{D}_k$

  $\{s_{I_k}^{k+1},\ldots,s_q^{k+1}\} \in_R \mathbb{S}$;
  $(I_{k+1},J_{k+1},\sigma_{k+1}) \leftarrow \mathcal{W}(x,s_1^0,\ldots,s_{J_0-1}^0,s_{J_0}^k,\ldots,s_{I_k-1}^k,s_{I_k}^{k+1},\ldots,s_q^{k+1};\rho)$ //round k+1
  if $\left((I_{k+1},J_{k+1}) \neq (I_0,J_0) \vee \boxed{(s_{I_0}^{k+1} = s_{I_0}^k)}\right)$ **then return** $(0,\perp)$ //Condition $\neg\mathsf{C}_k$

  $k := k+2$
end while
return $(1,\{\sigma_0,\ldots,\sigma_n\})$

The success condition. The success of the MF Algorithm is determined by the set of conditions $\mathbb{A}_0 := \{\mathsf{B}, \mathsf{C}_0, \ldots, \mathsf{C}_{n-1}, \mathsf{C}_2, \ldots, \mathsf{D}_{n-1}\}$ where

$$\begin{aligned}
\mathsf{B} &: (I_0 \geq 1) \wedge (J_0 \geq 1) \\
\mathsf{C}_k &: (I_{k+1}, J_{k+1}) = (I_k, J_k) \wedge (s_{I_k}^{k+1} \neq s_{I_k}^{k}) \quad (\text{for } k = 0, 2, \ldots, n-1) \\
\mathsf{D}_k &: (I_k, J_k) = (I_0, J_0) \wedge (s_{J_0}^{k} \neq s_{J_0}^{k-1}) \quad (\text{for } k = 2, 4, \ldots, n-1)
\end{aligned} \tag{14}$$

To be precise, the MF Algorithm is successful in the event E that all of the conditions in $\mathbb{A}_0$ are satisfied, *i.e.*,

$$\mathsf{E} : \mathsf{B} \wedge \left(\mathsf{C}_0 \wedge \mathsf{C}_2 \wedge \cdots \wedge \mathsf{C}_{n-1}\right) \wedge \left(\mathsf{D}_2 \wedge \mathsf{D}_4 \wedge \cdots \wedge \mathsf{D}_{n-1}\right). \tag{15}$$

The probability of this event, which is denoted by *mfrk*, is bounded by the MF lemma given below.

Lemma 3 (The Multiple-Forking Lemma [BPW12])**.** *Let $\mathcal{G}_I$ be a randomised algorithm that takes no input and returns a string. Let*

$$\begin{aligned}
\mathit{mfrk} &:= \Pr\left[(\mathsf{b} = 1) \mid x \xleftarrow{\$} \mathcal{G}_I; (\mathsf{b}, \{\sigma_0, \ldots, \sigma_n\}) \xleftarrow{\$} \mathcal{M}_{\mathcal{W},n}(x)\right] \quad \textit{and} \\
\mathit{acc} &:= \Pr\left[(I \geq 1) \wedge (J \geq 1) \mid x \xleftarrow{\$} \mathcal{G}_I; \{s_1, \ldots, s_q\} \in_R \mathbb{S}; (I, J, \sigma) \xleftarrow{\$} \mathcal{W}(x, s_1, \ldots, s_q)\right]
\end{aligned}$$

then

$$\mathit{mfrk} \geq \mathit{acc} \cdot \left(\frac{\mathit{acc}^n}{q^{2n}} - \frac{n}{|\mathbb{S}|}\right). \tag{16}$$

B Reduction $\mathcal{R}_1'$

Let $\Delta := (\mathbb{G}, p, g, g^\alpha)$ be the given DLP instance. The reduction involves invoking the GF Algorithm on the wrapper $\mathcal{W}$ as shown in **Algorithm 3**. As a result, it obtains a set of two congruences in two unknowns and solves for α.

Algorithm 3 Reduction $\mathcal{R}_1'(\Delta)$

Select $z \in_R \mathbb{Z}_p^*$ as the $\mathtt{msk}$ and set $\mathtt{mpk} := (\mathbb{G}, g, p, g^z)$.
$(\mathsf{b}, \sigma_0, \sigma_1) \xleftarrow{\$} \mathcal{F}_{\mathcal{W}}((\mathtt{mpk}, \mathtt{msk}), g^\alpha)$
if $(\mathsf{b} = 0)$ **then return** $\perp$ $\quad$ //$\mathsf{abort}_{1,2}$
Parse σ_i as $(\hat{b}_i, c_i, r_i, \beta_i, d_i)$.
Let $\beta := \beta_0$, $c := c_0$ and $r := r_0$
if $\beta = 0$ **then return** $((\hat{b}_0 - \hat{b}_1) - zc(d_0 - d_1))/r(d_0 - d_1)$
else return $\perp$ $\quad$ //$\mathsf{abort}_{1,3}$
end if

The Wrapper

Suppose that $q := q_{\mathrm{G}}$ and $\mathbb{S} := \mathbb{Z}_p$. $\mathcal{W}$ takes as input the master keys ($\mathtt{mpk}$,$\mathtt{msk}$), the problem instance g^α and $s_1, \ldots, s_q$. It returns a pair (I, σ) where I is an integer that refers to the target G-query and σ is the side-output. In order to track the index of the current G-oracle query, $\mathcal{W}$ maintains a counter ℓ, initially set to 1. It also maintains a table $\mathfrak{L}_{\mathrm{H}}$ [resp. $\mathfrak{L}_{\mathrm{G}}$] to manage the random oracle H [resp. G]. $\mathcal{W}$ initiates the EU-ID-CMA game by passing $\mathtt{mpk}$ as the challenge master public key to the adversary $\mathcal{A}$. The queries by $\mathcal{A}$ are handled as per the following specifications.

(a) **Random oracle query,** $\mathrm{H}(\mathtt{id}, R)$: $\mathfrak{L}_{\mathrm{H}}$ contains tuples of the form

$$\langle \mathtt{id}, R, c, r, \beta \rangle \in \{0,1\}^* \times \mathbb{G} \times \mathbb{Z}_p \times \mathbb{Z}_p \cup \{\perp\} \times \{0, 1, \phi\}.$$

Here, $(\mathtt{id}, R)$ is the query to the H-oracle and c is the corresponding output. Therefore, a query $\mathrm{H}(\mathtt{id}, R)$ is fresh if there exists no tuple $\langle \mathtt{id}_i, R_i, c_i, r_i, \beta_i \rangle$ in $\mathfrak{L}_{\mathrm{H}}$ such that $(\mathtt{id}_i = \mathtt{id}) \wedge (R_i = R)$. If such a tuple exists, then the oracle has to return the corresponding c_i as the output. The role of the r and β-field is the same as in $\mathcal{R}_1$. We now explain how the fresh H-oracle queries are handled. The query may be

(i) H_1, Explicit query made by $\mathcal{A}$: In this case $\mathcal{W}$ returns $c \in_R \mathbb{Z}_p$ as the output. $\langle \mathtt{id}, R, c, \perp, \phi \rangle$ is added to $\mathfrak{L}_{\mathrm{H}}$.

(ii) H_2, Explicit query made by $\mathcal{W}$: As in the previous case, $\mathcal{W}$ returns $c \in_R \mathbb{Z}_p$ as the output. As $\mathcal{W}$ knows $r = \log_g R$, $\langle \mathtt{id}, R, c, r, 1 \rangle$ is added to $\mathfrak{L}_{\mathrm{H}}$.

(iii) H_3, Implicit query by $\mathcal{W}$ in order to answer a signature query made by $\mathcal{A}$: See step (iii) of **Signature query** on how to program the random oracle in this situation.

(b) **Random oracle query,** $\mathrm{G}(m, A, c)$: $\mathfrak{L}_{\mathrm{G}}$ contains tuples of the form

$$\langle m, A, c, d, \ell \rangle \in \{0,1\}^* \times \mathbb{G} \times \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}^+.$$

Here, (m, A, c) is the query to the G-oracle and d is the corresponding output. The index of the query is stored in the ℓ-field. Therefore, a random oracle query $\mathrm{G}(m, A, c)$ is fresh if there exists no tuple $\langle m_i, A_i, c_i, d_i, \ell_i \rangle$, in $\mathfrak{L}_{\mathrm{G}}$ such that $(m_i = m) \wedge (A_i = A) \wedge (c_i = c)$. If such a tuple exists, then the oracle has to return the corresponding d_i as the output. We now explain how the fresh G-oracle queries are handled. The query may be

(i) G_1, Explicit query made by either $\mathcal{A}$ or $\mathcal{W}$: In this case $\mathcal{W}$ returns $d := s_\ell$ as the output. $\langle m, A, c, d, \ell \rangle$ is added to $\mathfrak{L}_{\mathrm{G}}$ and ℓ is incremented by one.

(ii) G_2, Implicit query by $\mathcal{W}$ in order to answer a signature query made by $\mathcal{A}$: See steps (i) and (iii) of **Signature query** on how to program the random oracle in this situation.

(c) **Extract query,** $\mathcal{O}_{\varepsilon}(\mathtt{id})$:

If there exists a tuple $\langle \mathtt{id}_i, R_i, c_i, r_i, \beta_i \rangle$ in $\mathfrak{L}_{\mathrm{H}}$ such that $(\mathtt{id}_i = \mathtt{id}) \wedge (r_i \neq \perp)$

(i) If $\beta_i = 0$, $\mathcal{W}$ *aborts* ($\mathsf{abort}_{1,1}$).

(ii) Otherwise, $\beta_i = 1$ and $\mathcal{W}$ returns $\mathtt{usk} := (r_i + zc_i, R_i)$ as the user secret key.

Otherwise

(iii) $\mathcal{W}$ chooses $r \in_R \mathbb{Z}_p$, sets $R := g^r$ and queries the H-oracle for $c := \mathrm{H}(\mathtt{id}, R)$. It returns $\mathtt{usk} := (r + zc, R)$ as the secret key.

(d) **Signature query,** $\mathcal{O}_s(\mathtt{id}, m)$:

If there exists a tuple $\langle \mathtt{id}_i, R_i, c_i, r_i, \beta_i \rangle$ in $\mathfrak{L}_{\mathrm{H}}$ such that $(\mathtt{id}_i = \mathtt{id}) \wedge (r_i \neq \perp)$

(i) If $\beta_i = 0$, $\mathcal{W}$ selects $s, \in_R \mathbb{Z}_p$ and sets $d := s_\ell$, $A := g^s (g^\alpha)^{-r_i d}$. It then adds $\langle m, A, c, d, \ell \rangle$ to $\mathfrak{L}_{\mathrm{G}}$ (deferred case G_2)[22] and increments ℓ by one. The signature returned is $\sigma := (A, s + zcd, R_i)$.

(ii) Otherwise, $\beta_i = 1$ and the user secret key is $\mathtt{usk} := (y, R_i)$, where $y = r_i + zc_i$ and $R_i = g^{r_i}$. $\mathcal{W}$ then selects $a \in_R \mathbb{Z}_p$, sets $A := g^a$ and queries the G-oracle with $d := \mathrm{G}(m, A, c_i)$. The signature returned is $\sigma := (a + yd, R_i, A_i)$.

[22] In the unlikely event of there already existing a tuple $\langle m_i, A_i, c_i, d_i, \ell_i \rangle$ in $\mathfrak{L}_{\mathrm{G}}$ with $(m_i = m) \wedge (A_i = A) \wedge (c_i = c)$ but $(d_i \neq d)$ then $\mathrm{G}(m, A, c)$ cannot be set to d. In that case $\mathcal{W}$ can simply increment ℓ and repeat step (i).

Otherwise, $\mathcal{W}$ tosses a coin β with a bias δ (i.e, $\Pr[\beta = 0]{=}\delta$). The value of δ will be quantified on analysis.

(iii) If $\beta = 0$, $\mathcal{W}$ selects $c, s, r \in_R \mathbb{Z}_p$ and sets $d := s_\ell$, $R := (g^\alpha)^r$, $A := g^s(g^\alpha)^{-rd}$. Next, it adds $\langle \mathtt{id}, (g^\alpha)^r, c, r, 0\rangle$ to $\mathfrak{L}_{\mathrm{H}}$ (deferred case H_3), $\langle m, A, c, d, \ell\rangle$ to $\mathfrak{L}_{\mathrm{G}}$ (deferred case G_2) and increments ℓ by one.[23] The signature returned is $\sigma := (s + zc_id, R, A)$.

(iv) Otherwise, $\beta = 1$ and $\mathcal{W}$ selects $a, r \in_R \mathbb{Z}_p$ and sets $A := g^a, R := g^r$. It then queries the respective oracles with $c := \mathrm{H}(\mathtt{id}, R)$ and $d := \mathrm{G}(m, A, c)$. The signature returned is $\sigma := (a + (r + zc)d, R, A)$.

At the end of the simulation, a successful adversary outputs a valid forgery $\hat{\sigma} := (\hat{b}, \hat{R}, \hat{A})$ on a $(\hat{\mathtt{id}}, \hat{m})$. Let $\langle \mathtt{id}_j, R_j, c_j, r_j, \beta_j\rangle$ be the tuple in $\mathfrak{L}_{\mathrm{H}}$ that corresponds to the target H-query. Similarly, let $\langle m_i, A_i, c_i, d_i, \ell_i\rangle$ be the tuple in $\mathfrak{L}_{\mathrm{G}}$ that corresponds to the target G-query. $\mathcal{W}$ returns $(\ell_i, (\hat{b}, c_j, r_j, \beta_j, d_i))$ as its own output. Note that the side-output σ consists of $(\hat{b}, c_j, r_j, \beta_j, d_i)$. That concludes the description of the wrapper.

Correctness of the discrete-log. In the event of successful forking, $\mathcal{R}_1'$ obtains two (related) sets of side-outputs σ_0 and σ_1, where σ_i (for $i = 1, 2$) is of the form $(\hat{b}_i, c_i, r_i, \beta_i, d_i)$. Due to dependency, the adversary is forced to follow the order $\mathrm{H} < \mathrm{G}$. As a result, $\beta_0 = \beta_1$ (denoted by β), $c_0 = c_1$ (denoted by c) and $r_0 = r_1$ (denoted by r). In addition, let $\hat{a}$ denote $\log_g \hat{A}_1 = \log_g \hat{A}_0$. $\mathcal{W}$ aborts in the event that $\beta = 1$ ($\mathsf{abort}_{1,3}$). In the case $\beta = 0$, $\mathcal{R}_1'$ ends up with a system of two congruences $\{\hat{b}_0 = \hat{a} + (r\alpha + zc)d_0, \hat{b}_1 = \hat{a} + (r\alpha + zc)d_1\}$ in two unknowns $\{\hat{a}, \alpha\}$. α can be solved for as shown below.

$$\alpha = \frac{(\hat{b}_0 - \hat{b}_1) - zc(d_0 - d_1)}{r(d_0 - d_1)} \tag{17}$$

Notice that (17) is precisely what $\mathcal{R}_1'$ outputs in **Algorithm 3**.

B.1 Analysis

The probability analysis is governed by the three events $\mathsf{abort}_{1,1}$, $\mathsf{abort}_{1,2}$ and $\mathsf{abort}_{1,3}$. The accepting probability of $\mathcal{W}$ is the same as for $\mathcal{R}_1$ and, as a result, so is the probability of $\mathsf{abort}_{1,2}$. The probability of event $\mathsf{abort}_{1,3}$, on the other hand, is the same as that with which $(\beta = 1)$, *i.e.* $\Pr\left[\mathsf{abort}_{1,3} \mid \neg\mathsf{abort}_{1,2}\right] = (1 - \delta)$. On putting it all together, we get

$$\begin{aligned}
\epsilon_1' &= \Pr\left[\neg\mathsf{abort}_{1,3} \wedge \neg\mathsf{abort}_{1,2}\right] \\
&\geq (1 - (1 - \delta)) \cdot (1 - \delta)^{q_\varepsilon}\epsilon \cdot \left(\frac{(1 - \delta)^{q_\varepsilon}\epsilon}{q_{\mathrm{G}}} - \frac{1}{p}\right) \\
&= \delta \cdot (1 - \delta)^{q_\varepsilon}\epsilon \cdot \left(\frac{(1 - \delta)^{q_\varepsilon}\epsilon}{q_{\mathrm{G}}} - \frac{1}{p}\right)
\end{aligned} \tag{18}$$

Assuming $p \gg 1$, (18) attains maximum value at the point $\delta = 1/(1 + 2q_\varepsilon)$, at which

$$\epsilon_1' \geq \frac{\epsilon^2}{2\exp(1)q_{\mathrm{G}}q_\varepsilon}.$$

Here, exp is the base of natural logarithm.

[23] $\mathcal{W}$ chooses different randomisers if there is a collision as explained in **Footnote 22**.