Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Espen Sund

Multi-user security for Schnorr-like signature schemes

Bachelor's thesis in Mathematical Sciences
Supervisor: Jiaxin Pan
June 2022

NTNU
Kunnskap for en bedre verden

Espen Sund

Multi-user security for Schnorr-like signature schemes

Bachelor's thesis in Mathematical Sciences
Supervisor: Jiaxin Pan
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Abstract

In this thesis I will first consider a security proof of the Schnorr signature scheme from the single-user to the multi-user setting. The proof consists of several steps where we begin with the Schnorr identification scheme which is based on the hardness of the Discrete logarithm problem and conclude by showing that the Schnorr signature scheme derived through a Fiat-Shamir transformation is secure in the multi-user setting. This proof is heavily based on the generic security proof presented in the paper [1], but explicitly applied to the Schnorr signature scheme with additional intuition. Next, I will look at the Guillou-Quisquater signature scheme and in a similar fashion as the slightly tighter proof of the Schnorr signature scheme presented in appendix B in [1], show that this also holds for the Guillou-Quisquater signature scheme. Lastly, I will explain which scheme conditions are necessary for this proof to hold.

Introduction

Digital signature schemes are an essential part of cryptography because they make it possible for senders to authenticate themselves, as well as offer data integrity which means that the data being sent was with high probability not altered during the transmission. Creating efficient signature schemes is therefore of great interest. Two such schemes are the Schnorr signature scheme proposed by C.P. Schnorr in 1991 [2] and the Guillou-Quisquater signature scheme derived from the Guillou-Quisquater identification scheme proposed by L.C. Guillou and J.J. Quisquater in 1988 [3]. Both signature schemes are derived from identification schemes through a Fiat-Shamir transformation and have short signatures that are efficient to compute. Since the Guillou-Quisquater signature scheme has similar properties as the Schnorr signature scheme and Schnorr is more well-known, we refer to them as Schnorr-like signature schemes. The main difference between the two schemes is the hardness assumptions the schemes are based on; the Schnorr signature scheme is based on the discrete logarithm problem and Guillou-Quisquater is based on the RSA problem.

Contents

1 Preliminaries

1.1 Definitions

Definition 1.1 (**Set notation**). *A set of p number of elements will be denoted by $[p]$, that is $[p] := \{1, \ldots, p\}$. The notation $\mathbb{Z}_p$ will be used when we are dealing with the residual ring of order p, that is $\mathbb{Z}_p := \mathbb{Z}/p\mathbb{Z}$. A group will always be denoted by $\mathbb{G}$.*

Definition 1.2 (**Negligible function**). *A function we will denote $\epsilon(\lambda)$ is negligible if for every polynomial p we have $\lim_{\lambda\to\infty} \epsilon(\lambda)p(\lambda) = 0$, that is, the fraction $\frac{1}{\epsilon(\lambda)}$ grows faster than any polynomial p. For simplicity we will write it as ϵ.*

Definition 1.3 (**Probabilistic polynomial time** PPT). *We say that an algorithm $\boldsymbol{A}$ is a probabilistic polynomial time algorithm,* PPT *algorithm for short, if it runs in polynomial time and may use randomness to produce non-deterministic results. All algorithms will be* PPT *unless stated otherwise.*

Definition 1.4 (**Random picking notation**). *Given a set S the notation $s \leftarrow\$ S$ denotes picking an element s randomly and uniformly from the set S. If $\boldsymbol{A}$ is a* PPT *algorithm, then $a \leftarrow\$ \boldsymbol{A}(b)$ denotes the random variable a which is the output of $\boldsymbol{A}$ on input b.*

Definition 1.5 (**Advantage notation**). *A fundamental part of proving security of a cryptographic scheme is to find an upper bound on an adversary $\mathcal{A}$'s probability of breaking the scheme. We call this probability the advantage and denote it as $\mathsf{Adv}_{\mathcal{A}}^{xxx}(n)$, where $\mathcal{A}$ is the adversary, XXX is the scheme and n is a variable that measures the input size of the problem, called the security parameter. Mostly, however, we will just use a negligible number ϵ to express the adversary's advantage when the scheme is assumed secure.*

Definition 1.6 (**Security reductions**). *When proving security implications of the form; if scheme XXX is secure, then scheme YYY is secure, we will prove the contrapositive statement. That is, if there exists an adversary $\mathcal{A}$ breaking the security of scheme YYY, then there exists an adversary $\mathcal{B}$ breaking the security of scheme XXX. The algorithm $\mathcal{B}$ that breaks the security of scheme XXX, using $\mathcal{A}$ as a subroutine is called the reduction. Ideally, the running time of $\mathcal{A}$ and $\mathcal{B}$ should be the same, in this case we call the reduction tight. If this is not the case the reduction might lose effectiveness and we get some loss factor Q. The loss factor can be expressed as $\mathsf{Adv}_{\mathcal{A}}^{yyy}(n) \leq Q \cdot \mathsf{Adv}_{\mathcal{B}}^{xxx}(n)$.*

Definition 1.7 (**Discrete logarithm problem (DLOG)**). *Given parameter $par = (p, g, \mathbb{G})$ where p and g are the order and a generator of $\mathbb{G}$, respectively. The* ***DLOG*** *asks us to return x given $X := g^x$, where $x \leftarrow\$ \mathbb{Z}_p$. The discrete logarithm assumption assumes that* ***DLOG*** *is hard for any* PPT *adversary $\mathcal{A}$. More specific,* ***DLOG*** *is (t, ϵ)-hard if $\Pr[g^x = X \mid X \leftarrow\$ \mathbb{G}; x \leftarrow\$ \mathcal{A}(X)] \leq \epsilon$, where ϵ is a negligible function and t is a bound on the running time of $\mathcal{A}$ [1].*

Definition 1.8 (**Standard model**). *The standard model is a computational model in cryptography in which adversaries are only limited by the amount of time and computational power available [4]. Moreover, since cryptographic schemes usually are based on complexity assumptions like the discrete logarithm assumption, we say that a scheme is secure in the standard model if it can be proven secure by only using such assumptions. This is in contrast to the random oracle model.*

Definition 1.9 (**Random oracle model (ROM)**). *In the random oracle model parties are provided oracle access to a publicly available random function, a random oracle (RO). The random oracle model enables security proofs of several important schemes because reductions may exploit various properties of a RO that can be realized only to a limited extent (if at all) in the standard model. In the programmable random oracle model we allow the security reduction in the random oracle model to dynamically choose the output of the random oracle [5]. Whenever we do a security reduction in this thesis that does not mention random oracles or the random oracle model, the reduction is done in the standard model.*

Definition 1.10 (**Oracle access**). *Let $\mathcal{A}$ be an adversary and O be an oracle. The notation $\mathcal{A}^{\mathsf{O}}(\cdot)$ denotes that $\mathcal{A}$ has oracle access to O. In other words, $\mathcal{A}$ can make queries x_i to O and get $\mathsf{O}(x_i)$ in return.*

Definition 1.11 (**Cryptographic primitives and protocols**). *Cryptographic primitives are the basic building blocks in cryptography and include among others one-way functions, i.e., functions that are easy to compute but hard to invert. By combining primitives we get something called cryptographic protocols which includes identifications schemes which will be defined later.*

1.2 Theorems

Theorem 1.1 (Jensen's inequality [6]). *Let X be a random variable whose range is contained in $I \subset \mathbb{R}$ and $g : I \to \mathbb{R}$ be a convex function, then*

$$E[g(X)] \geq g(E[X])$$

In the following theorem, the Multi-Instance Reset Lemma 1.2, we need to make the randomized algorithm $\mathcal{C}$'s randomness explicit. This will be done by letting it take random coins ρ as input, where the coins refer to the random choices $\mathcal{C}$ will make. When the randomness has been made explicit, $\mathcal{C}(I, h; \rho)$ will be a deterministic algorithm on input (I, h) and the randomness ρ. The proof of 1.2 is based on the one given in [1], with additional intuition and more detailed calculations.

Theorem 1.2 (Multi-Instance Reset Lemma [1]). *Fix an integer $N \geq 1$ and a non-empty set H. Let $\mathcal{C}$ be a randomized algorithm that on input (I, h) returns a pair (b, σ), where b is a bit and σ is called the side output. Let IG be a randomized algorithm that we call the input generator. The accepting probability of $\mathcal{C}$ is defined as*

$$acc := \Pr[b = 1 \mid I \leftarrow\!\$\, IG; h \leftarrow\!\$\, H; (b, \sigma) \leftarrow\!\$\, \mathcal{C}(I, h)]$$

The (multi-instance) reset algorithm $\mathcal{R}_{\mathcal{C}}$ associated with $\mathcal{C}$ is the randomized algorithm that takes input $I_1, \ldots, I_N$ and proceeds as follows.

Algorithm $\mathcal{R}_{\mathcal{C}}$

```
For $i \in [N]$ :
    pick random coins $\rho_i$
    $h_i \leftarrow\!\$\, H$
    $(b_i, \sigma_i) \leftarrow\!\$\, \mathcal{C}(I_i, h_i; \rho_i)$
If $b_1 = \cdots = b_N = 0$ then return $(0, \epsilon, \epsilon)$    // Abort in Phase 1
Fix $i^* \in [N]$ such that $b_{i^*} = 1$
For $j \in [N]$ :
    $h'_j \leftarrow\!\$\, H$
    $(b'_j, \sigma'_j) \leftarrow\!\$\, \mathcal{C}(I_{i^*}, h'_j; \rho_{i^*})$
If $\exists j^* \in [N] : (h_{i^*} \neq h'_{j^*}$ and $b'_{j^*} = 1)$ then return $(i^*, \sigma_{i^*}, \sigma'_{j^*})$
Else return $(0, \epsilon, \epsilon)$                          // Abort in Phase 2
```

Let

$$res := \Pr[i^* \geq 1 \mid I_1, \ldots, I_N \leftarrow\!\!\$\, IG; (i^*, \sigma, \sigma') \leftarrow\!\!\$\, \mathcal{R}_{\mathcal{C}}(I_1, \ldots, I_N)]$$

Then

$$res \geq \left(1 - \left(1 - acc + \frac{1}{|H|}\right)^N\right)^2$$

Intuition: Note that the reset algorithm $\mathcal{R}_{\mathcal{C}}$ runs $\mathcal{C}$ in N related executions on input I^* in Phase 2 and acts according to its output. The key idea is that the reset algorithm "rewinds" $\mathcal{C}$ whenever it runs it on some new $h \leftarrow\!\!\$\, \mathrm{H}$ where I and ρ remain the same. By "rewind" we mean observing what $\mathcal{C}$ would have output with a different input before it was executed the first time.

Proof. For fixed instance I and coins ρ, we define the probabilities

$$\begin{aligned} acc(I, \rho) &:= \Pr_{h \leftarrow\$\, \mathrm{H}}[b = 1 \mid (b, \sigma) \leftarrow\!\!\$\, \mathcal{C}(I, h; \rho)] \\ res(I, \rho) &:= \Pr_{h,h' \leftarrow\$\, \mathrm{H}}[b = 1 \wedge b' = 1 \wedge h \neq h' \mid (b, \sigma) \leftarrow\!\!\$\, \mathcal{C}(I, h; \rho); (b', \sigma') \leftarrow\!\!\$\, \mathcal{C}(I, h', \rho')] \end{aligned}$$

As I and ρ are fixed, the two events $b = 1$ and $b' = 1$ are independent because $b = 1$ only depends on h and $b' = 1$ only depends on h'. Using this we can bound the probability $res(I, \rho)$ as follows (Note: All the following probabilities are actually conditioned and have $h, h' \leftarrow\!\!\$\, \mathrm{H}$, $(b, \sigma) \leftarrow\!\!\$\, \mathcal{C}(I, h; \rho)$ and $(b', \sigma') \leftarrow\!\!\$\, \mathcal{C}(I, h'; \rho)$ like in the definitions of $res(I, \rho)$ and $acc(I, \rho)$ above):

$$\begin{aligned} res(I, \rho) &= \Pr[b = 1 \wedge b' = 1 \wedge h \neq h'] \\ &= \Pr[b = 1] \cdot \Pr[b' = 1 \wedge h \neq h' \mid b = 1] \\ &\geq \Pr[b = 1] \cdot \left(\Pr[b' = 1 \mid b = 1] - \frac{1}{|\mathrm{H}|}\right) && (1) \\ &= \Pr[b = 1] \cdot \left(\Pr[b' = 1] - \frac{1}{|\mathrm{H}|}\right) \\ &= acc(I, \rho) \cdot \left(acc(I, \rho) - \frac{1}{|\mathrm{H}|}\right) && (2) \end{aligned}$$

In (1) we have used $\Pr[X \wedge Y] = \Pr[X] - \Pr[X \wedge \neg Y] \geq \Pr[X] - \Pr[\neg Y]$ and $\frac{1}{|\mathrm{H}|}$ accounts for the fact that $\Pr[\neg(h' \neq h)] = \Pr[h' = h] = \frac{1}{|\mathrm{H}|}$. With expectation taken over $I \leftarrow\!\!\$$ IG and random coins ρ, we also bound

$$\begin{aligned} \mathbb{E}_{I,\rho}[res(I, \rho)] &\geq \mathbb{E}_{I,\rho}\left[acc(I, \rho) \cdot \left(acc(I, \rho) - \frac{1}{|\mathrm{H}|}\right)\right] && (3) \\ &\geq \mathbb{E}_{I,\rho}[acc(I, \rho)] \cdot \left(\mathbb{E}_{I,\rho}[acc(I, \rho)] - \frac{1}{|\mathrm{H}|}\right) && (4) \\ &= acc \cdot \left(acc - \frac{1}{|\mathrm{H}|}\right) && (5) \end{aligned}$$

Above, we used (2) to get (3) and arrived at (4) by applying Jensen's inequality 1.1 to the convex function $\phi(X) = X \cdot (X + 1/|\mathrm{H}|)$. In (5) we have used $acc = \mathbb{E}_{I,\rho}[acc(I,\rho)]$ since expectation is taken over $I \leftarrow\!\!\$\ \mathrm{IG}$ and random coins ρ which is the definition of acc.
Next, consider the random variables b_{i^*} and b'_j ($j \in [N]$) as defined during the execution of the algorithm $\mathcal{R}_{\mathcal{C}}(I_1, \cdots, I_N)$. By using Bayes' rule for conditional probability on $acc = \Pr[b_{i^*} = 1]$ and $\Pr\left[b'_j = 1 \wedge b_{i^*} = 1\right] = \mathbb{E}_{I^*,\rho_{i^*}}[res(I^*, \rho_{i^*})]$, we obtain

$$\Pr\left[b'_j = 1 \;\middle|\; b_{i^*} = 1\right] = \frac{\Pr\left[b'_j = 1 \wedge b_{i^*} = 1\right]}{\Pr[b_{i^*} = 1]} \geq acc - \frac{1}{|\mathrm{H}|}$$

Which lets us bound the following probability

$$\begin{aligned}\Pr[\text{no abort in phase 2} \mid \text{no abort in phase 1}] &= 1 - \prod_{j=1}^{N}\left(1 - \Pr\left[b'_j = 1 \;\middle|\; b_{i^*} = 1\right]\right) \\ &\geq 1 - \left(1 - acc + \frac{1}{|\mathrm{H}|}\right)^N\end{aligned}$$

We also have

$$\Pr[\text{no abort in phase 1}] = \prod_{i=1}^{N}(1 - \Pr[b_i = 1]) = 1 - (1 - acc)^N$$

Combining $\Pr[\text{no abort in phase 1}]$ and $\Pr[\text{no abort in phase 2} \mid \text{no abort in phase 1}]$ we complete the proof by using Bayes' rule for conditional probability and $1 - (1 - acc)^N \geq 1 - (1 - acc + 1/|\mathrm{H}|)^N$ to get

$$res = \Pr[\text{no abort in phase 1} \wedge \text{no abort in phase 2}] \geq 1 - (1 - acc + \frac{1}{|\mathrm{H}|})^N)^2$$

□

2 Schemes

2.1 Canonical identification scheme

A canonical identification scheme is a three-round protocol between a prover P and a verifier V. P initializes the communication with a message R called the commitment, V then selects a uniform challenge h from the set of challenges ChSet, and upon receiving a response s from P, makes a deterministic decision. The scheme can be formally defined as follows [1]:
A canonical identification scheme ID is defined as a tuple of algorithms ID := (IGen, P, ChSet, V).

- The key generation algorithm IGen takes parameters par as input and outputs the key pair (pk,sk). Where we assume that pk defines the set of challenges ChSet.

- The prover algorithm $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ is split into two algorithms. P_1 takes the secret key sk as input and returns a commitment R and a state state; P_2 takes as input the secret key sk, a commitment R, a challenge h and state and returns a response s. It is important to note that state makes sure P_2 has all the secret information P_1 had before terminating.

- The verifier algorithm V takes as input the public key pk and the conversation transcript (R, h, s) and outputs a bit, 1 (acceptance) or 0 (rejection).

The canonical identification scheme we are going to use in the first part is the well-known Schnorr identification scheme ID_S, defined as follows [1]:

```
IGen(par)
1 :  $\mathsf{sk} := x \leftarrow\$ \mathbb{Z}_p$
2 :  $\mathsf{pk} := X = g^x$
3 :  $\mathrm{ChSet} := \{0,1\}^n$
4 :  return $(\mathsf{pk}, \mathsf{sk})$

V(pk, R, h, s)
1 :  If $R = g^s \cdot X^{-h}$ then return 1
2 :  Else return 0

P_1(sk)
1 :  $r \leftarrow\$ \mathbb{Z}_p; R = g^r$
2 :  $\mathsf{state} := r$
3 :  return $(R, \mathsf{state})$

P_2(sk, R, h, state)
1 :  Parse $\mathsf{state} = r$
2 :  return $s = x \cdot h + r \mod p$
```

Note that in line 1, the verifier V recomputes the commitment as $R' = g^s \cdot X^{-h}$ and returns 1 if and only if $R' = R$. An identification scheme with this property is called **commitment-recoverable** [1].

2.2 Signature scheme

A digital signature scheme SIG is defined as a triple of algorithms SIG := (Gen, Sign, Ver).

- The key generation algorithm Gen(par) returns the public and secret key pair $(\mathsf{pk}, \mathsf{sk})$.

- The signing algorithm $\mathrm{Sign}(\mathsf{sk}, m)$ returns a signature σ of message m.

- The deterministic verification algorithm $\mathrm{Ver}(\mathsf{pk}, m, \sigma)$ returns a bit 1 (acceptance) or 0 (rejection).

We also require correctness, that is for all key pairs $(\mathsf{pk}, \mathsf{sk}) \in \mathrm{Gen(par)}$, for all messages $m \in \{0,1\}^*$, we have $\mathrm{Ver}(\mathsf{pk}, m, \mathrm{Enc}(\mathsf{sk}, m)) = 1$ [1]. By $\{0,1\}^*$ we mean a bit string of arbitrary size.

2.2.1 The Fiat-Shamir transformation

In order to transform a canonical identification scheme into a signature scheme, one usually uses something called the Fiat-Shamir transformation [7]. This transformation gets rid of the interaction between the prover P and the verifier V, by letting P get the challenge h from a hash function $H : \{0,1\}^* \to \text{ChSet}$, where $h = H(R,m)$ and R is the commitment. Signature schemes derived from the standard Fiat-Shamir transform have signatures σ that consists of the commitment R and the response s. When proving the security of the Schnorr signature scheme in section 3 we will use the Schnorr signature scheme $\text{SIG}[\text{ID}_S]$ derived from the Schnorr identification scheme ID_S (1) through the standard Fiat-Shamir transformation, given by:

$\text{Gen}(\text{par})$	$\mathsf{Sign}(\mathsf{sk}, m)$	$\mathsf{V}(\mathsf{pk}, m, \sigma)$
$\mathsf{sk} := x \leftarrow\$ \mathbb{Z}_p$	$r \leftarrow\$ \mathbb{Z}_p; R = g^r$	Parse $\sigma = (R, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$
$\mathsf{pk} := X = g^x$	$h = H(R, m)$	$h = H(R, m)$
return $(\mathsf{pk}, \mathsf{sk})$	$s = x \cdot h + r \mod p$	If $R = g^s X^{-h}$ then **return** 1
	$\sigma = (R, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$	Else **return** 0
	return σ	

The hash function we use $H : \{0,1\}^* \to \text{ ChSet}$ is chosen such that $|\text{ChSet}| = 2^n < p = |\mathbb{Z}_p|$ [1]. It is worth noting that the signer Sign runs both of the prover algorithms P_1 and P_2 before and after calculating the challenge h, respectively. Since the Schnorr identification scheme ID_S is commitment-recoverable we can also use an alternative Fiat-Shamir transform that derives signature schemes with signatures σ consisting of the challenge h and the response s. The alternative Schnorr signature scheme we then get is [1]:

$\text{Gen}(\text{par})$	$\mathsf{Sign}(\mathsf{sk}, m)$	$\mathsf{V}(\mathsf{pk}, m, \sigma)$
$\mathsf{sk} := x \leftarrow\$ \mathbb{Z}_p$	$r \leftarrow\$ \mathbb{Z}_p; R = g^r$	Parse $\sigma = (h, s) \in \{0,1\}^n \times \mathbb{Z}_p$
$\mathsf{pk} := X = g^x$	$h = H(R, m)$	$R = g^s X^{-h}$
return $(\mathsf{pk}, \mathsf{sk})$	$s = x \cdot h + r \mod p$	If $h = H(R, m)$ then **return** 1
	$\sigma = (h, s) \in \{0,1\}^n \times \mathbb{Z}_p$	Else **return** 0
	return σ	

This type of alternative signature scheme will be used in the last section when working with the slightly tighter security proof then the one we get in section 3.2. Note: I will **not** use the alternative Schnorr signature scheme in this thesis, this will only be done for the Guillou-Quisquater signature scheme which will be introduced later.

2.3 Security notions

When we are defining and proving security of the identification and signature schemes, we will refer to them as (t, ϵ, Q_h)-XXX-YYY secure, where in the case

of the Schnorr identification scheme, XXX $\in$ {KR, IMP, PIMP} and YYY = KOA. The elements in the tuple t, ϵ and Q_h will be upper bounds on the adversary's running time, advantage and oracle queries, respectively. The tuple might contain other elements, which will be specified when necessary. The second part of the security notion of a scheme, XXX-YYY, is divided into two parts. The XXX-part defines the attacker's goal and the YYY-part defines the attacker's capabilities. We define the attacker's goals as follows: In key-recovery (KR) the attacker tries to compute a valid secret-key; in impersonation (IMP), it tries to impersonate a prover by convincing an honest verifier; in parallel impersonation (PIMP), it tries to impersonate a prover by convincing an honest verifier in Q_{CH} many parallel sessions. By honest verifier we mean a verifier that follows the scheme procedure. In the case of the adversary's capabilities, we have key-only attack (KOA), where the adversary is only given the public-key [1].
In the case of the Schnorr signature scheme SIG[ID$_S$] (0), we will refer to the attacker's goals as XXX $\in$ {UF, MU-UF, MU-SUF} and attacker's capabilities as YYY $\in$ {CMA,KOA}. Where UF stands for unforgeability, that is, the adversary tries to forge a valid signature of a message; in multi-user unforgeability (MU-UF), the adversary tries to forge a valid signature of a message in a setting with multiple users. For the attacker's capabilities, KOA, is the same as for identification schemes and in chosen-message attack (CMA), the adversary can choose messages and get valid signatures in return.

2.4 Scheme conditions

In this section we will define some useful conditions for identification schemes which will be important in the security proofs in the next section. Examples of such are honest-verifier zero-knowledge (HVZK), special soundness (SS), uniqueness and bits of min-entropy. We will also define the attacker's capabilities from the previous section for both identification schemes and signature schemes more explicitly.

Definition 2.1 (**Honest-verifier zero-knowledge** (HVZK))**.** *A canonical identification scheme ID is said to be HVZK if there exists an algorithm* $\mathsf{Sim}(\mathsf{pk})$ *that outputs a properly distributed transcript* (R, h, s) *with respect to the public key* pk *[1].*

Intuitively, this means that an adversary does not learn anything by honestly following the protocol to obtain valid transcripts, since it could generate such transcripts by itself by only using the public key.

Definition 2.2 (**Special Soundness** (SS))**.** *A canonical identification scheme ID is said to be special sound if there exists an extraction algorithm* $\mathsf{Ext}(\mathsf{pk}, R, h, s, h', s')$ *that for any key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$$ *IGen(par) and two accepting transcripts* (R, h, s) *and* (R, h', s') *where* $h \neq h'$, *outputs a secret key* sk^* *such that the pair* $(\mathsf{pk}, \mathsf{sk}^*)$ $\in$ *IGen(par) [1].*

Intuitively, this means that if the scheme is secure and satisfies SS it should be hard to come up with two valid transcripts with $h \neq h'$.

Definition 2.3 (**Random-self reducibility** (RSR)). *A canonical identification scheme ID is said to be Random-self reducible if there is an algorithm Rerand and two deterministic algorithms Tran and Derand such that for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$$ *IGen(par) the following holds [1]:*

- pk' *and* pk'' *have the same distribution, where* $(\mathsf{pk}', \tau') \leftarrow\$$ *Rerand(*pk*) is the randomized key pair and* $(\mathsf{pk}'', \mathsf{sk}'') \leftarrow\$$ *IGen(par) is a freshly generated key pair.*

- *For all* $(\mathsf{pk}', \tau') \in$ *Rerand(*pk*), all* $(\mathsf{pk}'', \mathsf{sk}'') \in$ *IGen(par), and* $\mathsf{sk}^* =$ *Derand(*$\mathsf{pk}, \mathsf{pk}', \mathsf{sk}', \tau'$*), we have* $(\mathsf{pk}, \mathsf{sk}^*) \in$ *IGen(par).*

- *For all* $(\mathsf{pk}', \tau') \in$ *Rerand(*pk*), all transcripts* (R', h', s') *that are valid with respect to* pk'*, the transcript* $(R', h', s := \mathit{Tran}(\mathsf{pk}, \mathsf{pk}', \tau', (R', h', s'))$ *is also valid with respect to* pk*.*

Intuitively, this means that for a scheme satisfying RSR, no problem instance is harder than the average case. To see this, note that a worst case problem instance can with high probability be reduced to an average case one by applying the Rerand algorithm and then turned back to the worst case problem instance by using the Derand algorithm.

Definition 2.4 (**Uniqueness** and **bits of min-entropy**). *An identification scheme ID is called unique if for all* $(\mathsf{pk}, \mathsf{sk}) \in$ *IGen(par),* $(R, \mathsf{state}) \in \mathsf{P}_1(\mathsf{sk})$*,* $h \in$ *ChSet, there exists at most one response* $s \in \{0,1\}^*$ *such that* $\mathsf{V}(\mathsf{pk}, R, h, s) = 1$*, i.e., the verifier accepts. A canonical identification scheme ID has* α *bits of min-entropy, if for all key-pairs* $(\mathsf{pk}, \mathsf{sk}) \in$ *IGen(par), the commitment generated by the prover algorithm is chosen from a distribution with at least* α *bits of min-entropy. That is, for all strings* R' *we have* $\Pr[R = R'] \le 2^{-\alpha}$*, if* R *was honestly generated by the prover [1].*

Definition 2.5 (**Key-recovery** (KR) [1]). *A canonical identification scheme is said to be secure against key-recovery under key-only attack* (t, ϵ)*-KR-KOA , if for all adversaries* $\mathcal{A}$ *with running time bounded by* t*,*

$$\Pr\left[(\mathsf{pk}, \mathsf{sk}^*) \leftarrow\$ \, \mathit{IGen(par)} \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \, \mathit{IGen(par)} \\ \mathsf{sk}^* \leftarrow\$ \, \mathcal{A}(\mathsf{pk}) \end{array} \right] \le \epsilon$$

Definition 2.6 (**(Parallel) Impersonation** (PIMP/IMP)). *A canonical identification scheme ID is said to be* (t, ϵ, Q_{CH})*-PIMP-KOA secure, if for all adversaries* $\mathcal{A}$ *with at most* Q_{CH} *queries to the challenge oracle CH,*

$$\Pr\left[\mathsf{V}(\mathsf{pk}, R_{i^*}, h_{i^*}, s_{i^*}) \wedge i^* \in [Q_{CH}] \;\middle|\; \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \, \mathit{IGen(par)} \\ \mathsf{state} \leftarrow\$ \, \mathcal{A}(\mathsf{pk}) \\ (i^*, s_{i^*}) \leftarrow\$ \, \mathcal{A}^{CH(\cdot)}(\mathsf{pk}) \end{array} \right] \le \epsilon$$

where on the i-th query CH(R_i*) the challenge oracle returns* $h_i \leftarrow\$$ *ChSet to* $\mathcal{A}$*. Since PIMP is only a parallel version of IMP with* Q_{CH} *queries to CH,* (t, ϵ)*-IMP-KOA is defined as* $(t, \epsilon, 1)$*-PIMP-KOA [1].*

Definition 2.7 (Multi-user strongly unforgeable against chosen message attacks (MU-SUF-CMA)**).** *A signature scheme is said to be* (t, ϵ, N, Q_s)*-MU-SUF-CMA secure if for all adversaries* $\mathcal{A}$ *running in time at most* t *and making at most* Q_s *queries to the signing oracle,*

$$\Pr\left[\begin{array}{l} \mathit{Ver}(\mathsf{pk}_{i^*}, m^*, \sigma^*) = 1 \\ \wedge (i^*, m^*, \sigma^*) \notin \{(i_j, m_j, \sigma_j) |\ j \in [Q_s]\} \end{array} \middle| \begin{array}{l} \mathit{For}\ i = 1, \cdots, N : (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow\$\ \mathit{Gen(par)} \\ (i^*, m^*, \sigma^*) \leftarrow\$\ \mathcal{A}^{\mathit{Sign}(\cdot,\cdot)}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N) \end{array}\right] \leq \epsilon$$

where on the j*-th query* $(i_j, m_j) \in [N] \times \{0,1\}^*$ *where* $j \in [Q_s]$ *the signing oracle Sign returns* $\sigma_j \leftarrow\$\ \mathit{Sign}(\mathsf{sk}_{i_j}, m_j)$ *to* $\mathcal{A}$*, i.e., a signature on message* m_j *under public key* pk_{i_j}*.*
Notice that this definition covers strong security, in the sense that a new signature on a previously queried message is considered as a fresh forgery. By fresh forgery we mean $(i^*, m^*, \sigma^*) \notin \{(i_j, m_j, \sigma_j) |\ j \in [Q_s]\}$*. If we modify this condition to* $(i^*, m^*) \notin \{(i_j, m_j) |\ j \in [Q_s]\}$*, i.e., to break the scheme the attacker needs to come up with a signature on a message-key pair that has not been queried to the signing oracle, we get MU-UF-CMA security. It is also worth noting that by not allowing any signing queries we get MU-UF-KOA, i.e.,* (t, ϵ, N)*-MU-UF-KOA :=* $(t, \epsilon, N, 0)$*-MU-UF-CMA.*
Lastly, in the single-user setting, $N = 1$ *users,* (t, ϵ, Q_s)*-SUF-CMA security is defined as* $(t, \epsilon, 1, Q_s)$*-MU-SUF-CMA security. Similarly, non-strong* (t, ϵ, Q_s)*-UF-CMA security is defined as* $(t, \epsilon, 1, Q_s)$*-MU-UF-CMA security [1].*

3 Security Implications

In this section we will prove the security of the Schnorr signature scheme and lastly put everything together to get the main theorem in subsection 3.2. In order to do this we need theorem 3.1 about the properties of the Schnorr identification scheme. The proof is the same as the one given in [1], with correctness being explicitly shown.

3.1 Properties of the Schnorr identification scheme

Theorem 3.1. *The Schnorr identification scheme* ID_S (1) *is a canonical identification scheme with* $\log p$ *bits of min-entropy and it is unique, has special soundness (SS), honest verifier zero-knowledge (HVZK) and is random self-reducible (RSR). Furthermore if DLOG is* (t, ϵ)*-hard, then* ID_S *is* (t, ϵ)*-KR-KOA secure [1].*

Proof. First we show correctness of ID_S, i.e., the verifier V should always accept if the honest prover P executes the protocol correctly, but this is the case because for a valid transcript (R, h, s) we have $g^s \cdot X^{-h} = g^{x \cdot h + r} \cdot g^{-x \cdot h} = g^r = R$. In addition, since R in $(R, \mathsf{state}) \leftarrow\$\ \mathsf{P}_1(\mathsf{sk})$ is uniformly random over $\mathbb{G}$, we have for all strings R', the probability $\Pr[R = R'] = \frac{1}{p} = 2^{-\log p}$, where $p = |\mathbb{G}|$. Which gives ID_S $\log p$ bits of min-entropy.

Uniqueness is also straightforward to verify. For all key pairs $(X, x) \in \text{IGen}(\text{par})$, $(R, \mathsf{state} := r) \in \mathsf{P}_1(\mathsf{sk})$ and $h \in \{0,1\}^n$, the value $s \in \mathbb{Z}_p$ satisfying $g^s = X^h \cdot R \iff s = x \cdot h + r$ is uniquely determined.

To show honest-verifier zero-knowledge we let the simulation algorithm $\text{Sim}(\mathsf{pk} := X)$ sample both the challenge $h \leftarrow\!\!\$\, \{0,1\}^n$ and the response $s \leftarrow\!\!\$\, \mathbb{Z}_p$ and then output $(R := g^s \cdot X^{-h}, h, s)$. Since s is uniformly random over $\mathbb{Z}_p$ and R is the unique value satisfying $R = g^s \cdot X^{-h}$, (R, h, s) is indeed a real transcript.

For special soundness we define the extractor algorithm as follows; $\text{Ext}(\mathsf{pk} := X, R, h, s, h', s') := x^* = (s - s')/(h - h')$, where (R, h, s) and (R, h', s') are the two accepting transcripts and $h \neq h'$. Notice that (X, x^*) is in fact a valid key pair, since $R = g^s \cdot X^{-h} = g^{s'} \cdot X^{-h'}$ which gives $X = g^{(s-s')/(h-h')}$.

For random self-reducibility, we will define the rerandomization algorithm Rerand and the deterministic algorithms Derand and Tran as follows:

- $\text{Rerand}(X)$ chooses $\tau' \leftarrow\!\!\$\, \mathbb{Z}_p$ and outputs the key pair $(X' := X \cdot g^{\tau'}, \tau')$. For all key pairs $(X, x') \in \text{IGen}(\text{par})$, X' is uniform and has the same distibution as X'', where $(X'', x'') \in \text{IGen}(\text{par})$.
- $\text{Derand}(X, X', x', \tau')$ outputs $x^* := x' - \tau'$. We have, for all $(X', \tau') \in \text{Rerand}(X := g^x)$ and $(X', x') \in \text{IGen}(\text{par})$, $X' = g^{x'}$ and $x' = x + \tau'$ and thus $x^* = x$.
- $\text{Tran}(X, X', \tau', (R', h', s'))$ outputs $s = s' - \tau' \cdot h'$. We have, for all $(X', \tau') \in \text{Rerand}(X := g^x)$, if (R', h', s') is a valid transcript with respect to $X' := g^{x+\tau'}$ then $s = s' - \tau' \cdot h' = (x + \tau') \cdot h' + r' - \tau' \cdot h' = x \cdot h' + r'$ and (R', h', s) is valid with respect to X.

For the final part of the proof we want to show that if DLOG is (t, ϵ)-hard, then ID_S is (t, ϵ)-KR-KOA secure, but by plugging in $(\mathsf{pk}, \mathsf{sk}) = (X := g^x, x)$ in the definition of (t, ϵ)-KR-KOA security, this is exactly the **DLOG** assumption. □

3.2 Main theorem

Theorem 3.2. *If the Schnorr identification scheme ID_S (1) is (t, ϵ)-KR-KOA secure then the Schnorr signature scheme SIG[ID_S] (0) is $(t', \epsilon', Q_s, Q_h)$-UF-KOA secure and $(t'', \epsilon'', Q_s, Q_h)$-MU-SUF-CMA secure in the programmable random oracle model, where*

$$\epsilon' \leq 4\epsilon + \frac{Q_s Q_h}{p}, \quad t' \approx t$$

$$\frac{\epsilon''}{t''} \leq 24(Q_h + 1) \cdot \frac{\epsilon}{t} + \frac{Q_s}{p}$$

Here p comes from ID_S's $\log p$ bits of min-entropy. In order to prove the main theorem we need to combine the four next lemmas which will be done in subsection 3.4.

3.3 The necessary lemmas

Lemma 3.1 (**IMP-KOA security**). *If ID_S (1) is (t, ϵ)-KR-KOA secure, then ID_S is (t', ϵ')-IMP-KOA secure, where for any $N \geq 1$,*

$$\epsilon \geq (1 - (1 - \epsilon' + \frac{1}{|ChSet|})^N)^2, \quad t \approx 2Nt' \tag{1}$$

In particular, the success ratios are related as

$$\frac{\epsilon'}{t'} - \frac{1}{t'|ChSet|} \leq 6 \cdot \frac{\epsilon}{t} \tag{2}$$

In order to show the lemma we need to use the Multi-Instance Reset Lemma [1.2].

Proof intuition: We define algorithm $\mathcal{C}$ from the multi-instance reset lemma to execute the adversary $\mathcal{A}$ in the IMP-KOA experiment, such that $\mathcal{A}$'s advantage ϵ' is equal to the accepting probability acc of $\mathcal{C}$ in the multi-instance reset lemma. Next, we define the reduction algorithm $\mathcal{B}$ in the KR-KOA experiment to run the reset algorithm $\mathcal{R}_{\mathcal{C}}$ associated to $\mathcal{C}$, such that $\mathcal{B}$'s advantage ϵ is equal to the probability res. It is important to note that $\mathcal{B}$ has to use both the RSR property and the SS property of ID_S to make this happen. Substituting the pair (res, acc) with (ϵ, ϵ') in the multi-instance reset lemma gives the desired result. As a sanity check we note that $\mathcal{R}_{\mathcal{C}}$ executes $\mathcal{C}$ and therefore $\mathcal{B}$ executes in fact $\mathcal{A}$.

Proof. We will first show how to get (2) from (1). If $\epsilon' \leq 1/|\mathrm{ChSet}|$, then (2) holds trivially, because:

$$\frac{\epsilon'}{t'} - \frac{1}{t'|\mathrm{ChSet}|} \leq \frac{1}{t'|\mathrm{ChSet}|} - \frac{1}{t'|\mathrm{ChSet}|} = 0 \leq 6 \cdot \frac{\epsilon}{t}$$

Let us therefore assume $\epsilon' > 1/|\mathrm{ChSet}|$, and set $N := (\epsilon' - 1/|\mathrm{ChSet}|)^{-1}$ to obtain $t \approx 2t'/(\epsilon' - 1/|\mathrm{ChSet}|)$ and

$$\epsilon \geq (1 - (1 - \epsilon' + \frac{1}{|\mathrm{ChSet}|})^N)^2 = (1 - (1 - \frac{1}{N})^N)^2 \geq (1 - \frac{1}{e})^2 \geq \frac{1}{3}.$$

Dividing ϵ by t yields (2).
To prove (2), let $\mathcal{A}$ be an adversary against the (t', ϵ')-IMP-KOA security of ID_S. We now construct an adversary $\mathcal{B}$ against the (t, ϵ)-KR-KOA security of ID_S, with (t, ϵ) as claimed in (1).
We use the Multi-Instance Reset Lemma [1.2], where H := ChSet and IG runs

$(\mathsf{pk} := g^x, \mathsf{sk} := x) \leftarrow\!\!\$\ \mathrm{IGen}$ and returns pk as instance I. We first define adversary $\mathcal{C}(\mathsf{pk}, h; \rho)$ that executes $\mathcal{A}(\mathsf{pk}; \rho)$, answers $\mathcal{A}$'s single query $R := g^r$ with $h \leftarrow\!\!\$\ \mathrm{ChSet}$, and finally receives $s := x \cdot h + r$ from $\mathcal{A}$. If transcript (R, h, s) is valid with respect to pk (i.e., $\mathsf{V}(\mathsf{pk} := X, R, h, s) = 1 \iff R = g^s \cdot X^{-h}$), $\mathcal{C}$ returns $(b = 1, \sigma = (R, h, s))$; otherwise it returns $(b = 0, \epsilon)$. By construction, $\mathcal{C}$ returns $b = 1$ if and only if $\mathcal{A}$ is successful, which means

$$acc = \epsilon'$$

Adversary $\mathcal{B}$ is defined as follows. For each $i \in [N]$, it uses the RSR property of ID_S to generate a fresh rerandomized key pair $(\mathsf{pk}_i := g^{x+\tau_i}, \tau_i) \leftarrow\!\!\$\ \mathrm{Rerand}(\mathsf{pk} := g^x)$. Next, it runs $(i^*, \sigma, \sigma') \leftarrow\!\!\$\ \mathcal{R}_{\mathcal{C}}(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$ with $\mathcal{C}$ defined as above. If $i^* \geq 1$, then both transcripts $\sigma = (R, h, s)$ and $\sigma' = (R, h', s')$ are valid with respect to pk_{i^*} and $h \neq h'$. $\mathcal{B}$ uses the SS property of ID_S and computes $\mathsf{sk}_{i^*} \leftarrow \mathrm{Ext}(\mathsf{pk}_{i^*}, R, h, s, h', s')$. Finally, using the RSR property of ID_S, it returns $\mathsf{sk} = \mathrm{Derand}(\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}, \tau_{i^*})$ and terminates. By construction, $\mathcal{B}$ is successful if and only if $\mathcal{R}_{\mathcal{C}}$ is. By the Multi-Instance Reset Lemma we can bound $\mathcal{B}$'s success probability as

$$\epsilon = res \geq (1 - (1 - \epsilon' + \frac{1}{|\mathrm{ChSet}|})^N)^2$$

The running time t of $\mathcal{B}$ is that of $\mathcal{R}_{\mathcal{C}}$, meaning $2Nt'$ plus the N times the time to run Rerand and Derand algorithms of RSR plus the time to run the Ext algorithm of SS. The approximation $t \approx 2Nt'$ indicates that this is the dominating running time of $\mathcal{B}$. □

Lemma 3.2 (PIMP-KOA security). *If ID_S (1) is (t, ϵ)-IMP-KOA secure, then ID_S is (t', ϵ', Q_{CH})-PIMP-KOA secure, where*

$$\epsilon' \leq Q_{CH} \cdot \epsilon, \quad t' \approx t$$

Proof intuition: Since the adversary $\mathcal{A}$ in the PIMP-KOA experiment can ask Q_{CH} challenge queries, whereas the reduction algorithm $\mathcal{B}$ in the IMP-KOA experiment only can ask one challenge query. The reduction will randomly choose one of $\mathcal{A}$'s challenge queries to answer with querying its own challenge oracle, which gives a $1/Q_{CH}$ success rate whenever $\mathcal{A}$ is successful.

Proof. Let $\mathcal{A}$ be an adversary against the $(t', \epsilon', Q_{\mathrm{CH}})$-PIMP-KOA security of ID_S. We now construct an adversary $\mathcal{B}$ against the (t, ϵ)-IMP-KOA security of ID_S, with (t, ϵ) as above.
First, $\mathcal{B}$ obtains $\mathsf{pk} := X := g^x$ from its IMP-KOA experiment and forwards it to $\mathcal{A}$. Next, it randomly picks $i^* \leftarrow\!\!\$\ [Q_{\mathrm{CH}}]$. On $\mathcal{A}$'s i-th query to the challenge oracle $\mathrm{CH}_{\mathcal{A}}(R_i)$, it proceeds as follows. If $i \neq i^*$, then $\mathcal{B}$ returns $h_i \leftarrow\!\!\$\ \mathrm{ChSet}$. Otherwise, it defines $R := R_{i^*}$ and queries its own challenge oracle $h \leftarrow\!\!\$\ \mathrm{CH}_{\mathcal{B}}(R)$ and returns $h_{i^*} := h$ to $\mathcal{A}$. Eventually, $\mathcal{A}$ returns its forgery (i, s) to $\mathcal{B}$ and terminates. If $i \neq i^*$, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ outputs the response s

to its experiment and terminates. We note that if $i = i^*$ then $\mathcal{B}$ wins whenever $\mathcal{A}$ wins. Since i^* is chosen uniformly in $[Q_{\mathrm{CH}}]$ the probability of $i = i^*$ is $1/Q_{\mathrm{CH}}$. Which gives rise to the loss factor of Q_{CH} in the reduction, that is, $\mathsf{Adv}_{\mathcal{A}}^{\text{pimp-koa}}(n) \leq Q_{\mathrm{CH}} \cdot \mathsf{Adv}_{\mathcal{B}}^{\text{imp-koa}}(n)$. Since the running time t of $\mathcal{B}$ is roughly that of $\mathcal{A}$, we get $t \approx t'$. $\square$

Lemma 3.3 (**UF-KOA security**). *If the Schnorr identification scheme ID_S (1) is (t, ϵ, Q_{CH})-PIMP-KOA secure, then the Schnorr signature scheme SIG[ID_S] (0) is (t', ϵ', Q_h)-UF-KOA secure in the programmable random oracle model, where*

$$\epsilon' = \epsilon, \quad t' \approx t, \quad Q_h = Q_{CH} - 1$$

Proof intuition: The reduction $\mathcal{B}$ in the PIMP-KOA experiment will answer $\mathcal{A}$'s oracle queries in the UF-KOA experiment by querying its own challenge oracle CH. When $\mathcal{A}$ outputs its forgery $(m, \sigma = (R, s))$, $\mathcal{B}$ will look up which challenge query $i \in [Q_{\mathrm{CH}}]$ corresponds to σ and return (i, s). Since (R, h, s) is a valid transcript if and only if $\mathcal{A}$ is successful, we get the equality $\epsilon' = \epsilon$.

Proof. Let $\mathcal{A}$ be an adversary against the (t', ϵ', Q_h)-UF-KOA security. We want to construct an adversary $\mathcal{B}$ against the $(t, \epsilon, Q_{\mathrm{CH}})$-PIMP-KOA security, with parameters $(t, \epsilon, Q_{\mathrm{CH}})$ as claimed.
First, $\mathcal{B}$ gets $\mathsf{pk} := X := g^x$ from its PIMP-KOA experiment and forwards it to $\mathcal{A}$. If $\mathcal{A}$ makes a query $(R_i := g^{r_i}, m_i)$ to the random oracle, $\mathcal{B}$ returns $\mathrm{H}(R_i, m_i)$ if it is already defined, otherwise $\mathcal{B}$ makes a query to its random oracle, $h_i \leftarrow\!\!\$\ \mathrm{CH}(R_i)$, and programs the random oracle $\mathrm{H}(R_i, m_i) := h_i$. Eventually, $\mathcal{A}$ submits a forgery $(m, \sigma = (R, s))$ and terminates. We assume that $(R := g^r, m) \in \{(R_i, m_i)\}_{i \in [Q_{\mathrm{CH}}]}$, i.e., $\mathrm{H}(R, m)$ was queried by $\mathcal{A}$. If not, $\mathcal{B}$ makes a dummy query to $\mathrm{H}(R_i, m_i)$, which is simulated as above. Taking into account the dummy query and Q_h being the maximum number of oracle queries $\mathcal{A}$ can make, there are maximum $Q_{\mathrm{CH}} := Q_h + 1$ queries to $\mathcal{B}$'s challenge oracle CH. Let $i \in [Q_h + 1]$ be the unique index such that $(R_i, m_i) = (R, m)$. Adversary $\mathcal{B}$ outputs (i, s_i) and terminates. We note that (R_i, h_i, s_i) is a valid transcript and hence breaks PIMP-KOA security if and only if $\mathcal{A}$'s forgery is valid, giving us the equality $\epsilon = \epsilon'$. The running time of $\mathcal{B}$ is roughly that of $\mathcal{A}$, hence $t' \approx t$. $\square$

Lemma 3.4 (**MU-UF-KOA security**). *If SIG[ID_S] (0) is (t, ϵ)-UF-KOA secure, then SIG[ID_S] is (t', ϵ', N)-MU-UF-KOA secure, where*

$$\epsilon' = \epsilon, \quad t' \approx t$$

Proof intuition: The reduction algorithm $\mathcal{B}$ will use the RSR property of ID_S and the public key it receives from its UF-KOA experiment to generate the key pairs for each of the N users in the MU-UF-KOA experiment. When the adversary $\mathcal{A}$ submits a forgery on some message m^*, $\mathcal{B}$ will run the tran algorithm to obtain a valid transcript on m^* under its own public key. Since $\mathcal{B}$ is successful whenever $\mathcal{A}$ is, we get the equality $\epsilon' = \epsilon$.

Proof. Let $\mathcal{A}$ be an algorithm that breaks (t', ϵ', N)-MU-UF-KOA security of SIG[ID$_S$]. We will construct an adversary $\mathcal{B}$ that breaks (t, ϵ)-UF-KOA security of SIG[ID$_S$] with (t, ϵ) as stated above.
Adversary $\mathcal{B}$ is executed in the UF-KOA experiment and obtains a public key $\mathsf{pk} := X := g^x$. To simulate the public keys input to $\mathcal{A}$, for each $i \in [N]$, $\mathcal{B}$ generates $(\mathsf{pk}_i := g^{x+\tau_i}, \tau_i) \leftarrow\!\$$ Rerand(pk) by using the random-self reducibility (RSR) property of ID$_S$. Then $\mathcal{B}$ runs $\mathcal{A}$ on the input $(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$. Eventually, $\mathcal{A}$ will output its forgery $(i^*, m^*, \sigma^* := (R^*, s^*))$ in the MU-UF-KOA experiment. The reduction $\mathcal{B}$ then computes $h^* = H(R^*, m^*)$ and runs $s \leftarrow\!\$$ Trans$(\mathsf{pk}, \mathsf{pk}_{i^*}, \tau_{i^*}, (R^*, h^*, s^*))$. By the RSR property of ID$_S$, the random variables $(\mathsf{pk}, R^*, h^*, s^*)$ and $(\mathsf{pk}_{i^*}, R^*, h^*, s^*)$ are identically distributed. If $\sigma^* := (R^*, s^*)$ is a valid signature on the message m^* under pk_{i^*}, then (R^*, s) is also a valid signature on m^* under pk. Thus, we have $\epsilon = \epsilon'$. The running time t of $\mathcal{B}$ is the running time t' of $\mathcal{A}$ plus N times the time to run Rerand and Tran algorithms, which gives $t \approx t'$.

□

Lemma 3.5 (**MU-SUF-CMA security**). *If SIG[ID$_S$] (0) is (t, ϵ, N, Q_h)-MU-UF-KOA secure, then SIG[ID$_S$] is $(t', \epsilon', N, Q_s, Q_h)$-MU-SUF-CMA secure in the programmable random oracle model, where*

$$\epsilon' \leq 4\epsilon + \frac{Q_h Q_s}{p}, \quad t' \approx t$$

N is the number of users, p comes from ID$_S$'s $\log p$ bit min-entropy and Q_s and Q_h are upper bounds on the number of signing queries and hash queries, respectively.

Proof intuition: After receiving the public keys from its MU-UF-KOA experiment, the reduction $\mathcal{B}$ needs to make sure it is able to simulate $\mathcal{A}$'s signing queries without aborting with high probability as well as eventually being able to turn $\mathcal{A}$'s forgery in the MU-SUF-CMA experiment into a forgery in its own experiment. To do this $\mathcal{B}$ flips a secret bit b_i for each of the N users. If $b_i = 1$ then $\mathcal{B}$ defines $\mathcal{A}$'s i-th public key to be its own i-th public key, otherwise it generates a key pair and passes on the public key. By doing casework one can find an upper bound on $\mathcal{B}$ aborting in the signing phase, which subtracted from $\mathcal{A}$'s advantage gives a lower bound on $\mathcal{B}$ not aborting. Combining this with the probability of $\mathcal{B}$ not aborting after receiving $\mathcal{A}$'s forgery gives the desired result.

Proof. Let $\mathcal{A}$ be an adversary that breaks the $(t', \epsilon', N, Q_s, Q_h)$-MU-SUF-CMA security of SIG[ID$_S$]. We will construct an adversary $\mathcal{B}$ that breaks the (t, ϵ, N, Q_h)-MU-UF-KOA security of SIG[ID$_S$] with (t, ϵ) as stated above. Adversary $\mathcal{B}$ obtains public keys $(\mathsf{pk}_1 := g^{x_1}, \cdots, \mathsf{pk}_N := g^{x_N})$ from its MU-UF-KOA experiment, and has access to a random oracle H.
To prepare the public keys to $\mathcal{A}$, for each $i \in [N]$, adversary $\mathcal{B}$ picks a secret bit $b_i \leftarrow\!\$ \{0, 1\}$. If $b_i = 1$ then $\mathcal{B}$ defines $\mathsf{pk}'_i := \mathsf{pk}_i$, otherwise $\mathcal{B}$ generates the key pair $(\mathsf{pk}'_i, \mathsf{sk}'_i) \leftarrow\!\$$ Gen(par) itself. By doing this, all simulated public keys are

correctly distributed.
Adversary $\mathcal{B}$ runs $\mathcal{A}$ on input $(\mathsf{pk}'_1, \cdots, \mathsf{pk}'_N)$ answering hash queries to random oracle H' and signing queries as follows.
In the simulation of hash queries, $\mathcal{B}$ answers a hash query $H'(R, m)$ from $\mathcal{A}$ by querying its own hash oracle $H(R, m)$ and returning its answer.
In the simulation of signing queries, $\mathcal{B}$ answers $\mathcal{A}$'s j-th signing query (i_j, m_j) with a signature σ_j on m_j under pk_{i_j} according to the following case direction:

- Case A: $b_{i_j} = 0$. In that case sk'_{i_j} is known to $\mathcal{B}$ and the signature is computed as $\sigma_j := (R_j, s_j) \leftarrow\$ \text{ Sign}(\mathsf{sk}_{i_j}, m_j)$. After running the signing algorithm which involves making a hash query, $\mathcal{B}$ defines $H'(R_j, m_j) := H(R_j, m_j)$.

- Case B: $b_{i_j} = 1$. In that case sk'_{i_j} is unknown to $\mathcal{B}$ and the signature is computed using the HVZK property of ID_S. This is done by $\mathcal{B}$ running the simulation algorithm Sim to get a valid transcript $(R_j, h_j, s_j) \leftarrow\$ \text{ Sim}(\mathsf{pk}'_{i_j})$. If the hash value $H'(R_j, m_j)$ was already defined in one of $\mathcal{A}$'s hash/signing queries and $H'(R_j, m_j) \neq h_j$, B aborts. Otherwise, it defines the random oracle
$$H'(R_j, m_j) := h_j \tag{3}$$
and returns $\sigma_j := (R_j, s_j)$, which is a correctly distributed valid signature on m_j under pk'_{i_j}. Note that by (3), $\mathcal{B}$ makes $H(R_j, m_j) \neq H'(R_j, m_j)$ with high probability. Also note that for each signing query, $\mathcal{B}$ aborts with probability at most Q_h/p because for every hash query $\Pr[R = R_j] = 1/p$ since ID_S has min-entropy $\log p$. Furthermore, since the number of signing queries is bounded by Q_s, $\mathcal{B}$ aborts overall with probability $Q_h Q_s/p$.

Eventually, $\mathcal{A}$ submit its forgery $(i^*, m^*, \sigma^* := (R^*, s^*))$, which we will assume is a valid forgery in the MU-SUF-CMA experiment. That is, for $h^* = H'(R^*, m^*)$ we have $\mathsf{V}(\mathsf{pk}'_{i^*}, R^*, h^*, s^*) = 1$. Furthermore, it satisfies the freshness condition, i.e.,
$$(i^*, m^*, R^*, s^*) \notin \{(i_j, m_j, R_j, s_j) |\ j \in [Q_s]\}. \tag{4}$$
Using (4) together with $\text{SIG}[\text{ID}_S]$'s uniqueness condition, i.e., s is uniquely determined by R and h, we get the following freshness condition:
$$(i^*, m^*, R^*) \notin \{(i_j, m_j, R_j) : j \in [Q_s]\}. \tag{5}$$

After receiving $\mathcal{A}$'s forgery, $\mathcal{B}$ computes a forgery for the MU-UF-KOA experiment according to the following case distinction.

- Case 1: There exists a $j \in [Q_s]$ such that $(m^*, R^*) = (m_j, R_j)$. If there is more than one j, fix any of them. In that case we have $h^* = h_j$ and furthermore $i^* \neq i_j$ by the freshness condition (5).
  - Case 1a: $b_{i^*} = 1$ and $b_{i_j} = 0$. Then the hash value $h^* = H'(R^*, m^*)$ was not programmed by $\mathcal{B}$ in (3). That means $h^* = H'(R^*, m^*) = H(R^*, m^*)$ and $\mathcal{B}$ returns $(i^*, m^*, (R^*, s^*))$ as a valid forgery to its MU-UF-KOA experiment

- Case 1b: $b_{i^*} = b_{i_j}$ or $b_{i^*} = 0 \land b_{i_j} = 1$. Then $\mathcal{B}$ aborts.

  Note that in case 1 we always have $i^* \neq i_j$ which means that $\mathcal{B}$ does not abort with probability 1/4 in which case it outputs a valid forgery.

- Case 2: For all $j \in [Q_s]$ we have $(m^*, R^*) \neq (m_j, R_j)$.

  - Case 2a: $b_{i^*} = 1$. Then the hash value $h^* = H'(R^*, m^*)$ was not programmed by $\mathcal{B}$ in (3). That means $h^* = H'(R^*, m^*) = H(R^*, m^*)$ and $\mathcal{B}$ returns $(i^*, m^*, (R^*, s^*))$ as a valid forgery to its MU-UF-KOA experiment.
  - Case 2b: $b_{i^*} = 0$. Then $\mathcal{B}$ aborts.

  Note that in case 2, $\mathcal{B}$ does not abort with probability 1/2 in which case it outputs a valid forgery.

Using this we can get a lower bound on $\mathcal{B}$'s probability of returning a valid forgery in its MU-UF-KOA experiment.

$$\epsilon \geq \min\left\{\frac{1}{4}, \frac{1}{2}\right\} \cdot \left(\epsilon' - \frac{Q_s Q_h}{p}\right) = \frac{1}{4}\left(\epsilon' - \frac{Q_s Q_h}{p}\right)$$

Note that the $\min(1/4, 1/2)$ factor makes sure the inequality holds even if $\mathcal{A}$'s forgery is in Case 1 or Case 2 and the $(\epsilon' - Q_s Q_h/p)$ factor is a lower bound on $\mathcal{B}$ not aborting in the signing phase. The running time of $\mathcal{B}$ is that of $\mathcal{A}$ plus the Q_s executions of Sim. We write $t' \approx t$. This concludes the proof. □

3.4 Proof of the main theorem

Proof. Putting it all together, we have the following:

- (t, ϵ)-KR-KOA $\rightarrow$ (t', ϵ')-IMP-KOA: $\quad \frac{\epsilon'}{t'} - \frac{1}{t'|\text{ChSet}|} \leq 6 \cdot \frac{\epsilon}{t}, \quad t \approx 2Nt'$
- (t, ϵ)-IMP-KOA $\rightarrow$ (t', ϵ')-PIMP-KOA: $\quad \epsilon' \leq Q_{\text{CH}} \cdot \epsilon, \quad t' \approx t$
- (t, ϵ)-PIMP-KOA $\rightarrow$ (t', ϵ')-UF-KOA: $\quad \epsilon' = \epsilon, \quad t' \approx t, \quad Q_h = Q_{\text{CH}} - 1$
- (t, ϵ)-UF-KOA $\rightarrow$ (t', ϵ')-MU-UF-KOA: $\quad \epsilon' = \epsilon, \quad t' \approx t$
- (t, ϵ)-MU-UF-KOA $\rightarrow$ (t', ϵ')-MU-SUF-CMA: $\quad \epsilon' \leq 4\epsilon + \frac{Q_h Q_s}{p}, \quad t' \approx t$

Since all running times after the first implication are approximately the same, we will assume $t' = t$. By combining the first two implications to KR-KOA $\rightarrow$ PIMP-KOA, we get $\frac{\epsilon'}{t' Q_{\text{CH}}} - \frac{1}{t'|\text{ChSet}|} \leq 6 \cdot \frac{\epsilon}{t}$, which is the same for KR-KOA $\rightarrow$ MU-UF-KOA, because ϵ stays the same and the running time t is assumed to be the same. By combining this with the last implication to KR-KOA $\rightarrow$ MU-SUF-CMA, we get $(\epsilon' - Q_h Q_s/p)/(4t' Q_{\text{CH}}) - 1/(t'|\text{ChSet}|) \leq 6 \cdot \epsilon/t$. After multiplying with $4Q_{\text{CH}}$ and rearranging terms, we get

$$
\begin{aligned}
\frac{\epsilon'}{t'} &\leq 24 \cdot \frac{Q_{\text{CH}} \cdot \epsilon}{t} + \frac{Q_h Q_s}{pt'} + \frac{4Q_{\text{CH}}}{t'|\text{ChSet}|} \\
&\leq 24(Q_h + 1) \cdot \frac{\epsilon}{t} + \frac{(t'-1)Q_s}{pt'} + \frac{4}{|\text{ChSet}|} && \text{(a)} \\
&\leq 24(Q_h + 1) \cdot \frac{\epsilon}{t} + \frac{Q_s}{p} && \text{(b)}
\end{aligned}
$$

In (a) we have used $Q_h \leq t' - 1$ and $Q_h = Q_{\text{CH}} - 1$. The reason for the bound $Q_h \leq t' - 1$ can be seen as the hash queries are not the major part of the running time. In (b) we have used $t' - 1 \approx t'$ and removed $4/|\text{ChSet}|$ since it is negligible. Proving the second part of the main theorem. To prove the first part of the theorem, we only need to combine the last two security implications to UF-KOA $\rightarrow$ MU-SUF-CMA which gives:

$$\epsilon' \leq 4\epsilon + \frac{Q_h Q_s}{p}$$

Completing the proof of the theorem. □

4 Guillou-Quisquater

4.1 Identification scheme

The Guillou-Quisquater identification scheme (ID_{GQ}) is based on the assumption that the RSA problem is hard for any PPT adversary $\mathcal{A}$. The RSA problem can be summarized as follows: Given an RSA public key (N, e) and a ciphertext $C := m^e \mod N$ one should efficiently compute m. The public key satisfies $N = p \cdot q$ for prime numbers p and q and $e \geq 3$ such that $\gcd(\phi(N), e) = 1$, where $\phi(N)$ is the number of numbers less than and relative prime to N [8]. We denote $\mathbb{Z}^*{}_N$ to be the set $\{y \in \mathbb{Z}_N \mid \gcd(N, y) = 1\}$ where $|\mathbb{Z}^*{}_N| = \phi(N) = pq - p - q + 1 = (p-1)(q-1)$. ID_{GQ} is defined as follows [1]:

```
IGen(par)
    $\mathsf{sk} := x \leftarrow\$ \mathbb{Z}^*{}_N$
    $\mathsf{pk} := X := x^e \mod N$
    $\text{ChSet} := \mathbb{Z}_e$
    return $(\mathsf{pk}, \mathsf{sk})$

V(pk, R, h, s)
    If $R = s^e \cdot X^{-h} \mod N$
    and $(R, s) \in \mathbb{Z}^*{}_N \times \mathbb{Z}^*{}_N$ then return 1
    Else return 0

P_1(sk)
    1 :  $r \leftarrow\$ \mathbb{Z}^*{}_N; R = r^e \mod N$
    2 :  $\mathsf{state} := r$
    3 :  return $(R, \mathsf{state})$

P_2(sk, R, h, state)
    1 :  Parse $\mathsf{state} = r$
    2 :  return $s = x^h \cdot r \mod N$
```

It is worth noting that e in ID_{GQ} is chosen as both a prime and coprime to $\phi(N)$, i.e. $\gcd(\phi(N), e) = 1$.

4.2 Properties of ID_{GQ}

Like the Schnorr identification scheme ID_{GQ} also satisfies correctness, uniqueness, SS, HVZK and RSR. The proof of theorem 4.1 is the same as the one presented in [1], with correctness being explicitly shown.

Theorem 4.1. *ID_{GQ} satisfies correctness, uniqueness, special soundness (SS), honest-verifier zero-knowledge (HVZK) and random-self reducibility (RSR). It also has $\alpha = \log(\phi(N))$ bit min-entropy [1].*

Proof. We first show correctness. This is satisfied because for a valid transcript (R, h, s) we have $s^e \cdot X^{-h} \equiv x^{he} \cdot r^e \cdot x^{-he} \equiv r^e \equiv R \mod N$ which means verifier V accepts. For α bit of min-entropy we have for $R' \leftarrow\!\$\, \mathbb{Z}^*{}_N$ the probability $\Pr[R = R'] = 1/\phi(N) = 1/2^\alpha \implies \alpha = \log(\phi(N))$.

To show uniqueness we note that for all key pairs $(\mathsf{pk} = X := x^e, \mathsf{sk} = x) \in \mathrm{IGen(par)}$, $R := r^e \in \mathsf{P}_1(\mathsf{sk})$ and $h \in \mathbb{Z}_e$ then $s \in \mathbb{Z}^*{}_N$ satisfying $s^e \equiv R \cdot X^h \mod N \Leftrightarrow s \equiv rx^h \mod N$ is unique, because $\gcd(\phi(N), e) = 1$ implies there exists a unique $d \in \mathbb{Z}^*{}_N$ such that $ed \equiv 1 \mod \phi(N)$.

For honest-verifier zero-knowledge we can let the simulator algorithm $\mathrm{Sim}(\mathsf{pk})$ first sample $h \leftarrow\!\$\, \mathbb{Z}_e$ and $s \leftarrow\!\$\, \mathbb{Z}^*{}_N$ and then output the transcript $(R := s^e X^{-h}, h, s^*)$. Since (h, s) is uniform over $\mathbb{Z}_e \times \mathbb{Z}^*{}_N$ and R is the unique value satisfying $R \equiv s^e X^{-h} \mod N$.

For special soundness if given two accepting transcripts (R, h, s) and (R, h', s') with $h \neq h'$ we need to define an extractor algorithm $\mathrm{Ext}(X, R, h, s, h', s')$ that returns a valid secret key for the public key X. Since the transcripts are valid and assuming $h > h'$ we have $s^e X^{-h} \equiv R \equiv s'^e X^{-h'} \mod N$ which implies $(s/s')^e \equiv X^{h-h'} \mod N$. We also have $\gcd(e, h - h') = 1$ since e is a prime and $h, h' \in \mathbb{Z}_e$. This allows us to use the extended Euclidean algorithm to find integers $A, B \in \mathbb{Z}^*{}_N$ such that

$$A(e) + B(h - h') = \gcd(e, h - h') = 1$$

Then we define an extractor algorithm $\mathrm{Ext}(X, R, h, s, h', s') := x^* := X^A (s/s')^B$. Note that x^* is a valid secret key for X since $(x^*)^e = X^{A(e)} (s/s')^{B(e)} = X^{A(e)+B(h-h')} = X$

For random-self reducibility the Rerand algorithm and the deterministic algorithms Derand and Tran are defined as follows:

- $\mathrm{Rerand}(X_1)$ chooses $\tau_2 \leftarrow\!\$\, \mathbb{Z}^*{}_N$, computes $X_2 := X_1 \cdot {\tau_2}^e \mod N$ and returns (X_2, τ_2). Since X_2 is uniform and has the same distribution as X_3, where $(X_3, x_3) \leftarrow\!\$$ $\mathrm{IGen(par)}$, (X_2, τ_2) is indeed a valid rerandomized key pair under public key X_1.

- Derand(X_1, X_2, x_2, τ_2) outputs $x^* = x_2/\tau_2 \mod N$. We have, for all $(X_2, \tau_2) \leftarrow\$$ Rerand$(X_1 := x_1^e \mod N)$ with $(X_2, x_2) \in$ IGen(par), $X_2 = x_2^e \mod N$ and $x_2 = x_1 \cdot \tau_2 \mod N$ and thus $x^* = x_1 \mod N$.
- Tran$(X_1, X_2, \tau_2, (R_2 := r_2^e \mod N, h_2, s_2))$ outputs $s_1 = s_2/\tau_2^{h_2} \mod N$. We have, for all $(X_2, \tau_2) \in$ Rerand$(X_1 := x_1^e \mod N)$, if (R_2, h_2, s_2) is valid with respect to $X_2 := (x_1 \cdot \tau_2)^e \mod N$ then $s_1 = s_2/\tau_2^{h_2} = (x_1\tau_2)^{h_2} \cdot r_2/\tau_2^{h_2} = x_1^{h_2} \cdot r_2 \mod N$ and (R_2, h_2, s_1) is valid with respect to X_1.

□

4.3 Signature scheme

Since ID$_{GQ}$ (0) is commitment-recoverable we can apply the alternative Fiat-Shamir transform to arrive at the following Guillou-Quisquater signature scheme SIG[ID$_{GQ}$]:

```
Gen(par)                         Sign(sk, m)                           V(pk, m, σ)
sk := $x \leftarrow\$ \mathbb{Z}^*_N$      $r \leftarrow\$ \mathbb{Z}^*_N; R = r^e \mod N$     Parse $\sigma = (h, s) \in \mathbb{Z}_e \times \mathbb{Z}^*_N$
$X := x^e \mod N$                 $h = H(R, m)$                          $R = s^e X^{-h} \mod N$
pk := $X$                          $s = x^h \cdot r \mod N$                If $h = H(R, m)$ and $R \in \mathbb{Z}^*_N$ then **return** 1
**return** (pk, sk)                $\sigma = (h, s) \in \mathbb{Z}_e \times \mathbb{Z}^*_N$        Else **return** 0
                                 **return** $\sigma$
```

Where the hash function is of the form $H : \{0,1\}^* \to \mathbb{Z}_e$ [1].

4.4 Tighter security reduction

The next security proof shows that SUF-CMA security tightly implies MU-SUF-CMA security in the standard model. This can be seen as an improvement of the first part of the main theorem 3.2 which was done in the programmable random oracle model. It is an improvement of 3.2, because lemma 3.5 can in the single-user setting (UF-KOA → SUF-CMA) be proven more tightly secure, i.e., with constant factor 1 [1]. The reason will not be discussed in this thesis. Combining this improved bound (UF-KOA → SUF-CMA) with the next theorem (SUF-CMA → MU-SUF-CMA) gives a better bound than lemma 3.5.

Theorem 4.2. *If the Guillou-Quisquater signature scheme SIG[ID$_{GQ}$] is (t, ϵ, Q_s)-SUF-CMA secure then, for any $N' \geq 1$, SIG[ID$_{GQ}$] is (t', ϵ', N', Q_s)-MU-SUF-CMA secure, where*

$$\epsilon' \leq 2\epsilon + \frac{Q_s^2}{(p-1)(q-1)}, \quad t' \approx t$$

Q_s is an upper bound on the number of signing queries and N' is the number of users, not to be confused with $N = pq$.

Proof. Let $\mathcal{A}$ be an adversary breaking the (t', ϵ', N', Q_s)-MU-SUF-CMA security of SIG[ID$_{GQ}$]. We will construct an adversary $\mathcal{B}$ breaking the (t, ϵ, Q_s)-SUF-CMA security of SIG[ID$_{GQ}$] with t and ϵ as above. $\mathcal{B}$ is executed in the SUF-CMA experiment and obtains $\mathsf{pk} = X = x^e$ and has access to a signing oracle Sign.
In the simulation of public keys, for each $i \in [N']$, adversary $\mathcal{B}$ picks $a_i \leftarrow\$ \mathbb{Z}^*{}_N$, secret bits $b_i \leftarrow\$ \{0,1\}$ and computes

$$\mathsf{pk}_i = X_i := X^{b_i} \cdot {a_i}^e \tag{6}$$

That is, if $b_i = 0$ then $\mathsf{sk}_i = a_i$ is known to $\mathcal{B}$, otherwise $b_i = 1$ and $\mathsf{sk}_i = xa_i$ is unknown to $\mathcal{B}$. Adversary $\mathcal{B}$ then runs $\mathcal{A}$ on input $(\mathsf{pk}_1, \cdots, \mathsf{pk}_N)$ and answers $\mathcal{A}$'s signing queries as follows:
On $\mathcal{A}$'s j-th signing query $(i_j, m_j) \in [N'] \times \{0,1\}^*$, $\mathcal{B}$ is supposed to return σ_j on message m_j under pk_{i_j}. These are calculated as follows:

- Case A: $b_{i_j} = 0$. In this case $\mathsf{sk}_{i_j} = a_{i_j}$ is known to $\mathcal{B}$ and the signature is computed as $\sigma_j := (h_j, s_j) \leftarrow\$ \mathsf{Sign}(\mathsf{sk}_{i_j}, m_j)$.
- Case B: $b_{i_j} = 1$. In that case $\mathsf{sk}_{i_j} = xa_{i_j}$ is unknown to $\mathcal{B}$ and the signature is computed using $\mathcal{B}$'s signing oracle by first querying $(h_j, \hat{s}_j) \leftarrow\$ \mathsf{Sign}(m_j)$. Then $\sigma_j = (h_j, s_j := \hat{s}_j \cdot a_i^{h_j})$ is a valid signature on m_j under pk_{i_j}. Indeed, $\mathsf{V}(\mathsf{pk}_{i_j}, m_j) = 1$ because $H(s_j^e X_{i_j}^{-h_j}, m_j) = H((\hat{s}_j a_{i_j}^{h_j})^e X_{i_j}^{-h_j}, m_j) = H(\hat{s}_j^e X^{-h_j}, m_j) = h_j$.

Adversary $\mathcal{B}$ returns $\sigma_j = (h_j, s_j)$ which in both cases is a correctly distributed valid signature. For future reference we define $R_j := s_j^e \cdot X_{i_j}^{-h_j}$ and by (6)

$$r_j := R_j^{1/e} = s_j \cdot (x^{b_{i_j}} a_{i_j})^{-h_j} \tag{7}$$

We assume that

$$\forall k \neq j \in [Q_s] : r_k \neq r_j \tag{8}$$

Since s_j and therefore also r_j are uniform elements from $\mathbb{Z}^*{}_N$, condition (8) is not satisfied with probability at most $Q_s^2/\phi(N)$. This can be seen as

$$\begin{aligned}
\Pr[(8) \text{ is satisfied}] &= \frac{\phi(N)(\phi(N)-1)\cdots(\phi(N)-Q_s+1)}{\phi(N)^{Q_s}} \\
&\geq \frac{(\phi(N)-Q_s)^{Q_s}}{\phi(N)^{Q_s}} \\
&\geq \frac{\phi(N)^{Q_s} - \binom{Q_s}{1}\phi(N)^{Q_s-1}Q_s}{\phi(N)^{Q_s}} && \text{(a)} \\
&= 1 - \frac{Q_s^2}{\phi(N)}
\end{aligned}$$

In (a) we have used the fact that the sum of two consecutive terms in the expansion of $(\phi(N) - Q_s)^{Q_s}$ is non-negative since $\phi(N)$ is of exponential size

and therefore larger than a polynomial factor times Q_s.
Eventually $\mathcal{A}$ will submit a forgery $(i^*, m^*, \sigma^* := (h^*, s^*))$ and terminate. For the remainder of the proof we will assume σ^* is a valid forgery on m^* under pk_{i^*}, i.e., for $R^* := s^{*e} \cdot X_{i^*}^{-h^*}$ it holds that $H(R^*, m^*) = h^*$. Furthermore we assume σ^* is a valid fresh signature in the MU-SUF-CMA experiment:

$$(i^*, m^*, h^*, s^*) \notin \{(i_j, m_j, h_j, s_j) \mid j \in [Q_s]\} \tag{9}$$

After receiving $\mathcal{A}$'s forgery, $\mathcal{B}$ is supposed to compute its own valid forgery under $\mathsf{pk} = X$. Adversary $\mathcal{B}$ defines the set of all indices j such that it queried m_j to its signing oracle $J := \{j \in [Q_s] \mid b_{i_j} = 1\}$ and signs the messages as follows:

- Case 1: For all $j \in [Q_s]$ we have: $h^* \neq h_j$ or $r^* \neq r_j$.
  - Case 1a: $b_{i^*} = 1$. Then for $\hat{s}^* := s^* \cdot a_{i^*}^{-h^*}$ we have

    $$H(\hat{s}^{*e} X^{-h^*}, m^*) = H(s^{*e} X_{i^*}^{-h^*}, m^*) = h^*$$

    and hence

    $$\hat{\sigma}^* := (h^*, \hat{s}^*)$$

    is a correct signature on m^* under $\mathsf{pk} = X$. We now show that it is a fresh signature in the SUF-CMA experiment. If $h^* \notin \{h_1, \cdots, h_{Q_s}\}$ then we directly obtain $\hat{\sigma}^* := (h^*, \hat{s}^*) \notin \{(h_j, \hat{s}_j) \mid j \in J\}$, which means $(m^*, \hat{\sigma}^*)$ satisfies the freshness condition of the SUF-CMA experiment. On the other hand, if the set J^* of indices $j \in [Q_s]$ such that $h_j = h^*$ is non-empty, then we will use the condition $r_j \neq r^*$ to show that the corresponding $\hat{s}_j$ are all distinct from $\hat{s}^*$. Indeed, for all $k \in J \cap J^*$ we have $\hat{s}_k = x^{h^*} \cdot r_j \neq x^{h^*} \cdot r^*$ and therefore $\hat{s}^* = x^{h^*} \cdot r^* \notin \{\hat{s}_k \mid k \in J \cap J^*\}$. For all $k \in J \backslash J^*$ we have $h_k \neq h^*$ so $\hat{\sigma}^*$ also in this case satisfies the freshness condition of the SUF-CMA experiment.
  - Case 1b: $b_{i^*} = 0$. Then $\mathcal{B}$ aborts.

Note that $\mathcal{B}$ aborts with probability 1/2 in case 1. If it does not abort, it outputs a valid strong forgery.

- Case 2: There exists a $j \in [Q_s]$ such that $h^* = h_j$ and $r^* = r_j$ and $i^* = i_j$.
  - Case 2a: $b_{i_j} = 1$. As in case 1a,

    $$\hat{\sigma}^* := (h^*, \hat{s}^* := (s^* \cdot a_{i^*}^{-h^*}))$$

    is a correct signature on message m^* under $\mathsf{pk} = X$. By $r^* = r_j$ and $h^* = h_j$ we have $(h^*, s^*) = (h_j, s_j)$. Since we also have $i^* = i_j$, $\mathcal{A}$'s freshness condition (9) implies $m_j \neq m^*$ meaning $\hat{\sigma}^*$ is a valid fresh forgery in the SUF-CMA experiment.
  - Case 2b: $b_{i^*} = 0$. Then $\mathcal{B}$ aborts.

Note that $\mathcal{B}$ aborts with probability 1/2 in case 2. If it does not abort, it outputs a valid strong forgery.

- Case 3: There exists a $j \in [Q_s]$ such that $h^* = h_j \neq 0$ and $r^* = r_j$ and $i^* \neq i_j$.
  Note that if j exists it is uniquely defined by (8).
  - Case 3a: $b_{i_j} \neq b_{i^*}$. From (7), $r^* = r_j$ and $h^* = h_j \neq 0$ we get the two equations

    $$\begin{aligned} r^* &= s^*(x^{b_{i^*}} a_{i^*})^{-h^*} \\ r^* &= s_j(x^{b_{i_j}} a_{i_j})^{-h^*} \end{aligned}$$

    From which $\mathcal{B}$ can extract the single-user secret key $\mathsf{sk} = x$, since $b_{i_j} \neq b_{i^*}$. Using $\mathsf{sk} = x$ $\mathcal{B}$ computes a valid forgery on any fresh message.
  - Case 3b: $b_{i_j} = b_{i^*}$. Then $\mathcal{B}$ aborts.

In case 3, note that since $b_{i_j} = b_{i^*}$ with probability 1/2, it aborts with probability 1/2. If it does not abort, it outputs a valid strong forgery.

- Case 4: There exists a $j \in [Q_s]$ such that $h^* = h_j = 0$ and $r^* = r_j$ and $i^* \neq i_j$.
  Again if j exists it is uniquely defined by (8).
  - Case 4a: $b_{i_j} = 0$. Then

    $$\hat{\sigma}^* := (0, s^*)$$

    is a correct signature on m^* under $\mathsf{pk} = X$, since $H({s^*}^e, m^*) = h^* = 0$ from definition. For all $k \neq j$ with $h_k = h^* = 0$ we have by (8) $r^* \neq r_k$ and therefore $s^* = r^* \neq r_k = \hat{s}_k$. This means that $\hat{\sigma}^* = (0, s^*) = (0, r^*) \notin \{(h_k, \hat{\sigma}_k) \mid k \in J\}$. Therefore $(m^*, \hat{\sigma}^*)$ satisfies the freshness condition in the SUF-CMA experiment.
  - Case 4b: $b_{i_j} = 1$. Then $\mathcal{B}$ aborts.

Note that in case 4, $\mathcal{B}$ aborts with probability exactly 1/2. If it does not abort, it outputs a valid strong forgery.
Putting everything together, $\mathcal{B}$ returns a fresh strong forgery $(m^*, \hat{\sigma}^*)$ under $\mathsf{pk} = X$ with probability $\epsilon = \frac{1}{2}(\epsilon' - \frac{Q_s^2}{(p-1)(q-1)})$. Adversary $\mathcal{B}$ makes at most Q_s signing queries. Its running time is that of $\mathcal{A}$ in addition to some small computation for each signing query and each user, hence $t' \approx t$.

□

4.5 Necessary conditions

In the security proof from last subsection 4.2 we showed a tighter reduction for Guillou-Quisquater signature scheme (0) than the one we presented in the first part of the main theorem 3.2 for Schnorr signature scheme. Since SIG[ID$_{GQ}$] satisfies all the properties listed in 4.1 we know that these are sufficient for the security proof in 4.2. To finish my thesis, I will now explain which of these are necessary.

Correctness. By definition we required a canonical identification scheme to satisfy correctness. Therefore correctness is a **necessary** property.

Uniqueness. In Case 2a we obtain $s_j = s^*$ from $h_j = h^*$ and $r^* = r_j$. Therefore uniqueness is a **necessary** property.

Honest-verifier Zero-knowledge (HVZK). At no point during the proof do we rely on a simulator algorithm to produce a valid and correctly distributed transcript. Therefore HVZK is not a necessary property.

Special Soundness (SS). In Case 3a we can extract a secret key under the condition $h^* = h_j$. Since this is the only case where we can extract a secret key and the extractor algorithm in SS require two accepting transcripts with $h^* \neq h_j$, SS is not a necessary property.

Random Self-reducibility (RSR). During the simulation of the public keys $\mathcal{B}$ needs to rerandomize the public key from its experiment by using the Rerand algorithm in RSR. In Case 1a $\mathcal{B}$ also uses the Tran algorithm in RSR to get a valid signature under its public key. Therefore RSR is a **necessary** property.

References

[1] E. Kiltz, D. Masny, and J. Pan, "Optimal security proofs for signatures from identification schemes," Cryptology ePrint Archive, Report 2016/191, 2016, https://eprint.iacr.org/2016/191.

[2] C. P. Schnorr, "Efficient signature generation by smart cards," *J. Cryptol.*, vol. 4, no. 3, p. 161–174, jan 1991. [Online]. Available: https://doi.org/10.1007/BF00196725

[3] L. Guillou and J.-J. Quisquater, "A "paradoxical" indentity-based signature scheme resulting from zero-knowledge," 01 1995, pp. 216–231.

[4] D. Naccache, *Standard Model.* Boston, MA: Springer US, 2011, pp. 1253–1253. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_518

[5] M. Fischlin, A. Lehmann, T. Ristenpart, T. Shrimpton, M. Stam, and S. Tessaro, "Random oracles with(out) programmability," in *Advances in Cryptology – ASIACRYPT 2010*, ser. Lecture Notes in Computer Science, M. Abe, Ed., vol. 6477. Singapore: Springer, Heidelberg, Germany, Dec. 5–9, 2010, pp. 303–320.

[6] F. M. Dekking, C. Kraaikamp, H. P. Lopuhaä, and L. E. Meester, "A modern introduction to probability and statistics," 2005.

[7] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[8] R. L. Rivest and B. Kaliski, *RSA Problem.* Boston, MA: Springer US, 2011, pp. 1065–1069. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_475