One-more security assumptions

This report was written as handout for the 2025 edition of the self-study PhD course "Cryptographic Proof Techniques" by Prof. Elena Pagnin at Chalmers University of Technology, Göteborg, Sweden. It includes theoretical details from cited literature and the formalisation of some personal remarks.

The current version was updated on 2025-07-11 with feedback received during the presentation.

1 Introduction: on provable security

GAME-BASED PROOFS. We can establish the security of a scheme Π in terms of the goals of an attacker $\mathcal{A}$ and the tools it can access. In game-based security proofs, this is moelled as a *security game*: goals are the winning conditions, tools become oracle queries. For instance, a standard security notion for signature schemes Π_{Sign} is *existential unforgeability under chosen message attack* (EUF-CMA), where $\mathcal{A}$ has access to a signing oracle $\mathcal{O}_{\mathsf{Sign}}$ and aims at providing a valid message-signature pair (m^*, σ^*) for a new message m^*.

SECURITY REDUCTIONS. Security claims for signature schemes Π_{Sign} are usually of the form

"If prob is hard, scheme Π_{Sign} is EUF-CMA secure"

where prob is a computational problem believed to be hard to solve. In game-based proofs, such claims are proved via the counternominal proposition:

"If Π_{Sign} is not EUF-CMA secure, then prob is not hard."

In other words we show that, with access to a PPT adversary $\mathcal{A}$ that breaks the EUF-CMA security of Π_{Sign} with non-negligible probability $\epsilon_{\mathcal{A}}$, we are able to exhibit a PPT extractor $\mathcal{E}$ able to simulate an execution of Π_{Sign} to break prob with non-negligible probability $\epsilon_{\mathcal{E}}$. This success probability, compared to the best algorithms known to attack prob, allows to set the parameters of the scheme, so we want the reduction to be as tight as possible.

2 The One-More RSA Inversion assumption

ONE-MORE ONE-WAY FUNCTIONS. *One-More forgeries* were first formalised by Pointcheval and Stern in [1]. Intuitively, they wish to capture the attacker capability of querying an oracle before the challenge even starts. This is done by allowing t interactions with some oracle and then asking to produce "one more" forgery, the $(t+1)$-th. This later inspired the formal definition of *one-more one-way function*, introduced by Bellare et al. in [2] as an interactive game between a challenger

$\mathcal{C}$ and adversary $\mathcal{A}$. The challenger $\mathcal{C}$ generates $t+1$ inversion challenges. The adversary $\mathcal{A}$ can query the inversion oracle $\mathcal{O}^{\mathsf{inv}\text{-}f}$ for f in Figure 1, and it wins if it can invert all t challenges while performing less than t queries to the inversion oracle.

Definition 1 (One-more one-way function [2]). *A one-way function f is* one-more one-way *if it can be computed in polynomial time with respect to the size of its output, and no PPT adversary $\mathcal{A}$ has non-negligible probability of winning the one-more inversion game (informally described above).*

In [2], the authors formalise this definition with the one-way RSA function.

```
$\mathcal{O}^{\mathsf{inv}\text{-}f}(x)$
**if** $y \notin f(X)$ **return** $\perp$
$x \leftarrow f^{-1}(y)$
$q \leftarrow q + 1$
**return** $x$
```

```
$\mathcal{O}^{\mathsf{chal}\text{-}f}(\,)$
$x \leftarrow\!\!\$\ X$
$y \leftarrow f(x)$
$q \leftarrow q + 1$
**return** $y$
```

Figure 1: On the left, the inversion oracle in the one-more one-way inversion game. On the right, the challenge oracle in *alternative* one-more one-way inversion games.

THREE FLAVOURS OF RSA INVERSION. Bellare et al. adapt the notion of one-more one-way functions to the RSA function. Consider the classical RSA assumption, renamed STI-RSA in [2], where a parameter generation algorithm RsaGen with input the security parameter λ returns $N = pq$, the encryption exponent e and the decryption exponent d.

Assumption 2 (RSA Single-Target Inversion [2]). *Let* RsaGen *be the RSA parameter generation algorithm, and let game* $\mathsf{Game}^{\mathsf{RSA\text{-}STI}}_{\mathcal{A},\mathsf{RsaGen}}$ *be as defined in Figure 2. We say that the RSA Single-Target Inversion (RSA-STI) problem is hard if, for any PPT adversary $\mathcal{A}$ playing* $\mathsf{Game}^{\mathsf{RSA\text{-}STI}}_{\mathcal{A},\mathsf{RsaGen}}$*, it holds*

$$\mathsf{Adv}^{\mathsf{RSA\text{-}STI}}_{\mathcal{A},\mathsf{RsaGen}}(\lambda) := \mathbb{P}\left[\mathsf{Game}^{\mathsf{RSA\text{-}STI}}_{\mathcal{A},\mathsf{RsaGen}}(\lambda) = 1\right] = \mathsf{negl}(\lambda) \tag{1}$$

This can be generalised by generating $m+1$ challenges and allowing the adversary to query an RSA inversion oracle (Figure 1) at most m times.

Assumption 3 (RSA Known-Target Inversion [2]). *Let* RsaGen *be the RSA parameter generation algorithm and $m = m(\lambda)$ for λ security parameter. Let game* $\mathsf{Game}^{\mathsf{RSA\text{-}KTI}}_{\mathcal{A},\mathsf{RsaGen}}$ *be as defined in Figure 2. We say that the RSA Known-Target Inversion (RSA-KTI$_m$) problem is hard if, for any PPT adversary $\mathcal{A}$ playing* $\mathsf{Game}^{\mathsf{RSA\text{-}KTI}}_{\mathcal{A},\mathsf{RsaGen}}$*, it holds*

$$\mathsf{Adv}^{\mathsf{RSA\text{-}KTI}}_{\mathcal{A},\mathsf{RsaGen}}(m,\lambda) := \mathbb{P}\left[\mathsf{Game}^{\mathsf{RSA\text{-}KTI}}_{\mathcal{A},\mathsf{RsaGen}}(m,\lambda) = 1\right] = \mathsf{negl}(\lambda) \tag{2}$$

A further generalisation allows the adversary $\mathcal{A}$ to choose which $m+1$ out of n challenges it wishes to invert, with $m < n$. Due to this, we call it "chosen-target". The adversary $\mathcal{A}$ will also provide an injective map $\pi : \{1,\dots,m+1\} \to \{1,\dots,n\}$ to denote which $m+1$ challenges $y_{\pi(1)},\dots,y_{\pi(m+1)}$ it chose as target.

Assumption 4 (RSA Chosen-Target inversion [2]). *Let* RsaGen *be the RSA parameter generation algorithm,* $m = m(\lambda)$ *and* $n = n(\lambda)$ *for* λ *security parameter. Let game* $\mathsf{Game}^{\mathsf{RSA\text{-}CTI}}_{\mathcal{A},\mathsf{RsaGen}}$ *be as defined in Figure 2. We say that the RSA* (n, m)*-Chosen-Target Inversion (RSA-CTI*$_{n,m}$*) problem is hard if, for any PPT adversary* $\mathcal{A}$ *playing* $\mathsf{Game}^{\mathsf{RSA\text{-}CTI}}_{\mathcal{A},\mathsf{RsaGen}}$*, it holds*

$$\mathsf{Adv}^{\mathsf{RSA\text{-}CTI}}_{\mathcal{A},\mathsf{RsaGen}}(n,m,\lambda) \coloneqq \mathbb{P}\left[\mathsf{Game}^{\mathsf{RSA\text{-}CTI}}_{\mathcal{A},\mathsf{RsaGen}}(n,m,\lambda) = 1\right] = \mathsf{negl}(\lambda) \tag{3}$$

```
$\mathsf{Game}^{\mathsf{RSA\text{-}STI}}_{\mathcal{A},\mathsf{RsaGen}}(\lambda)$
$(N, e, d) \leftarrow \mathsf{RsaGen}(1^\lambda)$
$y \leftarrow\!\!\$\ \mathbb{Z}_N$;
$x \leftarrow \mathcal{A}(N, e, \lambda, y)$
return 1 if $x^e = y$
return 0
```

```
$\mathsf{Game}^{\mathsf{RSA\text{-}KTI}}_{\mathcal{A},\mathsf{RsaGen}}(m, \lambda)$
$(N, e, d) \leftarrow \mathsf{RsaGen}(1^\lambda)$
for $i = 1, \dots, m+1$
    $y_i \leftarrow\!\!\$\ \mathbb{Z}_N$;
$\vec{y} \leftarrow (y_1, \dots, y_{m+1})$
$\vec{x} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{inv\text{-}RSA}}}(N, e, \lambda, \vec{y})$
return 1 if $\forall i\ x_i^e = y_i$ and $q \leq m$
return 0
```

```
$\mathsf{Game}^{\mathsf{RSA\text{-}CTI}}_{\mathcal{A},\mathsf{RsaGen}}(n, m, \lambda)$
$(N, e, d) \leftarrow \mathsf{RsaGen}(1^\lambda)$
for $i = 1, \dots, n$
    $y_i \leftarrow\!\!\$\ \mathbb{Z}_N$;
$\vec{y} \leftarrow (y_1, \dots, y_n)$
$(x_1, \dots, x_{m+1}, \pi) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{inv\text{-}RSA}}}(N, e, \lambda, \vec{y})$
return 1 if $\forall i\ x_i^e = y_{\pi(i)}$ and $q \leq m$
return 0
```

Figure 2: Games for the three variants of the RSA assumption: *Single-Target Inversion*, *Known-Target Inversion*, *Chosen-Target Inversion*. The main differences between the one-more variants, RSA-KTI and RSA-CTI, are highlighted in grey.

Lemma 5 ([2]). *RSA-CTI*$_{n,m}$ *is hard if and only if RSA-KTI*$_m$ *is hard.*

Sketch of proof. Consider a reduction $\mathcal{B}$ playing the RSA-CTI$_{n,m}$ game on input $y_1, \dots, y_n$, then $\mathcal{B}$ can just run a PPT adversary $\mathcal{A}$ winning the RSA-KTI$_m$ game on the first $m+1$ points and return the injective map $\pi(i) = i$ for $i = 1, \dots, m+1$.
The other side is harder, we only provide the main idea. Consider a reduction $\mathcal{B}$ playing the RSA-KTI$_m$ game on input the target points $y_1, \dots, y_{m+1}$. First, $\mathcal{B}$ creates randomised targets $y'_1, \dots, y'_n$, where $y'_i = y'_i(y_1, \dots, y_{m+1})$, with the property that inverting them allows to invert the original targets. This is called "reversible embedding", and requires multiplying the y_i elevated by a known exponent, plus a random $(m+2)$-th element of known inverse to ensure the y'_i are uniformly and independently distributed. Then the reduction $\mathcal{B}$ can run some PPT RSA-KTI$_{n,m}$ adversary $\mathcal{A}$, making at most m oracle queries and returning $x'_i = y_i^{d'}$. The reduction builds an exponent matrix (a la index calculus), which is invertible if its determinant is non-zero, otherwise $\mathcal{B}$ returns $\perp$. For a complete proof of this implication, see Sections 3 and 4 of [2]. $\square$

ALTERNATIVE FORMULATIONS: CHALLENGE ORACLE. Finally, Bellare et al. define the *RSA Alternative KTI* (RSA-AKTI$_m$) and the *RSA Alternative CTI* (RSA-ACTI$_{n,m}$) assumptions. Adversaries $\mathcal{A}$ are not given their challenge points by the challenger $\mathcal{C}$, but they can access a challenge oracle $\mathcal{O}^{\mathsf{chal\text{-}RSA}}$ for the RSA function (Figure 1) to dynamically generate them. They prove that the original and alternative formulations are equivalent: RSA-AKTI$_m$ is hard if and only if RSA-KTI$_m$ is hard, and RSA-ACTI$_{n,m}$ is hard if and only if RSA-CTI$_{n,m}$ is hard. For the proof of equivalence see Theorems 5.3 and 5.4 in Section 5 of [2].

3 The One-More Discrete Logarithm assumption

GENERIC GROUP MODEL. The *generic group model* (GGM) acts as a sanity check for group-based games, ensuring that the structure of the algorithm itself does not leak information. Adversaries $\mathcal{A}$ only have access to elements of a group $(\mathbb{G}, \cdot)$ as images of an injective handle function $\Xi : \mathbb{G} \to S$ mapping onto some set S without group structure, and to an implicit group computation oracle $\mathcal{O}_{\mathbb{G}\times}$ to perform the operation $(\Xi(g), \Xi(h)) \mapsto \Xi(gh)$.

DISCRETE LOGARITHM. In [3], Diffie and Hellman ponder the possibility of basing what will be known as *public key cryptosystems* on problems that are computationally hard to invert. ElGamal will formalise the Discrete Logarithm assumption in [4].

Assumption 6 (Discrete Logarithm [3, 4]). *Let* GrGen *be a group generation algorithm, and let game* $\mathsf{Game}^{\mathsf{DL}}_{\mathcal{A},\mathsf{GrGen}}$ *be as defined in Figure 3. We say that the Discrete logarithm (DL) problem is hard if, for any PPT adversary* $\mathcal{A}$ *playing* $\mathsf{Game}^{\mathsf{DL}}_{\mathcal{A},\mathsf{GrGen}}$*, it holds*

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathcal{A},\mathsf{GrGen}}(\lambda) := \mathbb{P}\left[\mathsf{Game}^{\mathsf{DL}}_{\mathcal{A},\mathsf{GrGen}}(\lambda) = 1\right] = \mathsf{negl}(\lambda) \tag{4}$$

```
$\mathsf{Game}^{\mathsf{DL}}_{\mathcal{A},\mathsf{GrGen}}(\lambda)$
$(\mathbb{G}, g, p) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\!\!\$\ \mathbb{Z}_p,\ h \leftarrow g^x$
$y \leftarrow \mathcal{A}((\mathbb{G}, g, p), h)$
$\mathbf{return}\ 1\ \mathbf{if}\ y = x$
$\mathbf{return}\ 0$
```

Figure 3: Discrete Logarithm game.

ONE-MORE DISCRETE LOGARITHM PROBLEM. First formalised by Bellare et al. [2], this assumption allows access to a discrete logarithm oracle $\mathcal{O}^{\mathsf{DL}}$ for as many as t queries and then asks $\mathcal{A}$ to return t discrete logarithms plus "one more", hence the name.

Assumption 7 (One-More Discrete Logarithm [2]). *Let* GrGen *be a group generation algorithm, and let game* $\mathsf{Game}^{\mathsf{OMDL}}_{\mathcal{A},\mathsf{GrGen}}$ *be as defined in Figure 4. We say that the t-One-More Discrete logarithm (OMDL$_t$) problem is hard if, for any PPT adversary* $\mathcal{A}$ *playing* $\mathsf{Game}^{\mathsf{OMDL}}_{\mathcal{A},\mathsf{GrGen}}$*, it holds*

$$\mathsf{Adv}^{\mathsf{OMDL}}_{\mathcal{A},\mathsf{GrGen}}(t, \lambda) := \mathbb{P}\left[\mathsf{Game}^{\mathsf{OMDL}}_{\mathcal{A},\mathsf{GrGen}}(t, \lambda) = 1\right] = \mathsf{negl}(\lambda) \tag{5}$$

The security of OMDL in the GGM is a recent result. The first formal proof of this result was by Coretti et al. [5], later amended and corrected by Bauer et al. in [6]

Lemma 8 ([6]). *Under the GGM, for any adversary* $\mathcal{A}$ *playing* $\mathsf{Game}^{\mathsf{OMDL}}_{\mathcal{A},\mathsf{GrGen}}$*, if* DL *is hard then it holds*

$$\mathsf{Adv}^{\mathsf{OMDL}}_{\mathcal{A},\mathsf{GrGen}}(t, \lambda) \le \mathsf{negl}(\lambda)$$

In literature (e.g. [7]) the victory condition $q \le t$ is sometimes embedded in the oracle as a check: $\mathbf{if}\ q \ge t\ \mathbf{return}\ \perp$. While morally the same, this affects the abort rate in the security proof.

4 The Algebraic One-More Discrete Logarithm assumption

GGM VS AGM. The *algebraic group model* (AGM) is close to the GGM, but exploits no group computation oracle $\mathcal{O}_{\mathbb{G}_\times}$. Instead, it captures the idea that the adversary $\mathcal{A}$ knows how it used all received values $g_1, \dots, g_t$ to compute its current output h. Whenever $\mathcal{A}$ outputs a group element h, it is required to provide its representation with respect to all previous inputs, i.e. a vector $(\alpha_1, \dots, \alpha_t)$ such that

$$h = g_1^{\alpha_1} \dots g_t^{\alpha_t}$$

ALGEBRAIC ONE-MORE DISCRETE LOGARITHM. The Algebraic One-More Discrete Logarithm can be seen as a variant of OMDL where the adversary $\mathcal{A}$ behaves algebraically. It was introduced in [8] by Nick et al. as security assumption for the Schnorr-based multisignature scheme MuSig2.

Assumption 9 (One-More Discrete Logarithm [2]). *Let* GrGen *be a group generation algorithm, and let game* $\mathsf{Game}^{\mathsf{AOMDL}}_{\mathcal{A},\mathsf{GrGen}}$ *be as defined in Figure 4. We say that the Algebraic t-One-More Discrete logarithm (AOMDL$_t$) problem is hard if, for any PPT adversary $\mathcal{A}$ playing* $\mathsf{Game}^{\mathsf{AOMDL}}_{\mathcal{A},\mathsf{GrGen}}$*, it holds*

$$\mathsf{Adv}^{\mathsf{AOMDL}}_{\mathcal{A},\mathsf{GrGen}}(t,\lambda) := \mathbb{P}\left[\mathsf{Game}^{\mathsf{AOMDL}}_{\mathcal{A},\mathsf{GrGen}}(t,\lambda) = 1\right] = \mathsf{negl}(\lambda) \tag{6}$$

```
$\mathsf{Game}^{\mathsf{OMDL}}_{\mathcal{A},\mathsf{GrGen}}(t,\lambda)$
  $(\mathbb{G}, g, p) \leftarrow \mathsf{GrGen}(1^\lambda)$
  $q \leftarrow 0$
  for $i = 1, \dots, t+1$
      $x_i \leftarrow\$ \mathbb{Z}_p,\ h_i \leftarrow g^{x_i}$
  $\vec{x} \leftarrow (x_1, \dots, x_{t+1})$
  $\vec{h} \leftarrow (h_1, \dots, h_{t+1})$
  $\vec{y} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{DL}}}((\mathbb{G}, g, p), \vec{h})$
  return $1$ if $\vec{y} = \vec{x}$
      and $q \le t$
  return $0$

$\mathcal{O}^{\mathsf{DL}}(h)$
  $x \leftarrow \mathsf{dlog}(h)$
  $q \leftarrow q + 1$
  return $x$
```

```
$\mathsf{Game}^{\mathsf{AOMDL}}_{\mathcal{A},\mathsf{GrGen}}(t,\lambda)$
  $(\mathbb{G}, g, p) \leftarrow \mathsf{GrGen}(1^\lambda)$
  $q \leftarrow 0$
  for $i = 1, \dots, t+1$
      $x_i \leftarrow\$ \mathbb{Z}_p,\ h_i \leftarrow g^{x_i}$
  $\vec{x} \leftarrow (x_1, \dots, x_{t+1})$
  $\vec{h} \leftarrow (h_1, \dots, h_{t+1})$
  $\vec{y} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{DL}}}((\mathbb{G}, g, p), \vec{h})$
  return $1$ if $\vec{y} = \vec{x}$
      and $q \le t$
  return $0$

$\mathcal{O}^{\mathsf{DL}}(h, \alpha, (\beta_i)_{i=1,\dots,t+1})$
  $x \leftarrow \alpha + \sum_{i=1}^{t+1} x_i \beta_i$
  $q \leftarrow q + 1$
  return $x$
```

Figure 4: OMDL and AOMDL games. The main differences are highlighted in grey.

The adversary $\mathcal{A}$ only needs to return a representation when querying $\mathcal{O}^{\mathsf{DL}}$, so the only difference from OMDL$_t$ is in the oracle, which now requires (an exploits) the representation of h. Remark how this does not allow $\mathcal{A}$ to bypass the oracle $\mathcal{O}^{\mathsf{DL}}$, as

$$\mathsf{dlog}(h) = \mathsf{dlog}\left(g^\alpha h_1^{\beta_1} \dots h_{t+1}^{\beta_{t+1}}\right) = \mathsf{dlog}\left(g^\alpha g^{x_1\beta_1} \dots g^{x_{t+1}\beta_{t+1}}\right) = \alpha + \sum_{i=1}^{t+1} x_i \beta_i$$

and unless it can break DL, $\mathcal{A}$ has no knowledge of the x_i.
Also notice how requiring the representation of h to be expressed in terms of $g, h_1, \dots, h_{t+1}$ is

non-restrictive. When the game starts, all group elements known by $\mathcal{A}$ are the generator g and the challenge points $h_1 = g^{x_1}, \dots, h_{t+1} = g^{x_{t+1}}$. The AGM implicitly requires $\mathcal{A}$ to show *how* it operated on known elements to return any new h, therefore a representation is always readily available and in terms of $g, h_1, \dots, h_{t+1}$.

Lemma 10. *If OMDL$_t$ is hard, then AOMDL$_t$ is also hard.*

Proof. Consider a PPT adversary $\mathcal{A}$ playing $\mathsf{Game}^{\mathsf{AOMDL}}_{\mathsf{GrGen}}$ for a group generation algorithm GrGen and winning with non-negligible probability. We can exhibit a PPT adversary $\mathcal{B}$ for $\mathsf{Game}^{\mathsf{OMDL}}_{\mathsf{GrGen}}$ (and the same group generation algorithm GrGen) also winning with non-negligible probability by simply dropping the algebraic representation from all oracle queries. $\square$

FALSIFIABILITY. In [9], Gentry and Wichs introduce the notion of *falsifiability* of a cryptographic assumption. Their goal is to capture the confidence we have in a problem, with non-falsifiability implicitly meaning that the problem is hard to reason about.

Definition 11 (Falsifiability [9]). *A cryptographic assumption that can be modeled as an interactive game between a PPT challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is falsifiable if, for any adversary $\mathcal{A}$, the challenger $\mathcal{C}$ can decide in PPT if $\mathcal{A}$ won the game.*

Falsifiable assumptions are e.g. RSA, CDH, DDH, LWE, SVP, DL.

Lemma 12. *OMDL$_t$ is a non-falsifiable cryptographic assumption.*

Proof. A challenger $\mathcal{C}$ trying to simulate the discrete oracle $\mathcal{O}^{\mathsf{DL}}$ for an adversary $\mathcal{A}$ cannot do so in PPT, except if it can already break DL. $\square$

Lemma 13. *AOMDL$_t$ is a falsifiable cryptographic assumption.*

Sketch of proof. The game $\mathsf{Game}^{\mathsf{AOMDL}}_{\mathcal{A},\mathsf{GrGen}}$ of Figure 4 requires $\mathcal{A}$ to provide an algebraic representation $(\alpha, (\beta_i)_{i=1,\dots,t})$ of all queried h relative to the generator g and the challenges $h_1, \dots, h_{t+1}$. This allows the oracle $\mathcal{O}^{\mathsf{DL}}$ to answer in PPT by returning $\alpha + \sum_i x_i \beta_i$ instead of $\mathsf{dlog}(h)$. $\square$

5 Use cases

Many protocols consider one-more security assumptions: blind signatures, multisignatures, threshold signatures, identification protocols, etc. We will focus on blind and threshold signatures.

BLIND SIGNATURES: CHAUM. Blind signatures allow users to obtain a signature of a message m from a signing party without it learning m. Bellare et al. [2] introduce One-More RSA to prove the security of Chaum's blind variant of the FDH-RSA signature protocol.

Definition 14 (RSA Single-Target Inversion [2]). *Let* RsaGen *be the RSA parameter generation algorithm, and let game* $\mathsf{Game}^{\mathsf{RSA\text{-}OMF}}_{\mathcal{A},\mathsf{RsaGen}}$ *be as defined in Figure 5. We say that the blind FDH-RSA signature is one-more forgery secure if, for any PPT adversary $\mathcal{A}$ playing* $\mathsf{Game}^{\mathsf{RSA\text{-}OMF}}_{\mathcal{A},\mathsf{RsaGen}}$, *it holds*

$$\mathsf{Adv}^{\mathsf{RSA\text{-}OMF}}_{\mathcal{A},\mathsf{RsaGen}}(\lambda) := \mathbb{P}\left[\mathsf{Game}^{\mathsf{RSA\text{-}OMF}}_{\mathcal{A},\mathsf{RsaGen}}(\lambda) = 1\right] = \mathsf{negl}(\lambda) \tag{7}$$

Blind FDH-RSA

$$\begin{array}{lcl}
\mathsf{User}(N, e, \mathsf{m}) & & \mathsf{Signer}(N, d) \\
r \leftarrow\!\!\$\ \mathbb{Z}_n^* & & \\
\mathsf{m}' \leftarrow r^e \mathsf{H}(\mathsf{m}) & \xrightarrow{\mathsf{m}'} & \\
 & \xleftarrow{x'} & x' \leftarrow (\mathsf{m}')^d \mod N \\
x \leftarrow r^{-1} x' & &
\end{array}$$

```
$\mathsf{Game}^{\mathsf{RSA\text{-}OMF}}_{\mathcal{A},\mathsf{RsaGen}}(\lambda)$
$(N, e, d) \leftarrow \mathsf{RsaGen}(1^\lambda)$
$\mathsf{H} \leftarrow\!\!\$\ \{f : \{0,1\}^* \to \mathbb{Z}_N^*\}$
$((\mathsf{m}_1, x_1), \dots, (\mathsf{m}_{m+1}, x_{m+1})) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{inv\text{-}RSA}}}(N, e, \lambda)$
**return** 1 **if** $\mathsf{m}_1 \neq \mathsf{m}_2 \neq \ldots \neq \mathsf{m}_{m+1}$ and
  less than $m+1$ queries to $\mathcal{O}^{\mathsf{inv\text{-}RSA}}$ and
  $\forall i \quad \mathsf{H}(\mathsf{m}_i) \equiv x_i^e \mod N$
**return** 0
```

Figure 5: On the left, Chaum's blind FDH-RSA signature. On the right, its one-more forgery game.

Theorem 15 ([2]). *For any forger $\mathcal{F}$ attacking the RSA-OMF security of the FDH-RSA blind signature scheme in Figure 5, there exists a PPT adversary $\mathcal{A}$ against the RSA-ACTI$_{n,m}$ assumption such that*

$$\mathsf{Adv}^{\mathsf{RSA\text{-}OMF}}_{\mathcal{F},\mathsf{RsaGen}}(\lambda) \leq \mathsf{Adv}^{\mathsf{RSA\text{-}ACTI}}_{\mathcal{A},\mathsf{RsaGen}}(n, m, \lambda) \tag{8}$$

Sketch of proof. Consider a forger $\mathcal{F}$ able to break the RSA-OMF security of Chaum's signature. We can exhibit an adversary $\mathcal{A}$ against the RSA-ACTI$_{n,m}$ assumption using $\mathcal{F}$ as a subroutine. When $\mathcal{F}$ queries H for $\mathsf{H}(\mathsf{m})$, then $\mathcal{A}$ queries its challenge oracle and sets $\mathsf{H}(\mathsf{m}) \leftarrow \mathcal{O}^{\mathsf{chal\text{-}RSA}}$. When $\mathcal{A}$ queries $\mathcal{O}^{\mathsf{inv\text{-}RSA}}$ instead, then $\mathcal{A}$ truthfully queries its own $\mathcal{O}^{\mathsf{inv\text{-}RSA}}$. The adversary will save all interactions with $\mathcal{F}$ in hash, challenge, message, response arrays, and construct π by searching for element indices. Finally, $\mathcal{A}$ will need to adjust for values m_i that $\mathcal{F}$ returned by itself, i.e. those such that $\mathsf{H}(\mathsf{m}_i) = \perp$ in its array. During the presentation, we will read the proof of Lemma 6.4 of [2] for a full description of the security proof. □

<u>THRESHOLD SIGNATURES: SPARKLE.</u> Threshold signature schemes allow to distribute the signing authority among n parties so that only subsets of (at least) t of them can sign messages m. The security of the threshold Schnorr variant Sparkle, by Crites et al. [7], is based on the AOMDL assumption. An important step is their proof of adaptive threshold EUF-CMA without the forking lemma; this allows to avoid the exponential tightness loss. Due to a lack of time, we refer interested readers to [7] for the first version of the article, with a single-game proof, and [10], with a sequence-of-games proof.

Theorem 16 ([7]). *For any forger $\mathcal{F}$ attacking the adp-TS-EUF-CMA security of* Sparkle *threshold signature scheme, there exists a PPT adversary $\mathcal{A}$ against the AOMDL$_t$ assumption such that*

$$\mathsf{Adv}^{\mathsf{adp\text{-}TS\text{-}EUF\text{-}CMA}}_{\mathcal{F},\mathsf{GrGen}}(\lambda, \tau = 1) \leq \mathsf{Adv}^{t-\mathsf{AOMDL}}_{\mathcal{A},\mathsf{GrGen}}(\lambda) + \mathsf{negl}(\lambda) \tag{9}$$

Sources and research material

This report is based on the following publications.

[1] *First formalisation of One-More forgeries.* Pointcheval and Stern. *Provably secure blind signature schemes*. `ieeexplore.ieee.org/document/10158918/`. 1996.

[2] *First formalisation of OMRSAI and OMDL*. Bellare, Namprempre, Pointcheval, and Semanko. *The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme*. `https://eprint.iacr.org/2001/002`. 2003.

[3] *First use of the Discrete Logarithm problem.* Diffie and Hellman. *New Directions in cryptography*. `https://ieeexplore.ieee.org/document/1055638`. 1976.

[4] *First formalisation of the Discrete Logarithm problem.* ElGamal. *A public key cryptosystem and a signature scheme based on discrete logarithms*. `https://ieeexplore.ieee.org/document/1057074/`. 1985.

[5] *Sketch of proof of the security of OMDL in the GGM*. Coretti, Dodis, and Guo. *Non-Uniform Bounds in the Random-Permutation, Ideal-Cipher, and Generic-Group Models*. `https://eprint.iacr.org/2018/226`. 2018.

[6] *Amended and corrected the proof in [5]*. Bauer, Fuchsbauer, and Plouviez. *The One-More Discrete Logarithm Assumption in the Generic Group Model*. `https://eprint.iacr.org/2021/866`. 2021.

[7] *Version 1 of Sparkle, the security of this threshold signature is based on the AOMDL assumption*. Crites, Komlo, and Maller. *Fully Adaptive Schnorr Threshold Signatures*. `https://eprint.iacr.org/archive/2023/445/20230327:090418`.

[8] *Introduction of AOMDL*. Nick, Ruffing, and Seurin. *MuSig2: Simple two-round Schnorr multi-signatures*. `https://eprint.iacr.org/2020/1261`. 2021.

[9] *Definition of falsifiable problem*. Gentry and Wichs. *Separating succinct non-interactive arguments from all falsifiable assumptions*. `https://eprint.iacr.org/2010/610`. 2011.

[10] *Latest version of Sparkle, with a sequence-of-games security proof.* Crites, Komlo, and Maller. *Fully Adaptive Schnorr Threshold Signatures*. `https://eprint.iacr.org/archive/2023/445/20250603:205454`.