Digital signatures lecture notes

Luca Tarquini

June 2022

Contents

Chapter 1

Introduction

Signatures are the digital analogue of physical signatures. Properties we want are **authenticity** (document is signed **by that** person) and **integrity** (document has **not been changed** since signing).

Applications of signatures are code apps, certificates, identity cards, etc.

A digital signature scheme is a tuple $\Sigma = (Gen, Sign, Vfy)$ of 3 probabilistic polynomial-time (**PPT**) algorithms:

- $Gen(1^k) \rightarrow (pk, sk)$
- $Sign(sk, m) \rightarrow \sigma$
- $Vfy(pk, m, \sigma) \rightarrow \{0, 1\}$

$m \in \{0,1\}^*$, $k \in \mathbb{N}$ is the **security parameter** (length of the key).

Correctness: $\forall k, m \in \{0,1\}^*, \forall (pk, sk) \leftarrow Gen(1^k) : Vfy(pk, m, Sign(sk, m)) = 1$

Concrete security definitions combine two things: adversarial **goals** and adversary **capabilities**.
Capabilities:

- No-message attack (NMA): adversary gets **only** pk.
- Non-Adaptive chosen message attack (naCMA): adverary chooses $m_1, ..., m_q$, **then** obtains pk and signatures $\sigma_1, ..., \sigma_n$.
- Chosen message attack (CMA): adverary gets pk, **then** chooses one message at a time (adaptively) and gets the corresponding signature.

Goals:

- Universal Unforgeability (UUF): adversary has to forge signature for a message m **externally** given (chosen at random).
- Existential Unforgeability (EUF): adversary forges signature for **any** random message m not signed before.

A security experiment is a tool to formalize security definitions. Interactive process between two parties: adversary $\mathcal{A}$ and challenger $\mathcal{C}$.

EUF-CMA security experiment: $\mathcal{A}$ wins iff $Vfy(pk, m^*, \sigma^*) = 1 \wedge m^* \notin \{m_1, ..., m_q\}$, where q is a polynomial number of queries in the security parameter.

A protocol is **secure** in a security definition iff $Pr[\mathcal{A}$ wins related experiment] is negligible, that is a function `negl`: $\mathbb{N} \rightarrow \mathbb{N}$ such that: $\exists k_0, \forall c \in \mathbb{N}, \forall k > k_0, |negl(k)| < 1/k^c$.

If Σ is EUF-CMA secure, then Σ is also UUF-CMA secure. Assume $\mathcal{A}$ can break UUF-CMA, construct $\mathcal{B}$ that plays EUF-CMA experiment.
$\mathcal{C}$ sends pk to $\mathcal{B}$ (and other messages $m_1, ..., m_q$).
$\mathcal{B}$ chooses m^* and sends pk and m^* to $\mathcal{A}$.
$\mathcal{A}$ sends back (m^*, σ^*).
Whenever $\mathcal{A}$ succeeds, so does $\mathcal{B}$.

Strength of security definitions:
UUF $<$ EUF
NMA $<$ naCMA $<$ CMA

Information-theoretic security (cannot ever be broken) is NOT possible for digital signatures:

- There exists an **unbounded** adversary $\mathcal{A}$ with success probability 1 (brute-force).
- There exists a **PPT** adversary $\mathcal{A}$ with success probability $\frac{1}{2^L}$ (guessing) $\Rightarrow$ this is why we allow a negligible (and not zero) probability of success secure in our sense.

Message space extension It is usually easier to construct signatures with small message space, e.g., $\{0,1\}^{q(k)}$, q polynomial in the security parameter.
A cryprographic **hash function** $H = (Gen_H, Eval_H)$ consists of two PPT algorithms:

- $Gen_H(1^k)$ outputs a parameter t that defines a function $H_t : \{0,1\}^* \to \mathcal{M}_t$.
- $Eval_H(1^k, t, x)$ computes $H_t(x)$.

A hash function H is **collision-resistant** (CR) iff for $t \leftarrow Gen_H(1^k)$ and all PPT $\mathcal{A}$: $\Pr[\mathcal{A}(1^k, t) = (x, x') : H_t(x) = H_t(x') \wedge x \neq x']$ is negligible.
Unbounded message space signatures: given $\Sigma' = (Gen', Sign', Vfy')$ with message space $\mathcal{M}$, and CR hash $H : \{0,1\}^* \to \mathcal{M}$, construct $\Sigma = (Gen, Sign, Vfy)$ with:

- $Gen(1^k) \to (pk, sk) = ((pk', t), (sk', t))$
- $Sign((sk', t), m) = Sign'(sk', H_t(m)) \to \sigma$
- $Vfy((pk', t), m, \sigma) = Vfy'(pk', H_t(m), \sigma)$

Theorem: for every EUF-CMA PPT $\mathcal{A}$ on Σ there exist a EUF-CMA PPT $\mathcal{B}$ on Σ' and a PPT $\mathcal{C}$ on collision-resistance of H, such that $\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}}$ and roughly same runtimes.
Whenever $\mathcal{A}$ forges a signature for Σ (which is also valid for Σ'), either:

- $H(m)$ has been signed before, but $m \neq m_i \Rightarrow$ found collision.
- $H(m)$ has never been signed before in Σ'.

This construction is called **Hash-then-Sign**.

Chapter 2

One-time signatures

Signature schemes that can securely sign only one message. EUF-1-CMA and EUF-1-naCMA security experiments are introduced.
These signatures are easy to construct and represent an important building block.

One-way functions $f : \{0,1\}^* \to \{0,1\}^*$
Given x, easy (polynomial time in the size) to compute $f(x)$.
Given y, hard (negligible probability) to compute **any** x' in $f^{-1}(y)$.

Theorem: if one-way functions exist, then $\mathcal{P} \neq \mathcal{NP}$.
Proof: Consider the language $\mathcal{L}_f := \{(y, \bar{x}, 1^l) | \exists x \in \{0,1\}^l$ with prefix $\bar{x}$ and with $f(x) = y\}$.
Then $\mathcal{L}_f \in \mathcal{NP}$ since x is a witness for a word in the language.
But if $\mathcal{L}_f \in \mathcal{P}$, then $\exists$ poly-time algorithm D that decides $\mathcal{L}_f$. This D can be used to invert f by querying it bit-by-bit.
Realistically, this implies that constructions of one-way functions require **assumptions** (such as RSA, DLog, etc).

Another theorem justifying the use of assumptions is the following:
If Σ is a UUF-NMA secure signature scheme, then $\mathcal{P} \neq \mathcal{NP}$.
Take the restriction of Gen s.t. Gen' outputs only pk. Then Gen' is a one-way function. If it is not, the scheme can be broken. If one-way functions exist, $\mathcal{P} \neq \mathcal{NP}$.

Uselessness of UUF-NMA A UUF-NMA secure scheme can be built with **message-independent signatures**, with the help of a one-way function f.

- $Gen(1^k) : sk \leftarrow \{0,1\}^*, pk = f(sk)$.
- $Sign(sk, m) = sk$
- $Vfy(pk, m, \sigma) : f(\sigma) = pk$

2.1 Lamport's one-time signatures

- $Gen(1^k)$: choose $x_{1,0}, x_{1,1}, ..., x_{n,0}, x_{n,1}$ uniformly from $\{0,1\}^k$, and $y_{j,i} := f(x_{j,i})$.

$$sk = \begin{pmatrix} x_{1,0} & ... & x_{n,0} \\ x_{1,1} & ... & x_{n,1} \end{pmatrix} \tag{2.1}$$

$$pk = \begin{pmatrix} y_{1,0} & ... & y_{n,0} \\ y_{1,1} & ... & y_{n,1} \end{pmatrix} \tag{2.2}$$

- $Sign(sk, m)$: $\sigma = (x_{1,m_1}, x_{2,m_2}, ..., x_{n,m_n})$
- $Vfy(pk, m, \sigma)$: check that for all $i \in \{1, ..., n\}$, we have: $f(x_i^\sigma) = y_{i,m_i}$

Theorem: for every EUF-1-naCMA PPT $\mathcal{A}$, there is a PPT $\mathcal{B}$ on f with rougly same runtime and success probability $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{n}$.
Proof: $\mathcal{B}$ embeds $y = f(x)$ into pk as $y_{i,1-m_i}$ in his forgery (so that it is not used to sign the chosen message, need to know the preimage). Chooses other $x_{i,b}$ randomly. The probability that the embedded bit is forged is at least $\frac{1}{n}$ (single bit modified, forged message must be different).
Hence, $\mathcal{B}$ wins with probability $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{n}$.

Actually, this scheme is **EUF-1-CMA** secure.
This scheme is not very efficient though, as it requires **many evaluations** of one-way function and **large** keys.

2.2 Discrete-Log-based one-time signatures

Given a cyclic group $\mathbb{G}$ of prime order p, and a generator g, the DLog problem is stated as follows:
Given g and $y \leftarrow \mathbb{G}$, find $x \in \mathbb{Z}_p$ with $g^x = y$.

The DLog assumption states that for all PPT adversaries, the probability of solving the DLog problem is **negligible**.

Signature scheme:

- $Gen(1^k) : x \leftarrow \mathbb{Z}_p^*$, $\omega \leftarrow \mathbb{Z}_p$, $h := g^x$, $c := g^\omega \Rightarrow pk = (g, h, c), sk = (x, \omega)$
- $Sign(sk, m) : \sigma = \frac{\omega - m}{x}$
- $Vfy(pk, m, \sigma) : c = g^m h^\sigma$

This scheme is **EUF-1-naCMA** secure: for every EUF-1-naCMA PPT $\mathcal{A}$, there exists a PPT $\mathcal{B}$ with roughly same runtime and success probability $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$ that solves the DLog problem.
Proof: Choose σ randomly, compute $c = g^m h^\sigma$ so that σ is valid and send the pk. Upon receipt of (m^*, σ^*), two signing equations for two unknowns x, ω. Extract $x = \frac{m^* - m}{\sigma - \sigma^*}$.

EUF-CMA attack $g^{m_1} h^{\sigma_1} = c = g^{m_2} h^{\sigma_2} \Rightarrow m_1 + x\sigma_1 = m_2 + x\sigma_2$.
Find x, then compute ω as $g^\omega = c = g^m g^{x\sigma}$, secret key is then found.

2.3 RSA-based one-time signatures

Given $N = P \cdot Q$, where P, Q are large (> 1000 bits) prime numbers, $\phi(N) = (P-1) \cdot (Q-1) = |\mathbb{Z}_N^*|$ (totient, order of multiplicative invertible elements), choose $e \in \mathbb{N}$ uniformly between 1 and $\phi(N)$ with $gcd(e, \phi(N)) = 1$.
Then $d \in \mathbb{N}$ with $e \cdot d = 1 \mod \phi(N)$ can be found efficiently.
For $x \in \mathbb{Z}_N$, we have $x^{ed} = x \mod N$.

The RSA problem is stated as follows:
Given N, e and $y \leftarrow \mathbb{Z}_N$, find $x \in \mathbb{Z}_N$ with $x^e = y \mod N$.
The RSA assumption states that for all PPT adversaries, the probability of solving the RSA problem is **negligible**.

Signature scheme: (finite space 2^n)

- $Gen(1^k)$: Choose P, Q, choose **prime** e with $2^n < e < \phi(N)$, with $gcd(e, \phi(N)) = 1$.
  Choose $d = e^{-1} \mod \phi(N)$, $J, c \leftarrow \mathbb{Z}_N$.
  $pk = (N, e, J, c), sk = d$.
- $Sign(sk, m) : \sigma = (\frac{c}{J^m})^d \mod N$.
- $Vfy(pk, m, \sigma) : c = J^m \sigma^e \mod N$.

Correctness is trivially verified.

In this scheme the **prime-e-RSA assumption** is used, which is implied asymptotically by the RSA assumption.
The motivation is to have a cleaner reduction without security losses.

Shamir's trick Let $J, S \in \mathbb{Z}_N^*$ and $e, f \in \mathbb{Z}$, with:

- $gcd(e, f) = 1$.
- $J^f = S^e \mod N$.

Then, there is an algorithm that efficiently computes $x \in \mathbb{Z}_N$ with $x^e = J \mod N$.
Proof sketch: set $x = J^\alpha S^\beta$,[1] with $\alpha e + \beta f = 1$, since $gcd(e, f) = 1$.

Theorem: for every **EUF-1-naCMA** PPT $\mathcal{A}$, there exists a PPT $\mathcal{B}$ that solves the prime-e-RSA problem with roughly same runtime and success probability $\epsilon_\mathcal{B} \geq \epsilon_\mathcal{A}$.
Proof: set $J = y$, choose σ randomly and compute $c = J^m \sigma^e (\mod N)$ so that σ is valid.
Upon receipt of (m^*, σ^*), $J^m \sigma^e = c = J^{m^*}(\sigma^*)^e \Rightarrow J^{m-m^*} = (\frac{\sigma^*}{\sigma})^e$.
Since $0 < m - m^* < 2^n$, and $e > 2^n$ prime, then $gcd(e, m - m^*) = 1 \rightarrow$ Shamir's trick finds x with $x^e = J = y$.

EUF-CMA attack $J^{m_1}\sigma_1^e = c = J^{m_2}\sigma_2^e (\mod N) \Rightarrow J^{m_1 - m_2} = (\frac{\sigma_2}{\sigma_1})^e$.
Shamir's trick yields x with $x^e = J \mod N$.
Given x, m, σ and any m^*, can forge $\sigma^* := \sigma x^{m-m^*}$.

2.4 Many-time signatures from one-time signatures

EUF-CMA from EUF-naCMA and EUF-1-naCMA An EUF-CMA secure signature scheme Σ can be constructed from an EUF-naCMA secure scheme Σ' with the help of an EUF-1-naCMA secure one-time signature scheme $\Sigma^{(1)}$:

- $Gen(1^k) = Gen'(1^k) \Rightarrow (pk, sk) = (pk', sk')$
- $Sign(sk, m) : \sigma = (pk^{(1)}, \sigma^{(1)}, \sigma')$, where:
  $pk^{(1)}$ is a fresh new key from $Gen^{(1)}(1^k)$
  $\sigma^{(1)} \leftarrow Sign^{(1)}(sk^{(1)}, m)$ (signs the actual message)
  $\sigma' \leftarrow Sign'(sk, pk^{(1)})$ (binds $pk^{(1)}$ to sk)
- $Vfy(pk, m, \sigma) = Vfy'(pk, pk^{(1)}, \sigma') \wedge Vfy^{(1)}(pk^{(1)}, m, \sigma^{(1)})$

Theorem: for every PPT $\mathcal{A}$ that breaks EUF-CMA in Σ with at most q queries, there exist PPTs $\mathcal{B}$ and $\mathcal{C}$ with roughly same runtimes, and:

- $\mathcal{B}$ breaks EUF-1-naCMA in $\Sigma^{(1)}$ with probability ϵ_B
- $\mathcal{C}$ breaks EUF-naCMA in Σ' with probability ϵ_C

such that: $q \cdot \epsilon_B + \epsilon_C \geq \epsilon_A$.
In particular, if ϵ_A is non-negligible, then so is ϵ_B **or** ϵ_C.
Proof:

- Reduction to security of Σ': $\sigma_i = (pk_i^{(1)}, \sigma_i^{(1)}, \sigma_i')$. $\mathcal{C}$ wins iff $\mathcal{A}$ outputs valid forgery (m^*, σ^*) **and** $pk_*^{(1)} \notin \{pk_i^{(1)}\}$.
  $\epsilon_C = Pr[\mathcal{A} \text{ wins} \wedge pk_*^{(1)} \notin \{pk_i^{(1)}\}]$.

[1] The root x is of this form.

- Reduction to security of $\Sigma^{(1)}$: choose pk', sk' and i^*, relay to such challenger only if $i = i^*$, otherwise choose randomly. Compute σ_i using sk' and $\sigma_{i^*}^{(1)}$. Reduction wins iff $\sigma_*^{(1)}$ is valid ($\mathcal{A}$ outputs valid forgery) and $pk_*^{(1)}$ was used before (especially in i^*).

Whenever $\mathcal{A}$ wins, either $pk_*^{(1)}$ was used before or not. Let $p := Pr[pk_*^{(1)} \in \{pk_i^{(1)}\} | \mathcal{A} \text{ wins}]$:

- $\epsilon_B \geq \epsilon_A \cdot \frac{1}{q} \cdot p$.
- $\epsilon_C \geq \epsilon_A \cdot (1 - p)$.

Result is: $q \cdot \epsilon_B + \epsilon_C \geq \epsilon_A$

The following schemes are **EUF-q-CMA** secure (whether the needed hash functions are collision-resistant).

Naive approach Use q keypairs for q desired signature:

- $Gen(1^k) \to pk := (pk_1, ..., pk_q)$, $sk := (sk_1, ..., sk_q, state = 1)$
- $Sign(sk, m) : \forall st, \sigma_{st} \leftarrow Sign^{(1)}(sk_{st}, m)$, $\sigma := (\sigma_{st}, state)$, $state = state + 1$
- $Vfy(pk, m, (\sigma_i, i)) = Vfy^{(1)}(pk_i, m, \sigma_i), \forall i$

Complexity: $|pk| \in \mathcal{O}(q)$, $|sk| \in \mathcal{O}(q)$, $|\sigma| \in \mathcal{O}(1)$

Intermediate approach Use a hash function to compress the keys:

- $Gen(1^k) \to pk := (H, H(pk_1, ..., pk_q))$, $sk := (sk_1, ..., sk_q, pk_1, ..., pk_q, state = 1)$
- $Sign(sk, m) : \forall st, \sigma_{st} \leftarrow Sign^{(1)}(sk_{st}, m)$, $\sigma := (\sigma_{st}, state, pk_1, ..., pk_q)$, $state = state + 1$
- $Vfy(pk, m, (\sigma_i, i)) = Vfy^{(1)}(pk_i, m, \sigma_i) \wedge H(pk_1, ..., pk_q) == pk, \forall i$

Complexity: $|pk| \in \mathcal{O}(1)$, $|sk| \in \mathcal{O}(q)$, $|\sigma| \in \mathcal{O}(q)$

Merkle trees Use a Merkle Tree to compress the size of the signature:

- $Gen(1^k) \to pk := (H, hash\_root(pk_1, ..., pk_q))$, $sk := (sk_1, ..., sk_q, pk_1, ..., pk_q, state = 1)$
- $Sign(sk, m) : \forall st, \sigma_{st} \leftarrow Sign^{(1)}(sk_{st}, m)$, $\sigma := (\sigma_{st}, state, pk_i, co\_path)$, $state = state + 1$
- $Vfy(pk, m, (\sigma_i, i)) = Vfy^{(1)}(pk_i, m, \sigma_i) \wedge hash\_root == pk, \forall i$

where the **co-path**, given a starting vertex v, is the set of vertices $u_1, ..., u_n$, such that u_i is the sibling of the i-th vertex on the path from v to the root.
Complexity: $|pk| \in \mathcal{O}(1)$, $|sk| \in \mathcal{O}(q)$, $|\sigma| \in \mathcal{O}(log(q))$

Compressing the secret key Choose keypairs through a **pseudo-random function**. Need to store only the seed.

However, when signing, need to recompute all relevant keys (**slow**).
Complexity: $|pk| \in \mathcal{O}(1)$, $|sk| \in \mathcal{O}(1)$, $|\sigma| \in \mathcal{O}(log(q))$

Decreasing generation (and signing?) time So far, key generation and signing took $\mathcal{O}(q)$.
A solution is to build an entire tree where nodes are one-time-signatures keypairs. Every key signs the public keys of the two child nodes.
This tree can be built **lazily**.

Chapter 3

RSA-based signatures

Considering the RSA assumption, it is possible to construct more efficient signature schemes without relying on one-way functions only.

3.1 Textbook RSA

A first step towards this approach is given by the following (insecure) scheme:

- $Gen(1^k) : pk = (N, e), sk = (N, d)$, where N, e, d are chosen consistently with the RSA problem.
- $Sign(sk, m) : \sigma = m^d \mod N$
- $Vfy(pk, m, \sigma) : m = \sigma^e \mod N$

This scheme is UUF-NMA secure if the RSA assumption holds, but **NOT EUF-NMA** secure (choose signature first, extract m; multiplicative homomorphic).

In order to build more secure schemes, suitable **preprocessing** is performed on the message m before signing:

3.2 RSA PKCS #1 v1.5

Let H be a collision-resistant hash function:

- $Gen(1^k)$: as with textbook RSA.
- $Sign(sk, m) : m' := 0x00||0x01||0xFF||...||0xFF||0x00||spec.H||H(m)$, $\sigma' = (m')^d \mod N$
- $Vfy(pk, m, \sigma) : m' = \sigma^e \mod N$, check if m' is a valid encoding of m.

Security of this scheme is not clear: **no attacks known**, but also **no security proof**.
However, it is not trivially multiplicative homomorphic.

3.3 RSA-FDH

By demanding a suitable (but very strong) security requirement for a hash function $H : \{0,1\}^* \to \mathbb{Z}_N$, namely **collision-resistance**, it can be shown that this eliminates the weaknesses incurred by the homomorphism of the textbook RSA scheme. It is essential that H covers **all** of $\mathbb{Z}_N$, without being the identity (otherwise multiplicative homomorphic).

- $Gen(1^k)$: as with textbook RSA.
- $Sign(sk, m) : \sigma := H(m)^d \mod N$
- $Vfy(pk, m, \sigma) : \sigma^e \mod N = H(m)$.

Correctness is trivially satisfied.

3.3.1 The random oracle model

The random oracle model (ROM) assumes hash functions to be idealized, outputting **truly random** values as black boxes. For any input m, the function returns a unique (the same on the same input), random value $H(m)$. There is **no "real" attack** for those functions in such a model.
All parties use the same oracle.
It is **impossible** for any real hash function to implement ROM.

This idealization has the sole purpose to execute security proofs with **stronger assumptions**. A proof in the standard model has better outcomes than a proof in the ROM model, but some problems are solvable **only** in ROM.
It is **unclear** what a ROM security proof implies in practice.

3.3.2 Security proof of RSA-FDH in the ROM

In the following we will assume that for **every** signing query, the adversary $\mathcal{A}$ has previously queried $H(m_i)$, as well as the forged $H(m^*)$.
This assumption is not restrictive as other adversaries can be trivially reduced to those without affecting runtime (ROM is efficient).

Theorem: if H is modeled as the random oracle, then for every PPT $\mathcal{A}$ that breaks **EUF-CMA** security of RSA-FDH with at most q_H queries to the ROM H, there exists a PPT $\mathcal{B}$ with roughly same runtime that solves the RSA problem with success probability:

$$\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{q_H}. \tag{3.1}$$

Proof: $\mathcal{B}$ simulates also the RO for $\mathcal{A}$. Simulation works as follows: set $H(m_i) = x_i^e \mod N$ for randomly sampled x_i, such that signature is known, except for i^*, where $H(m_{i^*}) = y$, where y is given by the RSA challenge.[1]
Hope that forged m^* comes from the i^* query. This way, extraction and reduction is possible:
$\mathcal{B}$ wins whenever $\mathcal{A}$ wins and $m^* = m_{i^*}$. Thus, $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{q_H}$.

The q_H factor in the reduction makes it **lossy**, with an influence on the security parameters:
Relying on the assumption that the best known factorization algorithm for RSA is GNFS, with success probability 1 and runtime $t_{GNFS} := C \cdot exp((\frac{64}{9})^{\frac{1}{3}} n^{\frac{1}{3}} ln(n)^{\frac{2}{3}})$, the existence of adversaries $\mathcal{A}$ and $\mathcal{B}$ should contradict it.
$\mathcal{B}$ wins with probability $\epsilon_{\mathcal{B}}$ and runtime $t_{\mathcal{B}}$. The Las Vegas algorithm derived from it, which achieves success probability 1 (i.e., repeat subroutine until correct), has expected runtime $\frac{1}{\epsilon_{\mathcal{B}}} \cdot t_{\mathcal{B}}$.
Now, $\frac{1}{\epsilon_{\mathcal{B}}} \cdot t_{\mathcal{B}} \leq \frac{q_H}{\epsilon_{\mathcal{A}}} \approx \frac{q_H}{\epsilon_{\mathcal{A}}} \cdot t_{\mathcal{A}}$.

To contradict the GNFS assumption it must be: $t_{GNFS}(n) > \frac{q_H}{\epsilon_{\mathcal{A}}} \cdot t_{\mathcal{A}}$, $n = \lfloor log_2(N) \rfloor + 1$.
This implies that the larger the number of hash queries, the greater the RSA parameters must be for the theorem to hold.

Reduction loss can be decreased up to $\mathcal{O}(q_s)$, where q_s is the number of signing queries, with a better strategy in the proof (see Exercises).

3.4 RSA-PSS

Preprocessing in this scheme is a bit more complex:

- $Gen(1^k)$: as with textbook RSA.
- $Sign(sk, m) : \sigma := PSS\_Encode(m)^d \mod N$
- $Vfy(pk, m, \sigma) : y = \sigma^e \mod N$, check if y is a valid PSS encoding of m.

[1] A reduction can program the random oracle to any value of its choice, as long as it's uniformly distributed.

PSS Encoding Given k_0, k_1 s.t. $k_0 + k_1 \leq k - 1$, **two** hash functions $H : \{0,1\}^* \to \{0,1\}^{k_1}$, $G : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1-1}$, G can be split into: $\forall w \in \{0,1\}^{k_1} : G(w) = G_1(w)||G_2(w)$.
G_1 : first k_0 bits of G.
G_2 : rest of G.
Encoding works as follows:

- choose $r \leftarrow \{0,1\}^{k_0}$ uniformly
- $w := H(m||r)$
- $r^* := G_1(w) \oplus r$
- $\gamma := G_2(w)$

encoding $:= 0||w||r^*||\gamma$

Verification first checks if the first bit is 0, then splits $y = \sigma^e \mod N$ into $0, w', r^{*'}, \gamma'$, computes $r' = r^{*'} \oplus G_1(w')$ and outputs 1 iff $\gamma' = G_2(w') \wedge w' = H(m||r')$. A graphic illustration of the algorithm is shown below.

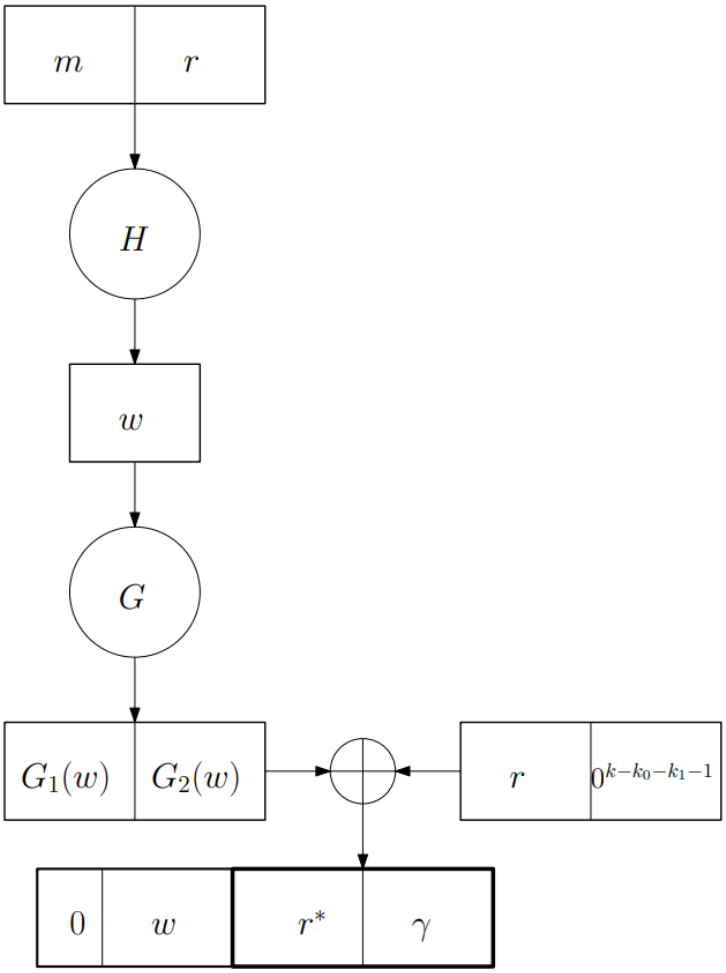

Figure 3.1: Illustration of the PSS encoding algorithm.

Theorem: Assume G, H **random oracles**. Then for every PPT $\mathcal{A}$ that breaks the EUF-CMA security of RSA-PSS in time $t_{\mathcal{A}}$, with at most q_H hash queries and q_s signature queries and success probability

$\epsilon_{\mathcal{A}}$, there exists a PPT $\mathcal{B}$ that solves the RSA problem in time $t_{\mathcal{B}}$ with success probability and runtime:

$$\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}} - (2(q_s + q_H)^2 + 1) \cdot (2^{-k_0} + 2^{-k_1}) \tag{3.2}$$

$$t_{\mathcal{B}} \leq t_{\mathcal{A}} + (q_s + q_h + 1) \cdot k_0 \cdot \Theta(n^3). \tag{3.3}$$

Proof sketch: embed RSA challenge in **every** hash query. Upon signature queries, choose fresh new r randomly with known signature $\rightarrow$ multiple encodings for the same message m. This way $\mathcal{A}$ always solves the RSA problem (no guess).

This scheme is in practice more efficient as the reduction is **less lossy** than RSA-FDH, even though two hash queries are required per every signature.

3.5 GHR signatures

So far, RSA schemes are either inefficient or rely on heuristic security (ROM). In order to construct EUF-CMA secure schemes in the **standard** model based on RSA, there is the following workaround, which **strengthens** the underlying assumption:

Strong RSA problem Given N, $y \leftarrow \mathbb{Z}_N$, but **not** e, find $x \leftarrow Z_N$, $e > 1$ with $x^e = y \mod N$.
The strong RSA problem is a potentially **easier** problem than standard RSA. Hence, the relative assumption is **stronger**.

Let $h : \{0,1\}^* \rightarrow \mathbb{P}$ (primes)

- $Gen(1^k)$: choose $N = P \cdot Q$, $s \leftarrow \mathbb{Z}_N$.
  $pk = (N, s, h)$, $sk = (pk, \phi(N))$.
- $Sign(sk, m) : \sigma := s^{1/h(m) \mod \phi(N)} \mod N$
- $Vfy(pk, m, \sigma) : \sigma^{h(m)} = s \mod N$

Furthermore, this scheme assumes that: $\forall m \in \{0,1\}^* : gcd(h(m), \phi(N)) = 1$, which can be enforced by letting h output **large** ($> N$) primes.

Theorem: for every PPT $\mathcal{A}$ that breaks **EUF-naCMA** security of the scheme, there are PPTs $\mathcal{B}$ and $\mathcal{C}$ that break collision resistance of h **OR** solve the sRSA problem, respectively, with success probability s.t. $\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}}$.

Proof: Divide the scenario into event $E_0 \vee E_1$:

Event E_0 : there is an m_i with $h(m_i) = h(m^*) \rightarrow$ reduce to collision-resistance of h.

Event E_1 : for all $i \in \{1, ..., q\}$ we have $h(m_i) \neq h(m^*)$. Adversary $\mathcal{C}$ gets as input (N, y) and sets up $s := y^{\prod_{i=1}^{q} h(m_i)} \mod N$ accordingly.[2] s is "well-distributed" because for different $m_i \Rightarrow$ different s, because h is invertible.
The signature for m_j is $\sigma_j = y^{\prod_{i=1 \setminus j}^{q} h(m_i)} \mod N$.

We have $gcd(h(m^*), \prod_{i=1}^{q} h(m_i)) = 1$, since h maps to **prime** numbers and E_1 occurred.
Now use Shamir's trick for $(\sigma^*)^{h(m^*)} = s = y^{\prod_{i=1}^{q} h(m_i)} \mod N$, with J being y and e being $h(m^*)$.
Hence, $(x, h(m^*))$ is the desired sRSA solution.

At least one of the two events must happen with a **non-negligible** probability if $\mathcal{A}$ succeeds.

How could we obtain such a hash function h? One solution would be to use a collision-resistant hash H and set $h(m) := nextPrime(H(m)||0^l)$ (padding serves to randomize the function). However, this

[2] All the previous messages are needed, that's why we are proving for EUF-naCMA.

solution is not very efficient (polynomial time for computing primes).

The construction of an efficient EUF-CMA secure scheme from RSA is still an **open problem**.

Chapter 4

Chameleon hash functions and signatures

The goal of these constructs is to build a signature scheme, such that **authenticity** can still be verified, even though it is **not possible to convince third parties** of the signed message anymore (i.e., signer's message deniability).

4.1 Chameleon hash functions

A chameleon hash function CH consists of two PPT algorithms $(Gen_{CH}, TrapColl_{CH})$ such that:

- $Gen_{CH}(1^k)$ outputs a trapdoor τ and a function $ch : \mathcal{M} \times \mathcal{R} \to \mathcal{N}$, where $\mathcal{M}$ is the message space, and $\mathcal{R}$ is a randomness space.
- $TrapColl_{CH}(\tau, m, r, m')$ outputs $r' \in \mathcal{R}$ such that $ch(m, r) = ch(m', r')$.

This means the **preimage** can be **changed** by the **owner** (like a chameleon changes colour).

We require this function to be **well-distributed**: for random r, r' is also random, and **collision-resistant** without the knowledge of τ (an adversary $\mathcal{A}$ that knows **only** ch cannot produce a collision with a non-negligible probability).

Chameleon hashing based on DLog As usual, $\mathbb{G}$ group of prime order p, generator g:

- $Gen(1^k) : x \leftarrow \mathbb{Z}_p^*$, $h := g^x$, $ch := (g, h)$, $\tau := x$.
  $ch(m, r) := g^m \cdot h^r$
- $TrapColl(\tau, m, r, m')$: compute r' equalizing $ch(m, r) = ch(m', r')$ and knowing $x \Rightarrow r' = \frac{m-m'}{x} + r \mod p$.

Theorem: for every PPT $\mathcal{A}$ that breaks collision-resistance of CH, namely given $ch = (g, h)$, outputs a tuple (m, r, m', r'), with $(m, r) \neq (m', r')$, there exists a PPT $\mathcal{B}$ that breaks the DLog problem in $\mathbb{G}$ with roughly same runtime and success probability $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$.

Proof: similar to DLog based one-time signatures.

Chameleon hashing based on RSA (prime-e-)

- $Gen(1^k)$: $N = P \cdot Q$, message space $\mathbb{Z}_{2^l}$, prime $e > 2^l$ with $gcd(e, \phi(N)) = 1$, $J \leftarrow \mathbb{Z}_N$, $ch := (N, e, J)$, $\tau := d$.
  $ch(m, r) := J^m \cdot r^e \mod N$
- $TrapColl(\tau, m, r, m')$: compute r' equalizing $ch(m, r) = ch(m', r')$ and knowing $d \Rightarrow r' = (J^{m-m'} \cdot r^e)^d \mod N$.

Theorem: for every PPT $\mathcal{A}$ that breaks collision-resistance of CH, there exists a PPT $\mathcal{B}$ that solves the **prime-e-RSA** problem with roughly same runtime and success probability $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$.

Proof: similar to RSA based one-time signatures.

4.2 Chameleon signatures

Given $CH = (Gen_{CH}, TrapColl_{CH})$ and a signature scheme $\Sigma' = (Gen', Sign', Vfy')$, a chameleon signature scheme Σ is defined as:

- $Gen(1^k)$ runs $Gen'(1^k)$ and $Gen_{CH}(1^k)$ such that $pk = (pk', ch)$ and $sk = (sk', \tau)$.
- $Sign(sk, m, ch)$ runs on ch of the **receiver!**
  $r \leftarrow \mathcal{R}$, $ch(m, r) =: y$, $\sigma' = Sign'(sk, y)$, $\sigma = (\sigma', r)$.
- $Vfy(pk, m, \sigma, ch)$ runs $Vfy'(pk', ch(m, r), \sigma')$.

Question: is the EUF-CMA security notion "strong" enough for this scheme? $\rightarrow$ **No**! Because it is **not realistic**!
An adversary could choose his own ch_i for every query, but the standard scheme enforces to use ch chosen by the challenger.
The following attack is possible if we take this possibility into account:

Suppose a DLog-based CH is used. $\mathcal{A}$ receives $ch = (g, h)$ from the challenger, but defines $ch_{\mathcal{A}} := (g^a, h)$, for $a \neq 1$ chosen by $\mathcal{A}$. This is still a valid CH function fulfilling the properties mentioned above.
$\mathcal{A}$ queries m under $ch_{\mathcal{A}}$ and receives $\sigma = (\sigma', r)$. $\mathcal{A}$ sends the forgery $(m^* = a \cdot m \neq m, \sigma)$ under ch which verifies to 1:
$1 = Vfy(pk, m, \sigma, ch_{\mathcal{A}}) = Vfy'(pk, ch_{\mathcal{A}}(m, r), \sigma') = Vfy'(pk, ch(a \cdot m, r), \sigma') = Vfy(pk, m^* = a \cdot m, \sigma, ch)$.

This means that an EUF-CMA secure chameleon signature scheme **breaks completely** once the adversary can query signatures for **different recipients**, if they can **choose** their own ch_i.

In the following, we will consider this **weak** version of EUF-CMA as it allows to develop simpler proofs.

Theorem: for every PPT $\mathcal{A}$ that breaks **EUF-CMA** security of Σ, there exist PPTs $\mathcal{C}$ and $\mathcal{B}$ that respectively break collision-resistance of CH **or EUF-naCMA** security of Σ' with roughly same runtime and success probability:

$$\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}} \tag{4.1}$$

Proof: Let (m^*, σ^*) be the output forgery by $\mathcal{A}$, and $\sigma_i = (\sigma'_i, r_i)$ the reply to m_i's query.
Two disjoint events happen: E_0, $\mathcal{A}$ wins and there is an i with $ch(m_i, r_i) = ch(m^*, r^*)$;
E_1, $\mathcal{A}$ wins and $ch(m^*, r^*) \neq ch(m_i, r_i)$ for all i.
If E_0, usual reduction to collision-resistance of ch.

If E_1, choose in advance $m'_i = ch(M_i, R_i)$ for arbitrary M_i, R_i. Since EUF-naCMA of Σ' excludes a signing oracle for Σ', use τ to generate $r_i : ch(m_i, r_i) = m'_i$.
Upon receipt of forgery (m^*, σ^*), σ'^* is a valid signature for $m'^* = ch(m^*, r^*)$, and $m'^* \neq m'_i$ because E_1 happened.

4.3 Chameleon hash functions are one-time signatures

Given $CH = (Gen_{CH}, TrapColl_{CH})$, construct a one-time signature scheme Σ with:

- $Gen(1^k)$:

  - $(ch, \tau) \leftarrow Gen_{CH}(1^k)$
  - $(\widetilde{m}, \widetilde{r}) \leftarrow \mathcal{M} \times \mathcal{R}$
  - $c := ch(\widetilde{m}, \widetilde{r})$
  - $pk := (ch, c), sk := (\tau, \widetilde{m}, \widetilde{r})$
- $Sign(sk, m) : \sigma := r = TrapColl_{CH}(\tau, \widetilde{m}, \widetilde{r}, m)$
- $Vfy(pk, m, \sigma) : c = ch(m, \sigma)$

Theorem: the scheme Σ is **EUF-1-naCMA** secure if CH is collision-resistant.

4.4 Stronger form of EUF-CMA security

The **strong EUF-CMA** experiment is won by an adversary $\mathcal{A}$ iff $Vfy(pk, m^*, \sigma^*) = 1 \wedge (m^*, \sigma^*) \notin \{(m_1, \sigma_1), ..., (m_q, \sigma_q)\}$.

This means that $\mathcal{A}$ can win even if m^* has been **signed before**, as long as σ^* is **fresh**.

Given an EUF-CMA secure scheme Σ', and a CHF CH, it is possible to construct a **sEUF-CMA secure** scheme Σ:

- $Gen(1^k)$:
  - $(pk', sk') \leftarrow Gen'(1^k)$
  - $(ch_F, \tau_F) \leftarrow Gen_{CH}(1^k)$
  - $(ch_H, \tau_H) \leftarrow Gen_{CH}(1^k)$
  - $pk := (pk', ch_F, ch_H), sk := (sk', \tau_H)$
- $Sign(sk, m)$:[1] Let m', σ' arbitrary,
  - $r_F \leftarrow \mathcal{R}$, $r'_H \leftarrow \mathcal{R}$
  - $h := ch_H(m'||\sigma', r'_H)$
  - $\widetilde{m} := ch_F(h, r_F)$
  - $\widetilde{\sigma} \leftarrow Sign'(sk', \widetilde{m})$
  - $r_H \leftarrow TrapColl_{CH}(\tau_H, m'||\sigma', r'_H, m||\widetilde{\sigma})$
  - $\sigma := (\widetilde{\sigma}, r_F, r_H)$
- $Vfy(pk, m, \sigma)$:
  - $h := ch(m||\widetilde{\sigma}, r_H)$ (circular dependency[2] generated by $TrapColl_{CH}$)
  - $\widetilde{m} = ch_F(h, r_F)$
  - $Vfy'(pk', \widetilde{m}, \widetilde{\sigma}) = 1$

Theorem: if CH is collision-resistant and Σ' is EUF-CMA secure, then Σ is sEUF-CMA secure.

Proof idea (see B.4.4): Case E_0 : "forgery contains reused $\widetilde{m}^*$" $= \widetilde{m}_i \rightarrow$ reduction to CHF-CR. But how can we rely on both being CR if we use one of them for signing? In reduction, guess which ch is broken, and use the other for the trapdoor needed for signing. Otherwise reduction would be useless (we have the trapdoor!). This is why two functions are needed! Case E_0 can thus be split into two sub-events.
If we could rely on **both** CHFs being collision-resistant, then we would have: same $\widetilde{m} \Rightarrow$ same $h, r_F \Rightarrow$ same $m, \widetilde{\sigma}, r_H \Rightarrow$ same m, σ. Hence, fresh $(m, \sigma) \Rightarrow$ fresh $\widetilde{m}$. Case E_1 : "forgery contains fresh (m^*, σ^*)" $\rightarrow$ fresh $\widetilde{m}^*$ (see above) $\rightarrow$ trivially reduce to EUF-CMA by relaying.

[1] Can also be done in another way relying on τ_F.

[2] A negligible change in the signature affects the whole scheme. Every piece of the construction is meaningful and protected.

Chapter 5

Pairing-based signatures

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of prime order p. A **pairing** is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

- Bilinearity: $\forall g_1, g_1' \in \mathbb{G}_1 : e(g_1 \cdot g_1', g_2) = e(g_1, g_2) \cdot e(g_1', g_2)$ and for $g_2, g_2' \in \mathbb{G}_2$ as well $\Rightarrow e(g_1^a, g_2) = e(g_1, g_2)^a = e(g_1, g_2^a)$.
- Non-degeneracy: for all generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2 : e(g_1, g_2) \neq 1$ (if prime p) and generates $\mathbb{G}_T$.
- e is efficiently computable.

Note: a pairing operation is **less** efficient than exponentiation. There are three types of pairings: if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$, the pairing is **symmetric** (type 1), otherwise it's asymmetric. Asymmetric pairings can be transformed to symmetric ones via an efficient (type 2), or not efficient (type 3), homomorphism.

Joux's 3-party key exchange Being bilinear, pairings can be used to efficiently exchange a secret key $k = e(g, g)^{abc}$, just by knowing a secret value a, and applying bilinearity on $e(g^b, g^c)^a$. This applies for groups where this problem is difficult without knowing at least a secret value, of course.

5.1 BLS signatures

The following scheme is a simple way of using pairings in a signature scheme, which outputs **short** signatures. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a symmetric pairing over a group of prime order p of generator g. Given a hash function $H : \{0,1\}^* \to \mathbb{G} \setminus \{1\}$:

- $Gen(1^k) : sk = x \leftarrow \mathbb{Z}_p^*, pk = (g, g^x)$
- $Sign(sk, m) : \sigma = H(m)^x \in \mathbb{G}$
- $Vfy(pk, m, \sigma) : e(H(m), g^x) = e(\sigma, g)$

Correctness is trivially verified. This signature scheme is built over groups where the following problem is hard:

Computational Diffie-Hellman problem (CDH) Given (g, g^x, g^y), compute g^{xy}. Note: self-bilinear pairings ($\mathbb{G}_T = \mathbb{G}_1 = \mathbb{G}_2$) efficiently break CDH: $e(g, g) = g^\alpha, \alpha \neq 1 \to e(g^x, g^y) = g^{\alpha xy}$. $g^{xy} = e(g^{\alpha xy}, g^{\alpha^{-2}})$, where $g^{\alpha^{-2}} = g^{\alpha^{p-3}} \mod p$ is efficiently computed from $e(g, g)$ with square-and-mult and applying Fermat's little theorem.
Theorem: Assuming H is modeled as a **random oracle**, for every PPT $\mathcal{A}$ that breaks **EUF-CMA** security of a BLS signature scheme, there exists a PPT $\mathcal{B}$ that solves the CDH problem with roughly same runtime and success probability $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{q_H}$, where q_H is the number of hash queries.
Proof: Assuming $\mathcal{A}$ always makes hash queries before, $\mathcal{B}$ simulates the random oracle and performs the EUF-CMA challenge, embedding g^y received from the CDH challenge in the hash query $H(m^*)$, of index i^*, guessed randomly. Otherwise, embed g^{y_i} from randomly chosen y_i in order for the EUF-CMA

simulation to be correct.
Just like RSA-FDH, the proof can be improved to loss of $\mathcal{O}(q_s)$ (# of signing queries, cf. Exercises).
Overall, this signature has a ROM security proof, which is neither tight. However, it provides **short**, efficient, and **simple** signatures, with interesting properties:

- **Aggregability**: in order to verify signatures of messages from many (potentially different) senders, a naive solution would just be to linearly iterate over them.

  BLS signatures ensure that verifying a single aggregate signature $|\sigma_{Agg}| = |\sigma_i|$ (plus cost of aggregating) is more efficient than verifying the single signatures:
  $\sigma_{Agg} := \prod_{i=1}^{n} \sigma_i$.
  $Vfy(pk_1, ..., pk_n, m_1, ..., m_n, \sigma_{Agg}) : e(\sigma_{Agg}, g) = \prod_{i=1}^{n} e(H(m_i), g^{x_i})$.

  Correctness and EUF-CMA security are mantained while performing only $n+1$ pairing operations instead of $2n$ (2 operations per signature) when verifying.

- **Batch verification**: in order to verify signatures of messages from a **single** sender, only **two** pairing operations are sufficient:
  $h := \prod_{i=1}^{n} H(m_i)$, $\sigma := \prod_{i=1}^{n} \sigma_i$
  $Vfy(pk, m_1, ..., m_n, \sigma) : e(\sigma, g) = e(h, g^x)$

5.2 Waters' signatures

This pairing-based signatures scheme is EUF-CMA secure under CDH in the **standard model**. In order to achieve this result, some sort of **programmability** of a hash function is needed, a property that simulates the intrinsic programming operations performed by a random oracle.

5.2.1 Programmable Hash Functions

Intuitively, such hash functions partition the set of hashable messages into a *controlled* set, for which a reduction can **sign** the message in the scheme, and an *uncontrolled* set, for which CDH can be embedded into m.
The most useful hash function would be the one that solves the DLog problem (using a trapdoor) for controlled messages, whilst for some m, DLog is hard even given the trapdoor. However, this is hard to achieve, as knowing most DLogs usually allows to infer it for **all** messages.
A refinement is needed: m controlled $\iff H(m) = h^{a_m} g^{b_m} \wedge a_m \neq 0$ m uncontrolled $\iff H(m) = h^{a_m} g^{b_m} \wedge a_m = 0$, where g, h are publicly known generators of a cyclic group $\mathbb{G}$, but exponents are only known to reduction. More formally: Given a group hash function family $H_\kappa : \{0,1\}^l \to \mathbb{G}$, a group hash function (GHF) is (v, w, γ)-programmable (PHF) iff it is equipped with the following PPT algorithms:

- $Gen(1^k) \to \kappa$
- $TrapGen(g, h) \to (\kappa, \tau)$
- $Eval(\kappa, m) \to H_\kappa(m) \in \mathbb{G}$ (deterministic)
- $TrapEval(\tau, m) \to (a_m, b_m)$, with $h^{a_m} g^{b_m} = H_\kappa(m)$ (deterministic)

and has the following properties:

- κ from Gen is **statistically close** to κ from $TrapGen$ (same distribution, indistinguishable).
- $TrapEval$ has (v, w, γ)-**well-distributed** outputs, for $v, w \in \mathbb{N}, \gamma \in [0, 1]$.

The second property means that for all generators g, h of $\mathbb{G}$, two sets of messages $m_1^*, ..., m_v^* \in \{0,1\}^l$, and $m_1, ..., m_w \in \{0,1\}^l$, and κ statistically close to the one in $TrapGen$:

$$Pr[a_{m_i^*} = 0 \quad \forall\, i = 1, ..., v \ \wedge \ a_{m_j} \neq 0 \quad \forall\, j = 1, ..., w] \geq \gamma, \tag{5.1}$$

over τ. Intuition: the **greater** γ, the more likely the signature reduction will succeed.
Claim: $\gamma \leq 1/2^{\min\{v,w\}} \Rightarrow$ cannot have both large v and w. This is not an issue in our application as we are interested in $(1, q, \gamma)$-**programmability** for the reduction proof.

5.2.2 Waters' programmable hash function

Let $q = q(k)$ be a polynomial. Then the following hash function is $(1, q, \gamma)$-**programmable** for $\gamma = 1/\Theta(q\sqrt{k})$:

- $Gen(1^k)$: choose $u_0, ..., u_k \leftarrow \mathbb{G} \Rightarrow \kappa = (u_0, ..., u_k)$
- $TrapGen(g, h)$: choose $\widehat{a_i} \in \mathbb{Z}_p$ suitably (see later), and $\widehat{b_i} \leftarrow \mathbb{Z}_p$ randomly. Let $u_i = h^{\widehat{a_i}} g^{\widehat{b_i}}$.
  $\tau = (\widehat{a_0}, ..., \widehat{a_k}, \widehat{b_0}, ..., \widehat{b_k})$
- $Eval(\kappa, m = m^{(1)}...m^{(k)}) : H_\kappa(m) = u_0 \prod_{i=1}^{i=k} u_i^{m^{(i)}}$, $m^{(i)} \in \{0, 1\}$
- $TrapEval(\tau, m = m^{(1)}...m^{(k)})$: compute $a_m = a_0 + \sum_{i=1}^{i=k} \left(\widehat{a_i} m^{(i)}\right)$, and $b_m = b_0 + \sum_{i=1}^{i=k} \left(\widehat{b_i} m^{(i)}\right)$, such that:
  $h^{a_m} g^{b_m} = h^{\widehat{a_0}} \prod_{i=1}^{i=k} h^{\widehat{a_i} m^{(i)}} \cdot g^{\widehat{b_0}} \prod_{i=1}^{i=k} g^{\widehat{b_i} m^{(i)}} = \left(h^{\widehat{a_0}} g^{\widehat{b_0}}\right) \prod_{i=1}^{i=k} \left(h^{\widehat{a_i}} g^{\widehat{b_i}}\right)^{m^{(i)}} = H_\kappa(m)$

By sampling $\widehat{b_i}$ uniformly from $\mathbb{Z}_p$, also u_i and thus κ from $TrapGen$ has a uniform distribution, regardless of how $\widehat{a_i}$ is chosen (this is in fact the purpose of having g^{b_m}). $\widehat{a_i}$ is chosen such that $(1, q, \gamma)$-well-distribution is achieved: The idea is to set up each of them as random walks: $\widehat{a_i} = \sum_{j=1}^{L} \widehat{a_{i,j}}$, for $\widehat{a_{i,j}} \leftarrow \{-1, 0, 1\}$ and $L = \Theta(q)$. This way, the sum of random walks a_m fulfills the following constraint: $1/\Theta(q\sqrt{k}) \leq Pr[a_m = 0] \leq 1/\Theta(q)$, depending on the Hamming weight of the message. The free result $Pr[\forall i : a_{m_i} \neq 0 \mid a_{m^*} = 0] \geq 1/2 \Rightarrow Pr[\forall i : a_{m_i} \neq 0 \wedge a_{m^*} = 0] \geq 1/\Theta(q\sqrt{k})$, which is what we were looking for.

5.2.3 Waters' signatures

Given a symmetric pairing and a PHF H, this signature scheme is based on the following algorithms:

- $Gen(1^k) : g^\alpha \in \mathbb{G}, \kappa \leftarrow Gen_H(1^k)$.
  $pk = (g, \kappa, e(g, g^\alpha)), sk = g^\alpha$
- $Sign(sk, m) : r \leftarrow \mathbb{Z}_p, \sigma_1 := g^r, \sigma_2 := g^\alpha \cdot H_\kappa(m)^r$.
  $\sigma = (\sigma_1, \sigma_2)$
- $Vfy(pk, m, \sigma) : e(g, \sigma_2) = e(g, g^\alpha) \cdot e(\sigma_1, H_\kappa(m))$

Correctness is trivially verified. Theorem: Let H be a $(1, q, \gamma)$-PHF for any polynomial q. For every PPT $\mathcal{A}$ that breaks **EUF-CMA** security of the scheme, there exists a PPT B that breaks **CDH** with roughly same runtime and success probability $\epsilon_\mathcal{B} \geq \gamma \cdot \epsilon_\mathcal{A}$.
Proof: $\mathcal{B}$ gets CDH challenge (g, g^x, g^y), generates $(\kappa, \tau) \leftarrow TrapGen(g, g^x)$ and sets $pk = (g, \kappa, e(g^x, g^y))$. This implicitly sets $\alpha = xy$, even if it does not know this value $\Rightarrow$ there is an **alternative** way to sign, thanks to the **decomposition** of the PHF!
$\mathcal{B}$ wins if $\mathcal{A}$ wins and $a_{m_i} \neq 0$ for all signature queries and $a_{m^*} = 0$ for the forgery m^*. This happens with probability γ by construction, thus: $\epsilon_\mathcal{B} \geq \gamma \cdot \epsilon_\mathcal{A}$.
How does simulation work in this setting? $TrapEval$ derives $H_\kappa(m) = (g^x)^{a_m} g^{b_m}$. Choose fresh $s \leftarrow \mathbb{Z}_p$ and set $\sigma_1 := (g^y)^{-1/a_m} \cdot g^s$. This implicitly sets $r = -y/a_m + s$, which we don't need to know.

$$\sigma_2 := (g^x)^{a_m s} \cdot (g^y)^{-b_m/a_m} \cdot g^{b_m s} \tag{5.2}$$

We actually need to show that $\sigma_2 = g^{xy} H(m)^r \Rightarrow$ Group factors in the exponent and multiply and divide by g^{xy}. This is valid **only** if $a_m \neq 0$!!!
Upon receipt of the forgery $(m^*, \sigma^*), H(m^*) = g^{b^*}$. Then extract g^{xy} as:

$$\sigma_2^* \cdot (\sigma_1^*)^{-b^*} = g^{xy} \cdot H(m^*)^{r^*} \cdot g^{-b^* r^*} = g^{xy} \tag{5.3}$$

Basically, all values are set up in a **clever way**, so that the g^{xy}-term cancels out when signing, and shows up only at the forgery.
This signature scheme is **less** efficient than BLS (has more pairing operations), but it's proved to be secure in the **standard model**.

Chapter 6

Identification-scheme-based signatures

The goal of an identification scheme is **asymmetric authentication** of parties. More specifically, a **verifier** V wants to be certain that a **prover** P knows a secret key sk. Of course, V should **not learn** sk itself, so simply sending the latter to it would not be useful.
Using a signature scheme in a **non-interactive** way allows an adversary to impersonate the authenticated prover! An interactive attempt, on the other hand, does not explicitly imply the guarantees mentioned above.
Clearly, a more formal approach should be defined:

6.1 Sigma protocols

Sigma-protocols are **three-move** protocols that prove knowledge of sk through the exchange of three messages: commitment $com : P \to V$, challenge $ch : V \to P$, response $res : P \to V$. V finally outputs a bit to indicate whether it is convinced.
Note: ch **must** be chosen uniformly at **random**. The verifier does not use any hidden coins, the protocol is **publicly accessible**. A sigma protocol has to satisfy the following properties: given a relation $\mathcal{R}$, with $(pk, sk) \in \mathcal{R}$ for *corresponding* (pk, sk),

- **Completeness**: $(pk, sk) \in \mathcal{R} \Rightarrow V$ outputs 1
- **Special soundness**: from any two accepting transcripts with same com, pk, it is possible to efficiently **extract** sk.
- **Special honest-verifier zero-knowledge**: given pk and **random** ch, can efficiently compute accepting $com, res \Rightarrow V$ **cannot convince** third parties that communication ever happened.

Example: $pk = (g, g^x), sk = x$ $com = g^k$ for $k \leftarrow \mathbb{Z}_p$ $ch = e \leftarrow \mathbb{Z}_p$ $res = k + xe \mod p$ The above properties can be easily verified for this scheme.
Other than identification schemes, sigma protocols can be used for proof systems and for **signature schemes**. The latter can be built by making the protocol **non-interactive**, as if the signer talks to itself. The following transformation allows this construction.

6.2 Fiat-Shamir transformation and derived signatures

Let H be a hash function with range of ch, then set $ch := H(com, m)$, and $\sigma := (com, res)$.
Let $\mathcal{R}$ be an efficiently computable relation for checking if (pk, sk) is a corresponding pair of keys. We say that $\mathcal{R}$ is **hard** if every adversary $\mathcal{A}$ has negligible probability to extract a corresponding key sk' by only knowing pk.
Theorem: for every PPT $\mathcal{A}$ that breaks **EUF-CMA** security on a Fiat-Shamir-based signature scheme, there exists a PPT $\mathcal{B}$ on the **hardness** of $\mathcal{R}$ with roughly same runtime and success probability $\Theta(\epsilon_{\mathcal{A}}^2/q)$, where q is the number of hash queries in the **random oracle** model (ROM).

Proof: The problem is $\mathcal{B}$ does not know sk to sign messages. The workaround is to make use of the special HVZK property of the protocol: when asked to **hash** com, m, just choose a *fresh* image $H(com, m)$. When asked to **sign** m, choose ch randomly and derive com', res', and set $H(com', m) := ch$.
What about extraction? The aim is to use special soundness, but the forgery (com^*, res^*) is not enough, as we need **two** different transcripts. Solution lies in "**time travel**": run experiment, if $\mathcal{A}$ does not fail then **rewind** to the first time we gave $\mathcal{A}$ the value $H(com^*, m^*) = ch^*$. This time, output $H(com^*, m^*) = ch' \neq ch^*$, then **hope** $\mathcal{A}$ forges again on m^* (forking lemma).
These signature schemes are very simple and **efficient**, but have a **lossy** reduction! **Schnorr signatures** use the **DLog** scheme for the sigma protocol: $pk = (g, g^x), sk = x$. $Sign(sk, m)$: choose $k \leftarrow \mathbb{Z}_p$, set $e = H(com, m) = H(g^k, m)$, $\sigma = (com, res) = (g^k, k + xe \mod p)$ $Vfy(pk, m, \sigma) : g^{\sigma_2} = \sigma_1 \cdot (g^x)^e$

Appendix A

Proof strategies

Let A be an assumption (e.g., "f is a one-way function"), and S be a security claim (e.g., "Lamport signatures with f are EUF-1-naCMA secure"). To need to show: $A \Rightarrow S$, we often show: $\neg S \Rightarrow \neg A$.
More practically, show that for every adversary $\mathcal{A}$ breaking the security claim, there exists an adversary $\mathcal{B}$ breaking the security assumption. Challenger $\mathcal{C}$ challenges $\mathcal{B}$, $\mathcal{B}$ transforms $\mathcal{C}$'s input and challenges $\mathcal{A}$ (simulation). Transforming $\mathcal{A}$'s successful output, $\mathcal{B}$ responds to $\mathcal{C}$ (extraction).
This proof system must further show that $\mathcal{A}$ must not be able to distinguish $\mathcal{B}$ from a honest challenger $\Rightarrow$ show that exchanged values have same distribution. Moreover, hope that runtimes are comparable and success probability reduction is tight.
For statements like: $(A \wedge B) \Rightarrow S$ it is equivalent to show $\neg S \Rightarrow (\neg A \vee \neg B)$.

Appendix B

Summary

Hash-then-Sign EUF-CMA + CR $H \rightarrow$ EUF-CMA $\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}}$

Lamport's one-time signatures One-way function $\rightarrow$ EUF-1-CMA $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{n}$, where n is the length of the message

DLog-based one-time signatures DLog hard $\rightarrow$ EUF-1-naCMA $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$

RSA-based one-time signatures Prime-e-RSA hard $\rightarrow$ EUF-1-naCMA $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$

Reusable scheme from one-time signatures EUF-1-naCMA + EUF-naCMA $\rightarrow$ EUF-CMA $q \cdot \epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}}$

Textbook RSA RSA hard $\rightarrow$ UUF-NMA, **NOT** EUF-NMA $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$

RSA PKCS #1 v1.5 No attacks known, no security proof

RSA-FDH RSA hard $\rightarrow$ EUF-CMA in **ROM** $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{q_H}$, can be improved to $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{2e \cdot q_S}$

RSA-PSS RSA hard $\rightarrow$ EUF-CMA in **ROM**, less lossy reduction $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}} - w, w \in \mathcal{O}((q_S + q_H)^2 \cdot (2^{-k_0} + 2^{-k_1}))$ (negligible)

GHR signatures sRSA hard + CR $H \rightarrow$ EUF-naCMA in the **standard** model $\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}}$

DLog-based CHF DLog hard $\rightarrow$ CHF collision resistant $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$

RSA-based CHF Prime-e-RSA hard $\rightarrow$ CHF collision resistant $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$

Chameleon signatures EUF-naCMA + CHF collision resistant $\rightarrow$ EUF-CMA (weak version) $\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}}$

CHFs are one-time signatures CHF collision resistant $\rightarrow$ EUF-1-naCMA (also sEUF-1-naCMA) $\epsilon_{\mathcal{B}} \geq \epsilon_{\mathcal{A}}$

Strong EUF-CMA security EUF-CMA + CHF family collision resistant $\rightarrow$ sEUF-CMA $\epsilon_{\mathcal{B}} + 2 \cdot \epsilon_{\mathcal{C}} \geq \epsilon_{\mathcal{A}}$

BLS signatures CDH hard $\rightarrow$ EUF-CMA in **ROM** (also aEUF-CMA) $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{q_H}$, can be improved to $\epsilon_{\mathcal{B}} \geq \frac{\epsilon_{\mathcal{A}}}{2e \cdot q_S}$

Waters' PHF $(1, q, \gamma)$-PHF $\Rightarrow \gamma \geq 1/\Theta(q\sqrt{k})$

Waters' signatures CDH hard $\rightarrow$ EUF-CMA in the **standard** model $\epsilon_{\mathcal{B}} \geq \gamma \cdot \epsilon_{\mathcal{A}}$

Fiat-Shamir-based signature schemes $\mathcal{R}$ hard $\rightarrow$ EUF-CMA in **ROM** $\epsilon_{\mathcal{B}} \approx \Theta(\frac{\epsilon_{\mathcal{A}}^2}{q_H})$