CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 12, 2022

Lecture 13

Digital signatures

Motivation and definitions

RSA signatures

Using hash functions

Full Domain Hash

RSA PSS (Probabilistic Signature Scheme)

DLog-based signature schemes

Plan

Motivation and definitions

RSA signatures

Using hash functions

Full Domain Hash

RSA PSS (Probabilistic Signature Scheme)

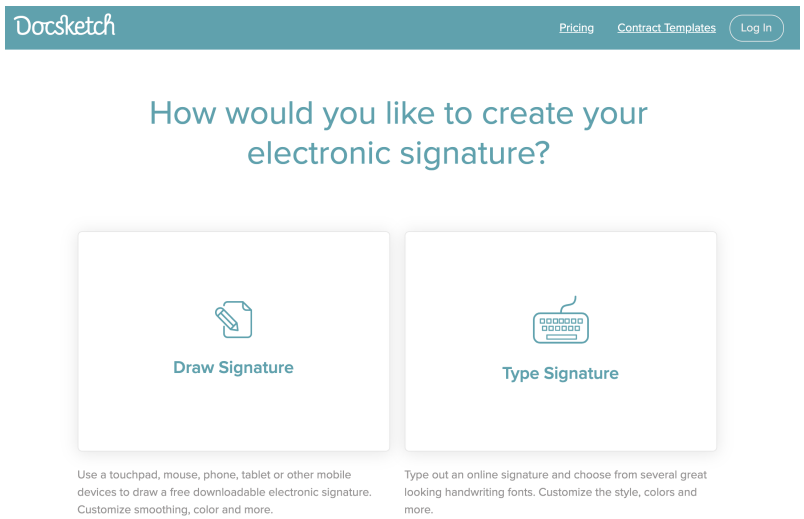
DLog-based signature schemes

Handwritten signatures

Handwritten signatures are in common use to sign documents, checks, . . .

Handwritten signatures

There are many ways to create digital versions of handwritten signatures.



Handwritten signatures

Problem: A digitized handwritten signature is easily copied, leading to forgery.

To be secure, a digital signature must depend not only on the signer, but also on the message being signed.

Digital signatures syntax and correctness

A digital signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ consists of three algorithms that operate as follows:

- $(vk, sk) \xleftarrow{\$} \mathcal{K}$ — generate a verification key vk and matching signing key sk
- $\sigma \xleftarrow{\$} \mathcal{S}_{sk}(M)$ — apply signing algorithm $\mathcal{S}$ to signing key sk and message M to get a signature σ. Algorithm $\mathcal{S}$ may be randomized.
- $d \leftarrow \mathcal{V}_{vk}(M, \sigma)$ — apply verification algorithm $\mathcal{V}$ to verification key vk, message M and candidate signature σ to get a decision $d \in \{0, 1\}$.

Correctness requirement: $\Pr[\mathcal{V}_{vk}(M, \mathcal{S}_{sk}(M)) = 1] = 1$ for all (vk, sk) that may be output by $\mathcal{K}$ and all M in the underlying message space. The latter may depend on the verification key.

How it works

Step 1: Key generation

Alice locally computes $(vk, sk) \xleftarrow{\$} \mathcal{K}$ and stores signing key sk.

Step 2: Alice enables any prospective signature verifier to get vk.

Step 3: Alice can generate a signature σ of a message M using sk.

Step 4: Anyone holding vk can verify that σ is Alice's signature on M.

We don't require privacy of vk but we do require authenticity: the sender should be assured vk is really Alice's key and not someone else's. Towards this, one could

- Put verification keys in a trusted but public "phone book", on one's Facebook page or personal webpage, ...
- Use certificates as we will see later.

Signatures versus MACs

Signatures are the public-key version of MACs.

In a MAC, the signing and verifying keys are the same key K. Only entities that hold K can verify MACs, and verifiers can thus forge MACs.

With signatures, anyone holding vk can verify Alice's signature under sk. Verifiers cannot forge signatures.

Suppose Alice uses a MAC to sign checks, the key K shared between her and her bank. If the bank's servers are breached, and K is compromised, the adversary can forge Alice's checks.

If signatures are used instead, the bank holds Alice's verification key vk. The adversary recovering it cannot forge Alice's checks.

UF-CMA Security of a DS scheme

Intent: Adversary should not be able to get a verifier to accept σ as Alice's signature of M unless Alice has previously signed M, even if adversary can obtain Alice's signatures on messages of the adversary's choice.

A change from UF-CMA for MACs: Adversary gets the verification key.

As with MA schemes, the definition does not require security against replay. That is handled on top, via counters or time stamps.

UF-CMA

Let $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ be a signature scheme.

Game $\mathrm{UFCMA}_{\mathcal{DS}}$

procedure Initialize
$(vk, sk) \xleftarrow{\$} \mathcal{K}$; $S \leftarrow \emptyset$
return vk

procedure Finalize(M, σ)
$d \leftarrow \mathcal{V}_{vk}(M, \sigma)$
return $(d = 1 \wedge M \notin S)$

procedure Sign(M)
$\sigma \xleftarrow{\$} \mathcal{S}_{sk}(M)$
$S \leftarrow S \cup \{M\}$
return σ

Definition: uf-cma advantage (digital signature version)

The uf-cma advantage of an adversary A is

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = \Pr\left[\mathrm{UFCMA}^{A}_{\mathcal{DS}} \Rightarrow \text{true}\right]$$

UF-CMA: Explanations

The "return vk" statement in **Initialize** means the adversary A gets the verification key vk as input. It does not get sk.

It can call **Sign** with any message M of its choice to get back a correct signature $\sigma \xleftarrow{\$} \mathcal{S}_{sk}(M)$ of M under sk. Notation indicates signing algorithm may be randomized.

To win, it must output a message M and a signature σ that are

- Correct: $\mathcal{V}_{vk}(M, \sigma) = 1$
- New: $M \notin S$, meaning M was not a query to **Sign**

Interpretation: Sign represents the signer and **Finalize** represents the verifier. Security means that the adversary can't get the verifier to accept a message that is not authentic, meaning was not already signed by the sender.

Different types of signature forgery

The crypto terminology sometimes distinguishes between different kinds of forgery.

- Existential forgery is when the adversary if free to choose (M, σ). This is what the UF-CMA game is about.
- Selective forgery is similar, except that the adversary must have chosen M prior to any oracle call.
- Universal forgery is when the adversary is able to sign any message.

Our notion of UF-CMA security is that even the least ambitious (easiest) goal for the adversary should be hard.

Plan

Motivation and definitions

RSA signatures

Using hash functions

Full Domain Hash

RSA PSS (Probabilistic Signature Scheme)

DLog-based signature schemes

Plain RSA signature scheme

Let $\mathcal{K}_{\rm rsa}$ be an RSA generator. The plain RSA signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is defined via:

Alg $\mathcal{K}$	**Alg** $\mathcal{S}_{(N,d)}(y)$	**Alg** $\mathcal{V}_{(N,e)}(y, x)$
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\rm rsa}$	$x \leftarrow y^d \bmod N$	If $(x^e \bmod N = y)$
Return $((N, e), (N, d))$	Return x	then return 1
		Else return 0

Above, $vk = (N, e)$ is the verification key and $sk = (N, d)$ is the signing key.

The message space is $\mathbb{Z}_N^*$, and $y \in \mathbb{Z}_N^*$ is the message. The signature is $x \in \mathbb{Z}_N^*$.

Correctness: If $x \leftarrow \mathcal{S}_{(N,d)}(y)$ then

$$x^e \bmod N = (y^d)^e \bmod N = y^{ed \bmod \varphi(N)} = y^1 = y \ .$$

Security of plain RSA signatures

We let $f(x) = x^e \bmod N$ and $f^{-1}(y) = y^d \bmod N$.

To forge the signature $x = f^{-1}(y)$ of a message y, the adversary, given N, e but not d, must compute $y^d \bmod N$.

But the RSA generator $\mathcal{K}_{\mathrm{rsa}}$ is assumed OW-secure, so this task should be hard and the scheme should be secure.

Correct?

Security of plain RSA signatures

We let $f(x) = x^e \bmod N$ and $f^{-1}(y) = y^d \bmod N$.

To forge the signature $x = f^{-1}(y)$ of a message y, the adversary, given N, e but not d, must compute $y^d \bmod N$.

But the RSA generator $\mathcal{K}_{\mathrm{rsa}}$ is assumed OW-secure, so this task should be hard and the scheme should be secure.

Correct?

Of course not...

Attacks on plain RSA

adversary $A((N, e))$
Return $(1, 1)$

$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$ because $1^d \bmod N = 1$

Attacks on plain RSA

adversary $A((N, e))$
Return $(1, 1)$

$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$ because $1^d \bmod N = 1$

adversary $A((N, e))$
Pick some distinct $y_1, y_2 \in \mathbb{Z}^*_N \setminus \{1\}$
$x_1 \leftarrow \mathbf{Sign}(y_1)$; $x_2 \leftarrow \mathbf{Sign}(y_2)$
Return $(y_1 y_2 \bmod N, x_1 x_2 \bmod N)$

$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$ because

$$(y_1 y_2)^d \bmod N = y_1^d y_2^d \bmod N = x_1 x_2 \bmod N$$

Attacks on plain RSA

adversary $A((N, e))$
$x \xleftarrow{\$} \mathbb{Z}_N^*$
$y \leftarrow x^e \bmod N$
Return (y, x)

$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$ because

$$y^d \bmod N = x^{ed} \bmod N = x^1 \bmod N = x$$

This adversary returns a valid signature of something that is probably garbage. But it is definitely a valid attack, because the adversary achieves a high advantage in the game!

Other issues

In plain RSA, the message is an element of $\mathbb{Z}_N^*$. We really want to be able to sign strings of arbitrary length.

Plan

Motivation and definitions

RSA signatures

Using hash functions

Full Domain Hash

RSA PSS (Probabilistic Signature Scheme)

DLog-based signature schemes

RSA PKCS#1 signatures

Signer has $vk = (N, e)$ and $sk = (N, d)$ where $|N| = 1024$.

Let h: $\{0,1\}^* \to \{0,1\}^{160}$ be a hash function (like SHA1) and let $n = 1024/8 = 128$.

Then

$$H_{PKCS}(M) = 00||01||\underbrace{FF||\ldots||FF}_{n-22}||\underbrace{h(M)}_{20}$$

And

$$\mathcal{S}_{N,d}(M) = H_{PKCS}(M)^d \bmod N$$

Idea:
- Force the element of $\mathbb{Z}_N^*$ to be of a very specific form, so that we can thwart the previous attacks.
- The first $n - 20$ bytes are a "sanity check". A signature that doesn't have the right form is rejected.

Does 1-wayness prevent forgery?

One-wayness:

No adversary should have a high advantage in the OW game.

Unforgeability:

No adversary should have a high advantage in the UFCMA game.

OW and UFCMA games

Game $\mathrm{OW}_{\mathcal{K}_{\mathrm{rsa}}}$

procedure Initialize
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\mathrm{rsa}}$
$x \xleftarrow{\$} \mathbb{Z}_N^*$; $y \leftarrow x^e \mod N$
return N, e, y

procedure Finalize(x')
return $(x = x')$

Game $\mathrm{UFCMA}_{\mathcal{DS}}$

procedure Initialize
$(vk, sk) \xleftarrow{\$} \mathcal{K}$; $S \leftarrow \emptyset$
return vk

procedure Finalize(M, σ)
$d \leftarrow \mathcal{V}_{\mathrm{vk}}(M, \sigma)$
return $(d = 1 \wedge M \notin S)$

procedure Sign(M)
$\sigma \xleftarrow{\$} \mathcal{S}_{sk}(M)$
$S \leftarrow S \cup \{M\}$
return σ

Does 1-wayness prevent forgery?

- Goal of a UFCMA adversary: Given (N, e, h), find M and σ such that $\sigma^e = H_{PKCS}(M) \bmod N$.
  In this game, there is no requirement that y be random.
- Goal of a OW adversary: Given (N, e, y), find $y^d \bmod N$.
  Here, y is random.

Problem: 1-wayness of RSA does not imply hardness of computing $y^d \bmod N$ if y is not random

H_{PKCS} revisited

Recall

$$H_{PKCS}(M) = 00||01||FF||\ldots||FF||h(M)$$

The first $n - 20 = 108$ bytes out of n are fixed so $H_{PKCS}(M)$ does not look "random" even if h is a RO or perfect.

We cannot hope to show RSA PKCS#1 signatures are secure assuming (only) that RSA is 1-way no matter what we assume about h and even if h is a random oracle. Such a theorem can't exist.

Plan

Motivation and definitions

RSA signatures

Using hash functions

Full Domain Hash

RSA PSS (Probabilistic Signature Scheme)

DLog-based signature schemes

Goal

We will validate the hash-then-decrypt paradigm

$$\mathcal{S}_{N,d}(M) = \mathbf{H}(M)^d \bmod N$$

by showing the signature scheme is provably UF-CMA assuming RSA is 1-way as long as **H** is a RO.

This says the paradigm has no "structural weaknesses" and we should be able to get security with "good" choices of **H**.

Full-Domain-Hash (FDH) [BR96]

Let $\mathcal{K}_{\rm rsa}$ be an RSA generator. Let $\mathbf{H} : \{0,1\}^* \to \mathbb{Z}_N^*$. The RSA FDH signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is defined via:

Alg $\mathcal{K}$	**Alg** $\mathcal{S}_{(N,d)}(M)$	**Alg** $\mathcal{V}_{(N,e)}(M, x)$
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\rm rsa}$	$y \leftarrow \mathbf{H}(M)$	$y \leftarrow \mathbf{H}(M)$
Return $((N, e), (N, d))$	$x \leftarrow y^d \bmod N$	If $(x^e \bmod N = y)$
	Return x	then return 1
		Else return 0

Above, $vk = (N, e)$ is the verification key and $sk = (N, d)$ is the signing key.

The message space is $\{0,1\}^*$, and $M \in \{0,1\}^*$ is the message. The signature is $x \in \mathbb{Z}_N^*$.

Correctness: If $x \leftarrow \mathcal{S}_{(N,d)}(M)$ and $y \leftarrow \mathbf{H}(M)$ then

$$x^e \bmod N = (y^d)^e \bmod N = y^{ed \bmod \varphi(N)} = y^1 = y \ .$$

$\mathbf{H}$ needs to be collision resistant

Suppose we have an adversary C that can find a collision for $\mathbf{H}$, meaning it returns (M_1, M_2) such that $M_1 \neq M_2$ but $\mathbf{H}(M_1) = \mathbf{H}(M_2)$. Then we can break the RSA FDH signature scheme $\mathcal{DS}$ via:

adversary $A((N, e))$
$(M_1, M_2) \xleftarrow{\$} C$
$x_1 \leftarrow \mathbf{Sign}(M_1)$
return (M_2, x_1)

We have $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{DS}}(A) = 1$ because $\mathbf{H}(M_1) = \mathbf{H}(M_2)$ implies M_1, M_2 have the same signatures:

$$x_1 = \mathcal{S}_{(N,d)}(M_1) = \mathbf{H}(M_1)^d \bmod N = \mathbf{H}(M_2)^d \bmod N = \mathcal{S}_{(N,d)}(M_2)$$

Conclusion: UF-CMA security of RSA FDH requires that $\mathbf{H}$ be collision-resistant.

This condition is necessary for UF-CMA. But it is not by itself sufficient.

Instantiating **H** in RSA FDH

We seek suitable functions **H**: $\{0,1\}^* \to \mathbb{Z}_N^*$.

First let **G** be an XOF (eXtendable Output length Function), meaning it takes W, ℓ and returns ℓ bits. Possible choices are:

- $\mathbf{G}(W, \ell) = \mathrm{SHAKE256}(W, \ell)$
- $\mathbf{G}(W, \ell)$ is the first ℓ bits of the sequence

  $$\mathrm{SHA256}(\langle 0 \rangle \| W) \,\|\, \mathrm{SHA256}(\langle 1 \rangle \| W) \,\|\, \cdots \,\|\, \mathrm{SHA256}(\langle 2^8 - 1 \rangle \| W)$$

  where $\langle i \rangle$ is a 1-byte encoding of i and we assume $\ell \leq 2^8 \cdot 256$.

Instantiating **H** in RSA FDH

Let k be the security parameter associated to our RSA generator $\mathcal{K}_{\text{rsa}}$, so that $2^{k-1} < N < 2^k$.

We could set $\mathbf{H}(M) = \mathbf{G}(M, k-1)$. However, this function is not onto $\mathbb{Z}_N$, since outputs y in the range $2^{k-1}, \ldots, N-1$ are never returned.

Instead we could set $\mathbf{H}(M) = \mathbf{G}(M, 2k) \bmod N$.

These functions return outputs in $\mathbb{Z}_N$, while we want outputs in $\mathbb{Z}_N^* \subseteq \mathbb{Z}_N$. We can simply ignore this, reasoning as follows.

First, for N a product of two distinct, odd primes, the RSA function $x \mapsto x^e \bmod N$ remains a permutation on $\mathbb{Z}_N$, with inverse $y \mapsto y^d \bmod N$, so correctness of the RSA FDH signature scheme is maintained.

Second, if an output $y = \mathbf{H}(M)$ is non-zero and in $\mathbb{Z}_N \setminus \mathbb{Z}_N^*$, then we can factor N, so this is unlikely to happen and security is maintained.

Plan

Motivation and definitions

RSA signatures

Using hash functions

Full Domain Hash

RSA PSS (Probabilistic Signature Scheme)

DLog-based signature schemes

RSA PSS [BR96]

Let $\mathcal{K}_{\mathrm{rsa}}$ be an RSA generator with k-bit public keys, and ℓ a parameter satisfying $17 \leq 2\ell + 1 < k$. Let $\mathbf{H}_1, \mathbf{H}_2 : \{0,1\}^* \to \{0,1\}^\ell$ and $\mathbf{H}_3 : \{0,1\}^* \to \{0,1\}^{k-2\ell-1}$.

Example: $k = 2048$ and $\ell = 256$.

The key generation algorithm of the RSA PSS (Probabilistic Signature Scheme) $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is the usual one:

Alg $\mathcal{K}$
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\mathrm{rsa}}$
Return $((N, e), (N, d))$

So $vk = (N, e)$ is the verification key and $sk = (N, d)$ is the signing key.

RSA PSS signing and verifying

Alg $\mathcal{S}_{(N,d)}(M)$
$r \xleftarrow{\$} \{0,1\}^{\ell}$
$w \leftarrow \mathbf{H}_1(M \,\|\, r)$
$r^* \leftarrow \mathbf{H}_2(w) \oplus r$
$y \leftarrow 0 \,\|\, w \,\|\, r^* \,\|\, \mathbf{H}_3(w)$
$x \leftarrow y^d \bmod N$
Return x

Alg $\mathcal{V}_{(N,e)}(M, x)$
$y \leftarrow x^e \bmod N$
$b \,\|\, w \,\|\, r^* \,\|\, P \leftarrow y$
$r \leftarrow r^* \oplus \mathbf{H}_2(w)$
If $(\mathbf{H}_3(w) \neq P)$ then return 0
If $(b = 1)$ then return 0
If $(\mathbf{H}_1(M \,\|\, r) \neq w)$ then return 0
Return 1

The message space is $\{0,1\}^*$, and $M \in \{0,1\}^*$ is the message. The signature is $x \in \mathbb{Z}_N^*$.

PSS standardization and usage

- RSA PKCS#1 v2.1, v2.2 / RFC 8017
- IEEE P1363a
- ANSI X9.31
- RFC 3447
- ISO/IEC 9796-2

Plan

Motivation and definitions

RSA signatures

Using hash functions

Full Domain Hash

RSA PSS (Probabilistic Signature Scheme)

DLog-based signature schemes

Schnorr signature scheme

Let $G = \langle g \rangle$ be a cyclic group whose order m is a prime number. Let $\mathbf{H} : \{0,1\}^* \to \mathbb{Z}_m$. The Schnorr signature scheme $\mathcal{DS} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is defined via

Alg $\mathcal{K}$	**Alg** $\mathcal{S}^{\mathbf{H}}_x(M)$	**Alg** $\mathcal{V}^{\mathbf{H}}_X(M,(R,s))$
$x \xleftarrow{\$} \mathbb{Z}_m$	$r \xleftarrow{\$} \mathbb{Z}_m$; $R \leftarrow g^r$	If $(R \notin G)$ then return 0
$X \leftarrow g^x$	$c \leftarrow \mathbf{H}(R \Vert M)$	$c \leftarrow \mathbf{H}(R \Vert M)$
Return (X, x)	$s \leftarrow (r + cx) \bmod m$	If $(g^s = RX^c)$ then return 1
	Return (R, s)	Else return 0

Above, $vk = X$ is the verification key and $sk = x$ is the signing key.

The message space is $\{0,1\}^*$, and $M \in \{0,1\}^*$ is the message. The signature is $(R, s) \in G \times \mathbb{Z}_m$.

Correctness: If $(R, s) \xleftarrow{\$} \mathcal{S}_x(M)$ then

$$g^s = g^{r+cx} = g^r (g^x)^c = RX^c \ .$$

EdDSA signature scheme

EdDSA [BDLSY12] is a Schnorr-based signature scheme over an elliptic curve group.

Signing key sk is a random string of length a parameter b. It is expanded into a $2b$-bit string $x_1 \| x_2$. A clamping function is applied to x_1 to get the Schnorr signing key $x \in \mathbb{Z}_m$.

Signing is made deterministic by setting r to a hash of x_2 and the message.

There are several variants of the scheme.

These schemes are widely standardized, including RFC 8032 and FIPS 186-5. The scheme is used in many places including OpenSSH and GnuPG.

DSA

Another class of DL-based signatures evolved from the El Gamal signature scheme.

DSA (Digital Signature Algorithm) is a version over a group $\mathbb{Z}_p^*$ where p is a prime. It was proposed by NIST in 1991 and is in the FIPS 186-4 standard. However a draft of FIPS 186-5 indicates approval may not continue.

ECDSA (Elliptic Curve Digital Signature Algorithm) is a DSA variant over an elliptic curve group. It is also in FIPS 186-4.

Randomization in signatures

We say a signature scheme is randomized if its signing algorithm is randomized.

Randomized signature schemes include PSS, Schnorr, EdDSA, DSA/ECDSA.

Re-using coins (random choices) across different signatures is not secure, but there are (other) ways to make these schemes deterministic without loss of security. Namely, determine the coins (randomness) of the signing algorithm as a hash of the signing key and message.

Unlike for encryption, in signatures, randomness is not necessary for security:

- It is possible to have secure deterministic signature schemes (RSA FDH is one example).
- However, in a randomized signature scheme, botching the randomness part can have dramatic consequences!

Random numbers, the wrong way

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Hashing in signatures

While randomness is not necessary in signatures, hashing, on the other hand, is a very useful tool.

In an exercise, a signature scheme with no hashing (or with a very short hash) is almost certainly going to be weak.

Example:

- We are in the RSA setting. $sk = (N, d)$, $vk = (N, e)$. N has k bits.
- Messages are bit strings of length $k - 128$, i.e. integers such that $0 \leq M < 2^{k-128}$.
- Signature of M is M^d mod N (M is at least 127 bits shorter than a typical element of $\mathbb{Z}_N^*$).
- A signature σ is valid only if σ^e mod $N < 2^{k-128}$.

Weakness?

Hashing in signatures

While randomness is not necessary in signatures, hashing, on the other hand, is a very useful tool.

In an exercise, a signature scheme with no hashing (or with a very short hash) is almost certainly going to be weak.

Example:

- We are in the RSA setting. $sk = (N, d)$, $vk = (N, e)$. N has k bits.
- Messages are bit strings of length $k - 128$, i.e. integers such that $0 \leq M < 2^{k-128}$.
- Signature of M is M^d mod N (M is at least 127 bits shorter than a typical element of $\mathbb{Z}_N^*$).
- A signature σ is valid only if σ^e mod $N < 2^{k-128}$.

Weakness? Pretty much the same as plain RSA. $(1, 1)$ is a valid signature, and so is the product of the signatures of two short messages.

Other example

- We are in the RSA setting. $sk = (N, d)$, $vk = (N, e)$. N has k bits.
- Messages are bit strings of length $k - 144$, i.e. integers such that $0 \leq M < 2^{k-144}$.
- Signature of M is $(\mathrm{CRC}_{16}(M) \times 2^{k-1-16} + M)^d \bmod N$.
- A signature σ is valid only if the high 16 bits are the CRC of the low $k - 144$ bits, and the middle 128 bits are zero.

Weakness?

Other example

- We are in the RSA setting. $sk = (N, d)$, $vk = (N, e)$. N has k bits.
- Messages are bit strings of length $k - 144$, i.e. integers such that $0 \leq M < 2^{k-144}$.
- Signature of M is $(\mathrm{CRC}_{16}(M) \times 2^{k-1-16} + M)^d \bmod N$.
- A signature σ is valid only if the high 16 bits are the CRC of the low $k - 144$ bits, and the middle 128 bits are zero.

Weakness? A CRC is not a hash function! And 16 bits are too few anyway. It is easy for the adversary to find (short) messages whose CRC is zero, and whose product also has zero CRC.

The signature universe

Aggregate signatures, anonymous signatures, blind signatures, chameleon signatures, convertible undeniable signatures, delegatable signatures, forward-secure signatures, functional signatures, fuzzy signatures, group signatures, homomorphic signatures, identity-based signatures, invariant signatures, key-homomorphic signatures, leakage-resilient signatures, list signatures, malleable signatures, multi-signatures, online/offline signatures, partially-blind signatures, policy-based signatures, proactive signatures, redactable signatures, rerandomizable signatures, ring signatures, sanitizable signatures, structure-preserving signatures, threshold signatures, transitive signatures, undeniable signatures, unique signatures, ...