# Introduction to Cryptology 11.2 - Digital Signatures

#### Federico Pintore

Mathematical Institute, University of Oxford (UK)



Michaelmas term 2020

#### Overview

Digital signatures provide integrity and authenticity in the public-key setting.

Public-key analogue of MACs.

### Overview

Digital signatures provide integrity and authenticity in the public-key setting.

Public-key analogue of MACs.

A concrete application: digital signatures allow clients to verify that software updates are authentic.

- An update is signed by the company using their secret key;
- each client can verify the authenticity of the update by verifying the signature against the company's public key.

If a signature  $\sigma$  on a message m is verified correctly against a given public key PK, it ensures that:

- the message was indeed sent by the *owner* of the public key;
- the message was not modified in transit.

### **Digital signatures and MACs**

- Key distribution and key management are hugely simplified.
- Signatures are publicly verifiable, therefore they are transferable.
- Signers cannot deny having signed a message (non-repudiation).
- MACs produce tags that are shorter than signatures, and they are more efficient to generate/verify .

#### **Digital signature schemes**

A digital signature scheme S = (KeyGen, Sign, Verify) consists of three PPT algorithms:

- (PK, SK) ← KeyGen(n): on input a security parameter n, it returns a public key PK and its matching secret key SK.
- $\sigma \leftarrow \text{Sign}(SK, m)$ : it takes a secret key SK and a message m from the message space  $\mathcal{M}$ , and returns a signature  $\sigma$ .
- 1/0 ← Verify(PK, m, σ): a deterministic algorithm that, on input a public key PK, a message m and a signature σ, returns either 1 (valid signature) or 0 (invalid signature).

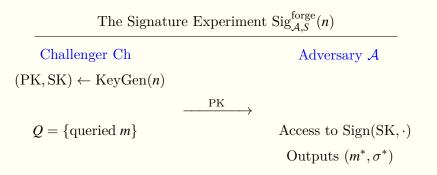
<u>Correctness</u>: for every  $m \in \mathcal{M}$ , and except with negligible probability over (PK, SK)  $\leftarrow$  KeyGen(n), it holds

Verify(PK, m, Sign(SK, m)) = 1.

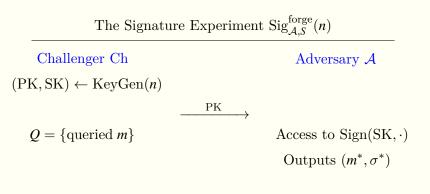
## Unforgeability

The Signature Experiment  $\operatorname{Sig}_{\mathcal{A},S}^{\operatorname{forge}}(n)$ 

## Unforgeability



### Unforgeability



 $\mathcal{A}$  wins the game, i.e.  $\operatorname{Sig}_{\mathcal{A},S}^{\operatorname{forge}}(n) = 1$ , if  $m^* \notin Q$  and  $\operatorname{Verify}(\operatorname{PK}, m^*, \sigma^*) = 1$ .

#### Definition

A signature scheme S = (KeyGen, Sign, Verify) is existentially unforgeable under an adaptive chosen-message attack, if for every PPT adversaries A, it holds

$$\Pr(\operatorname{Sig}_{\mathcal{A},S}^{\operatorname{forge}}(n) = 1) \le \operatorname{negl}(n).$$

### Hash-and-Sign Paradigm

Let S = (KeyGen, Sign, Verify) be a digital signature scheme for messages of length  $\ell(n)$ , and  $(\text{KeyGen}_H, H)$  a hash function with output length  $\ell(n)$ .

The signature scheme S' = (KeyGen', Sign', Verify') for messages of arbitrary length is defined as follows:

- PK, SK) ← KeyGen'(n): it runs KeyGen and KeyGen<sub>H</sub> on input a security parameter n, obtaining a pair of keys (PK', SK') and a key s. It outputs PK := (PK', s) and SK := (SK', s).
- ▶  $\sigma \leftarrow \text{Sign}'(\text{SK}, m \in \{0, 1\}^*)$ : it takes a secret key (SK', s) and a message m, and returns  $\sigma := \text{Sign}(\text{SK}', H^s(m))$ .
- 1/0 ← Verify'(PK, m, σ): on input a public key (PK', s), a message m and a signature σ, it and outputs 1 if Verify(PK', H<sup>s</sup>(m), σ) = 1, 0 otherwise.

#### Hash-and-Sign Paradigm

#### Theorem

If *S* is an existentially unforgeable digital signature scheme for messages of length  $\ell(n)$  and  $(\text{KeyGen}_H, H)$  is a collision-resistant hash function with output length  $\ell(n)$ , then *S'* is an existentially unforgeable digital signature scheme for arbitrary-length messages.

## **Further Reading**

Carlisle Adams and Steve Lloyd.

Understanding PKI: concepts, standards, and deployment considerations.

Addison-Wesley Professional, 2003.

- Dan Boneh, Ben Lynn, and Hovav Shacham.
  Short signatures from the Weil pairing.
  Journal of cryptology, 17(4):297–319, 2004.
  - Tim Dierks.

The transport layer security (TLS) protocol version 1.2. 2008.

Carl Ellison and Bruce Schneier.

Ten risks of PKI: What you're not being told about public key infrastructure.

Comput Secur J, 16(1):1–7, 2000.

### Further Reading

#### Amos Fiat and Adi Shamir.

How to prove yourself: Practical solutions to identification and signature problems.

In Advances in Cryptology—CRYPTO'86, pages 186–194. Springer, 1987.

 Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov.
 The most dangerous code in the world: validating SSL certificates in non-browser software.
 In Proceedings of the 2012 ACM conference on Computer and communications security, pages 38–49. ACM, 2012.

## Further Reading III

#### Hugo Krawczyk.

Cryptographic extraction and key derivation: The HKDF scheme.

In Annual Cryptology Conference, pages 631–648. Springer, 2010.

Hugo Krawczyk, Kenneth G Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Advances in Cryptology–CRYPTO 2013, pages 429–448. Springer, 2013.

#### Leslie Lamport.

Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International Palo Alto, 1979.