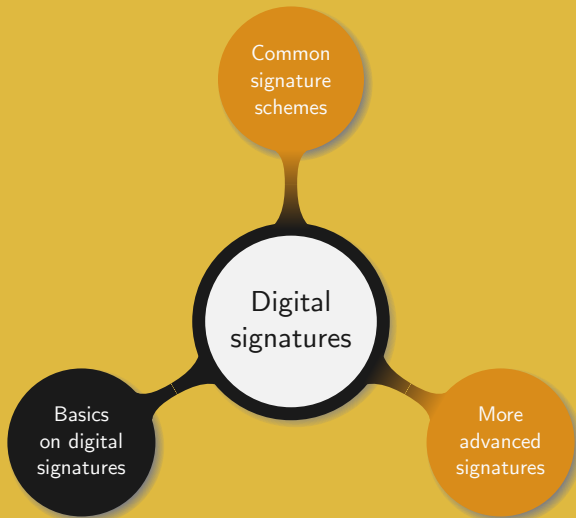


# Introduction to Cryptography

## 5. Digital signatures

Manuel – Summer 2022



Middle age:

- Document sealed with a wax imprint of an insignia
- Nobody can reproduce the insignia

Modern time:

- Sign credit card slip
- Compare to the signature at the back of the credit card

Reusing a signature:

- Photocopy
- Cut and paste
- Highly noticeable

Signing an electronic document:

- Digitalize the signature
- Paste it on the electronic document

Reusing a signature:

- Copy and paste on any document
- Anybody can do it
- Signature is not specific to an individual

Basic idea for a solution:

- Prevent the signature from being separated from its message
- Signature must be easily verified

Setup for signatures:

- Message to encrypt is not necessarily secret
- A message might be encrypted after being signed

The signature must be:

- Tied to the signer and to the message being signed
- Easy to verify by anybody
- Hard to forge

*Similar to public key cryptography*

Similar to attacks on encryption schemes (1.11) we define attacks on signature schemes:

- Key-only attack: Eve has only access to the public key
- Known message attack: Eve has a list of previously signed messages
- Chosen message attack: Eve chooses the messages to be signed

Eve's goal when attacking signature schemes:

- Selective forgery: Eve is given a message and is able to sign it without being given the private key
- Existential forgery: Eve can find a pair  $\langle \text{message}, \text{signature} \rangle$  without being given the private key

Drawback:

- Public key cryptography primitives are used
- Signing a whole message  $m$  is then slow

Solution: sign the hash of  $m$  using a public hash function

Benefits:

- Faster to generate
- Smaller to store or send
- Conveys the same knowledge as  $m$  itself

Given a hash function  $h$ , denote the signature of the hash of a message  $m$  by  $\text{sig}(h(m))$

## Existential forgery:

- Using known message attack:
  - ① Get a pair  $\langle m, \text{sig}(h(m)) \rangle$
  - ② Compute  $h(m)$  and attempt to find  $m'$  such that  $h(m) = h(m')$
  - ③ Considered impossible if  $h$  is second pre-image resistant
- Using chosen message attack:
  - ① Find two message  $m$  and  $m'$  such that  $h(m) = h(m')$
  - ② Persuade the signer to sign  $m$
  - ③ Attach  $\text{sig}(h(m)) = \text{sig}(h(m'))$  to  $m'$
  - ④ Considered impossible if  $h$  is collision resistant
- Using key-only attack:
  - ① Take a signature scheme, without hash function, which is vulnerable to existential forgery using key-only attack
  - ② Compute a signature on  $h(m)$  for some unknown  $m$
  - ③ Determine such an  $m$
  - ④ Considered impossible if  $h$  is pre-image resistant

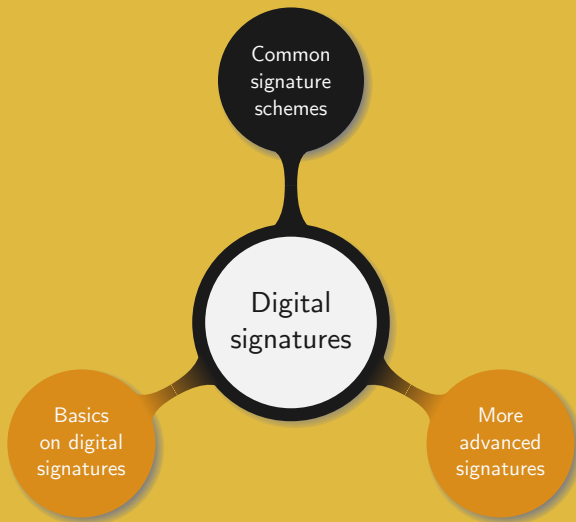


In the previous chapter we investigated the birthday paradox and illustrated how Eve could use this attack to cheat Alice when signing a contract (4.15).

Such an attack can be conducted as soon as the hash is used in place of the whole document. Therefore Alice should be careful and not sign the document. She should rather slightly alter it, for instance by adding a comma or space.

The document being different from the original its hash will be a totally different value. Hence, Eve cannot append Alice signature to the fraudulent contract.

Eve is then defeated and Alice can enjoy a nice contract.



## Initial setup:



- $p, q$  two primes
- $n = pq$  and  $\varphi(n)$
- $e, d$  such that  $ed \equiv 1 \pmod{\varphi(n)}$

- The  $n$  from Bob
- The  $e$  from Bob



## Signature:



- Compute  $s \equiv m^d \pmod{n}$
- Share  $m$  and  $s$

## Verification:

- The message  $m$  from Bob
- Compute  $m' \equiv s^e \pmod{n}$
- Compare  $m'$  to  $m$



Reusing a signature:

- Given a signature  $s$  with its message  $m$
- Impossible to sign  $m'$  using  $s$  since  $s^e \not\equiv m' \pmod{n}$

Generating a signature:

- Given a message  $m$  find  $s$  such that  $s^e \equiv m \pmod{n}$
- This is exactly solving the RSA problem (3.51)

Generating a message:

- Given a signature  $s$  generate a message  $m \equiv s^e \pmod{n}$
- It is very unlikely that  $m$  is meaningful

## Initial setup:



- $G$  a group of prime order  $p$
- $\alpha$  a generator of  $G$
- $x$  a secret integer
- $G$  from Bob
- $\alpha$  from Bob
- $\beta \equiv \alpha^x \bmod p$  from Bob



## Signature:



- Select a random  $k$ , with  $\gcd(k, p-1) = 1$
- Compute  $r \equiv \alpha^k \bmod p$
- Compute  $s \equiv k^{-1}(m - xr) \bmod (p-1)$

## Verification:

- The triple  $\langle m, r, s \rangle$
- Compute

$$v = \beta^r r^s \equiv \alpha^{xr} \alpha^{k \cdot k^{-1}(m - xr)} \equiv \alpha^m \bmod p$$

- The signature is valid only if  $v \equiv \alpha^m \bmod p$



Example. Set  $p = 467$ ,  $\alpha = 2$  and  $x = 127$ . Then  $\beta = 2^{127} \equiv 132 \pmod{467}$ . The variable  $x$  is kept secret, all the others are publicly known.

Signing the message  $m = 100$ :

- Randomly choose  $k = 213$  and keep it since  $\gcd(213, 466) = 1$
- Compute  $r = 2^{213} \equiv 29 \pmod{467}$
- As  $k^{-1} \equiv 431 \pmod{466}$ ,  $s = 431 \cdot (100 - 127 \cdot 29) \equiv 51 \pmod{466}$

To verify the signature  $\langle 100, 29, 51 \rangle$ , anyone can compute both:

- $132^{29} \cdot 29^{51} \equiv 189 \pmod{467}$
- $2^{100} \equiv 189 \pmod{467}$

First we notice that if  $x$  is discovered by an attacker, he can signed any document.

Then we observe that given only a message  $m$  he can try to:

- Find  $s$  such that

$$\beta^r r^s \equiv \alpha^m \pmod{p}. \quad (5.1)$$

This can be rewritten  $r^s \equiv \beta^{-r} \alpha^m \pmod{p}$ , and finding  $s$  means solving the DLP.

- Set  $s$  and solve eq. (5.1) for  $r$ . No feasible solution is known.
- Find  $r$  and  $s$  simultaneously. It is not known how to do it, but there is no prove that it is impossible to do.

Note that  $k$  must remain secret otherwise it is simple to recover  $x$ . Indeed if  $\gcd(r, p-1) = 1$ , then  $x \equiv (m - ks)r^{-1} \pmod{p-1}$ .

*Generating a message and its signature only knowing the public key*

Let  $i$  and  $j$  be two integers such that  $0 \leq i, j \leq p - 2$ . Define  $r$  as  $\alpha^i \beta^j \bmod p$ . Then  $\alpha^m$  can be expressed as

$$\alpha^m \equiv \beta^r \left( \alpha^i \beta^j \right)^s \bmod p.$$

Rearranging the different terms yields  $\alpha^{m-is} \equiv \beta^{r+js} \bmod p$ . This congruence is clearly true if both  $m - is$  and  $r + js$  are 0 mod  $(p - 1)$ .

Assuming  $\gcd(j, p - 1) = 1$ , we can determine  $m$  and  $s$  from the two previous equations. Therefore, by construction the signature

$$\langle m, r, s \rangle = \langle -rij^{-1} \bmod (p - 1), \alpha^i \beta^j \bmod p, -rj^{-1} \bmod (p - 1) \rangle$$

is a valid signature. Note that  $m$  is very unlikely to be meaningful.



Example. Set  $p = 467$ ,  $\alpha = 2$  and  $\beta = 132$ . Select  $i = 99$  and  $j = 179$ , and then  $j^{-1} \equiv 151 \pmod{466}$ .

The signature is defined by  $\langle m, r, s \rangle$  with

$$\begin{cases} r \equiv 2^{99} \cdot 132^{179} & \equiv 117 \pmod{467} \\ s \equiv -117 \cdot 151 & \equiv 41 \pmod{466} \\ m \equiv 99 \cdot 41 & \equiv 331 \pmod{466} \end{cases}$$

The verification is given by

$$132^{117} \cdot 117^{41} \equiv 303 \equiv 2^{331} \pmod{467}$$

Given a valid signature  $\langle m, r, s \rangle$  an attacker can construct and sign various other messages.

Generate  $h, i$ , and  $j$  such that  $\gcd(hr - js, p - 1) = 1$ . Then the triple  $\langle m', r', s' \rangle$  defines a valid signature if

$$\begin{cases} r' \equiv r^h \alpha^i \beta^j \pmod{p} \\ s' \equiv sr'(hr - js)^{-1} \pmod{p-1} \\ m' \equiv r'(hm + is)(hr - js)^{-1} \pmod{p-1} \end{cases}$$

Again this method leads to an existential forgery but cannot be modified into selective forgery. As such those two attacks represent no real threat for Elgamal signatures.

Let  $\langle m_1, r_1, s_1 \rangle$  and  $\langle m_2, r_2, s_2 \rangle$  be the two signatures. If they are generated using a common  $k$ , then  $r_1 = r_2 = r = \alpha^k \bmod p$  and

$$\begin{cases} \beta^r r^{s_1} & \equiv \alpha^{m_1} \bmod p \\ \beta^r r^{s_2} & \equiv \alpha^{m_2} \bmod p. \end{cases}$$

Thus  $\alpha^{m_1 - m_2} \equiv \alpha^{k(s_1 - s_2)} \bmod p$ , and from corollary 3.18 we get

$$m_1 - m_2 \equiv k(s_1 - s_2) \bmod (p - 1).$$

Since this congruence has  $d = \gcd(s_1 - s_2, p - 1)$  solutions (lemma 4.9) it is simple to test all of them and recover  $k$ . Once  $k$  is known  $x$  can be recovered as noticed on slide 5.15, and signatures can be forged at will.

### Digital Signature Algorithm (DSA):

- Proposed in 1991 by the NSA
- Adopted as a standard in 1994
- Variant of Elgamal signature scheme
- As in Elgamal the hash of the message is signed
- SHA-1 is the historical choice, SHA-2 (SHA-3) is now recommended
- For a given security level DSA defines two lengths  $l_1$  and  $l_2$  for the DLP and the hash to feature a balanced security

## Initial setup:



- A prime  $q$ ,  $|q| = l_2$
- A prime  $p$ ,  $|p| = l_1$  and  $q \mid (p-1)$
- $g$  a generator of  $G = U(\mathbb{F}_p)$
- $\alpha \equiv g^{(p-1)/q} \bmod p$
- $x$  a secret integer
- $p$  from Bob
- $q$  from Bob
- $\alpha$  from Bob
- $\beta \equiv \alpha^x \bmod p$  from Bob



## Signature:



- Select a random  $k$ ,  $0 < k < q$
- Compute  $r \equiv (\alpha^k \bmod p) \bmod q$
- Compute  $s \equiv k^{-1}(m + xr) \bmod q$

## Verification:

- The triple  $\langle m, r, s \rangle$
- Compute  $v = (\alpha^{s^{-1}m \bmod q} \beta^{s^{-1}r \bmod q} \bmod p) \bmod q$
- The signature is valid only if  $v = r$



Observe how the verification works:

By definition of  $s$  we know that  $m \equiv (-xr + ks) \bmod q$ . This implies  $s^{-1}m \equiv (-xrs^{-1} + k) \bmod q$ . Therefore we can write

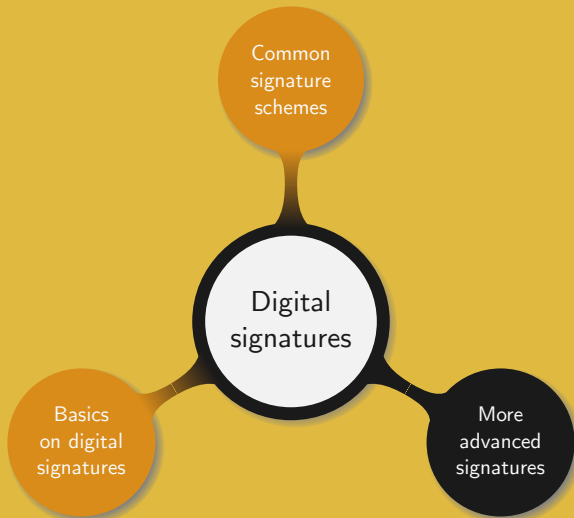
$$k \equiv s^{-1}m + xrs^{-1} \bmod q.$$

And we finally get

$$\begin{aligned} r &\equiv \alpha^k \bmod p \\ &\equiv \alpha^{s^{-1}m + xrs^{-1} \bmod q} \bmod p \\ &\equiv \alpha^{s^{-1}m \bmod q} \beta^{s^{-1}r \bmod q} \bmod p \\ &= v. \end{aligned}$$

## Why is DSA different from Elgamal?

- $r$  only “carries part of the information” on  $k$   
e.g. if  $l_1 = 3072$  and  $l_2 = 256$ , then about  $2^{2816}$  values mod  $p$  reduce to a same integer mod  $q$
- From the initial setup (slide 5.21),  $\alpha^q \equiv 1 \pmod{p}$ . Pohlig-Hellman attack (3.78) does not apply, since  $q$  is prime. Not even a little piece of information can be recovered.
- Verification step requires only two modular exponentiations vs. three in Elgamal case





Basic idea: sign a document without knowing its content

Typical setup: Bob made a new discovery and wants to record it publicly without unveiling it

Strategy: Bob gets his discovery signed by some known authority but without revealing or showing it the content

Danger: what is signed?

Initial setup:



- $p, q$  two primes
- $n = pq$  and  $\varphi(n)$
- $e, d$  such that  $ed \equiv 1 \pmod{\varphi(n)}$

- $n, e$  from Alice
- A random integer  $k$ ,  $\gcd(k, n) = 1$
- For a message  $m$  compute  $t \equiv k^e m$



Blind signature:



- $t$  from Bob
- Compute  $s \equiv t^d \pmod{n}$
- Send  $s$  to Bob

Message signature:

- Bob computes

$$\frac{s}{k} \equiv \frac{t^d}{k} \equiv \frac{k^{ed} m^d}{k} \equiv m^d \pmod{n}$$

- Bob later shares  $m$  and  $e$



## Remark.

- $k$  being random  $k^e \bmod n$  is also random and so is  $k^e m \bmod n$
- Alice cannot get any information on what she is signing
- The final value is the same as if Bob had gotten his message signed following the standard procedure
- Verification happens as in “regular RSA signatures”
- There is no need to keep  $d, p$  and  $q$

Primary goal: design a signature that cannot be verified without the cooperation of the signer

Secondary goals:

- Prevent the signer to disavow a previous signature
- Allow the signer to prove that a forged signature is a forgery

Applications: prevent the illegal distribution of documents without the approval of the author

Structure: composed of three algorithm: signature, verification, and disavowal

## Initial setup:



- $p$  and  $q$  two primes  $p = 2q + 1$
- $G$  a subgroup of  $\mathbb{F}_p^*$  of order  $q$
- $\alpha$  a generator of  $G$
- $x$  a secret integer

- $G$  from Bob
- $\alpha$  from Bob
- $\beta \equiv \alpha^x \bmod p$  from Bob



## Signature:



- A message  $m$  in  $G$
- Compute  $s \equiv m^x \bmod p$

## Verification:



- 3 Compute  $t \equiv r^{x^{-1} \bmod q} \bmod p$
- 4 Share  $t$  with Alice

- 1 Choose random  $e_1, e_2 \in \mathbb{F}_q^*$
- 2  $r \equiv s^{e_1} \beta^{e_2} \bmod p$
- 5 Valid if and only if  $t \equiv m^{e_1} \alpha^{e_2} \bmod p$



Remark. On a valid signature we have:

$$\begin{aligned}t &\equiv r^{x^{-1}} \bmod p \\ &\equiv s^{e_1 x^{-1}} \beta^{e_2 x^{-1}} \bmod p\end{aligned}$$

Noting that  $s \equiv m^x \bmod p$ , and  $\beta \equiv \alpha^x \bmod p$ , we get

$$t \equiv m^{e_1} \alpha^{e_2} \bmod p.$$

Example. Let  $p = 467$ , then 2 is a primitive element of  $\mathbb{F}_p^*$  and 4 is a generator of the group  $G$  of order 233. Taking  $x = 101$ ,  $\beta \equiv 4^{101} \equiv 449 \bmod 467$ .

Signing the message  $m = 119$  yields  $119^{101} \equiv 129 \bmod 467$ .

To verify the signature, randomly select  $e_1 = 38$  and  $e_2 = 397$ , then send  $r = 13$  while  $t = 9$  is replied. Finally test that 9 is congruent to  $119^{38} 4^{397} \bmod 467$ .

2-round verification:



Play the verification protocol using two random values  $e_1, e_2 \in \mathbb{F}_q^*$  and expect

$$t_1 \not\equiv m^{e_1} \alpha^{e_2} \pmod{p}$$



Re-play the verification protocol using two random values  $f_1, f_2 \in \mathbb{F}_q^*$  and expect

$$t_2 \not\equiv m^{f_1} \alpha^{f_2} \pmod{p}$$



Concludes that the signature is a forgery if and only if

$$\left(t_1 \alpha^{-e_2}\right)^{f_1} \equiv \left(t_2 \alpha^{-f_2}\right)^{e_1} \pmod{p}$$



Remark. The disavowal protocol has two goals:

- Convince Alice that an invalid signature is a forgery
- Prevent Bob from pretending that a valid signature is a forgery

If the signature is invalid then the verification fails. The question is then to know if Bob played a fair game, following the protocol when constructing  $t_1$  and  $t_2$ .

The last step, testing the congruence

$$\left(t_1 \alpha^{-e_2}\right)^{f_1} \equiv \left(t_2 \alpha^{-f_2}\right)^{e_1} \bmod p,$$

ensures Alice that Bob is not trying to disavow a valid signature.

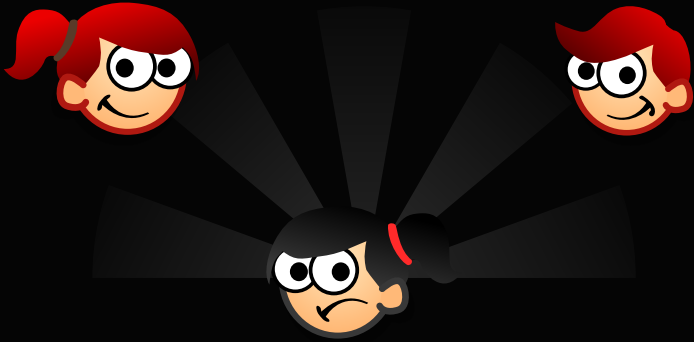


As investigated earlier (1.61), zero-knowledge proofs can be used to authenticate. In fact this can also be extended to signatures.

General strategy:

- Send at once all the committed values  $C_1, \dots, C_n$
- For a message  $m$  compute  $h$ , the hash of  $\langle C_1, \dots, C_n, m \rangle$
- Extract  $n$  bits,  $h_1, \dots, h_n$ , from  $h$  to represent the random requests
- Define the signature of  $m$  as  $\langle h_1, \dots, h_n, R_1, \dots, R_n \rangle$ , where  $R_i$  is the response to challenge  $h_i$  for the committed  $C_i$
- To ensure a proper security level  $n$  should be at least 128





Thank you!