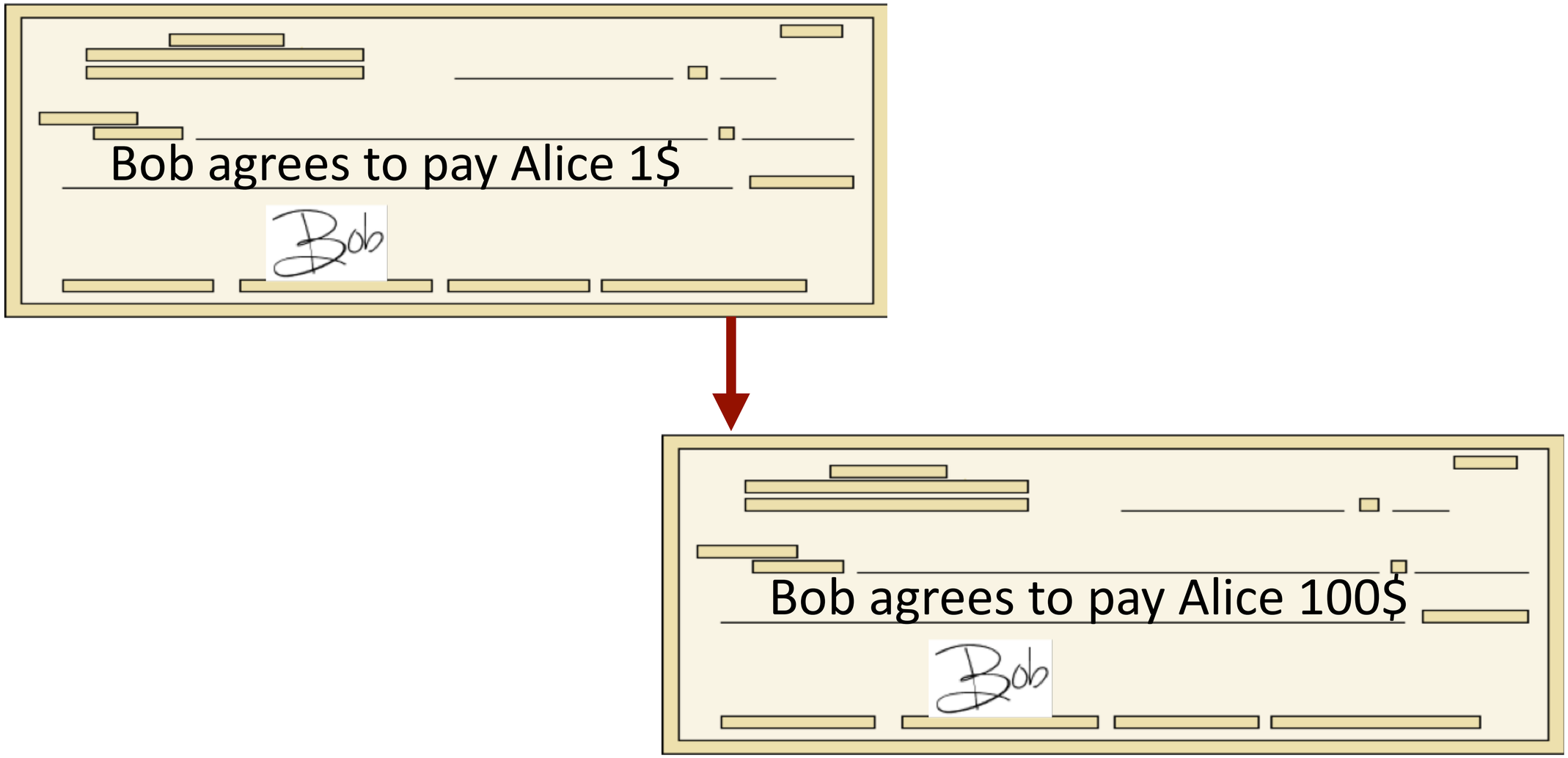
HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Introduction to Cryptography and Security

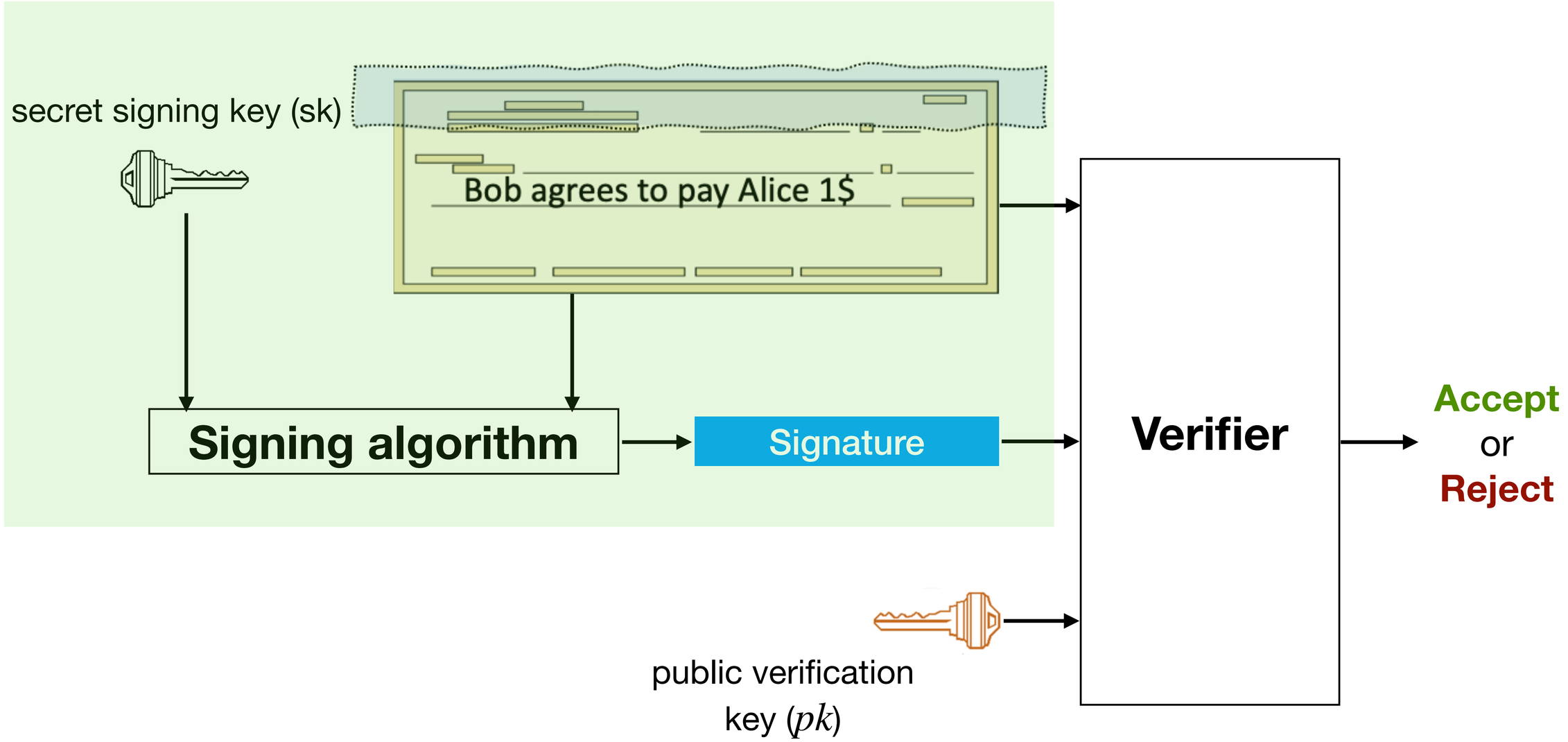
Digital Signatures

Contents

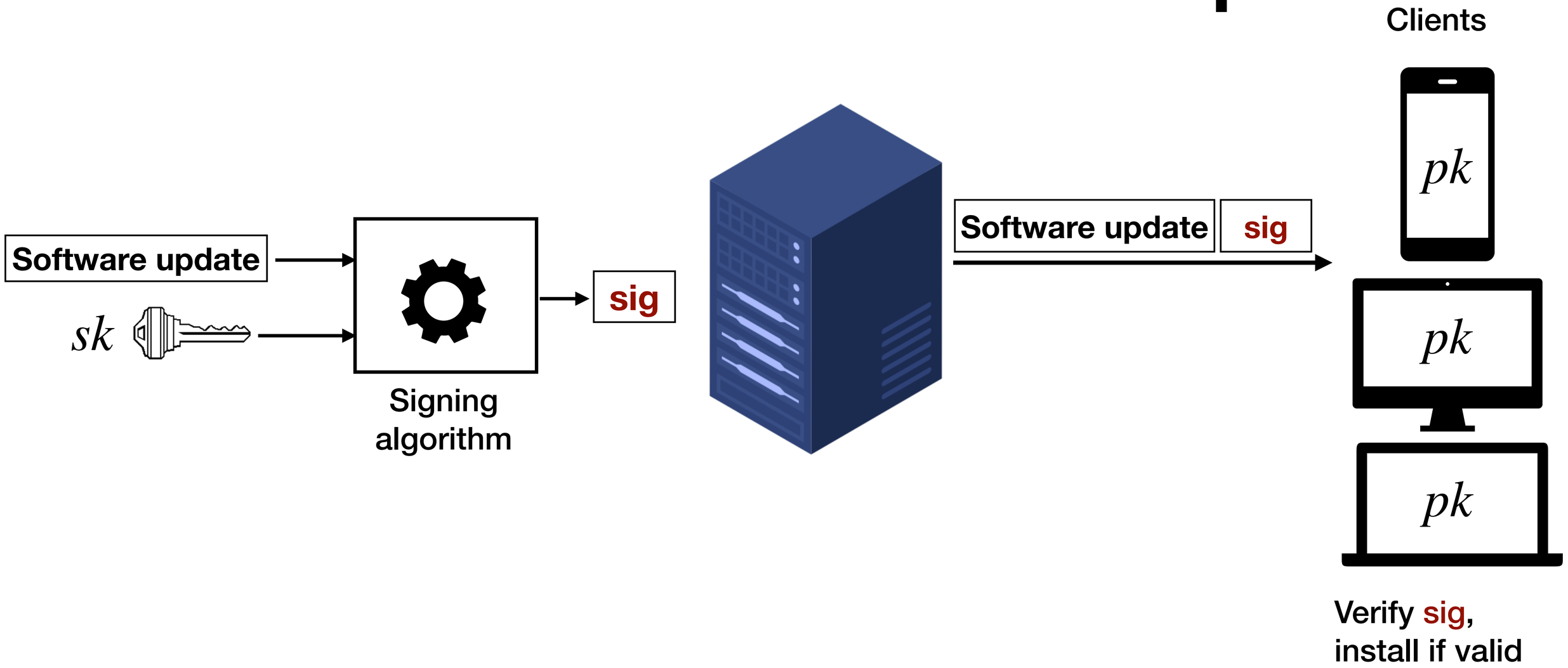
Physical signatures



Solution: make signature depend on document



A more realistic example



Digital signatures: syntax

Definition. A *signature scheme* $(\ Gen,\ S,\ V\ )$ is a triple of algorithms:

- $Gen()$ randomized algorithm outputs a key pair (pk, sk)
- $S(sk, m \in M)$ outputs signature σ
- $V(pk, m, \sigma)$ output '**accept**' or '**reject**'

Consistency

- for all (pk, sk) output by $Gen()$,
- and for all message $m \in M$, we have

$$V(pk,\ m,\ S(sk, m)) \ = \text{‘}\textbf{accept}\text{’}$$

Security

- Attacker's power: **Chosen message attack**

  for $m_1, m_2, \ldots, m_q$, attacker is given $\sigma_i = S(sk, m_i)$

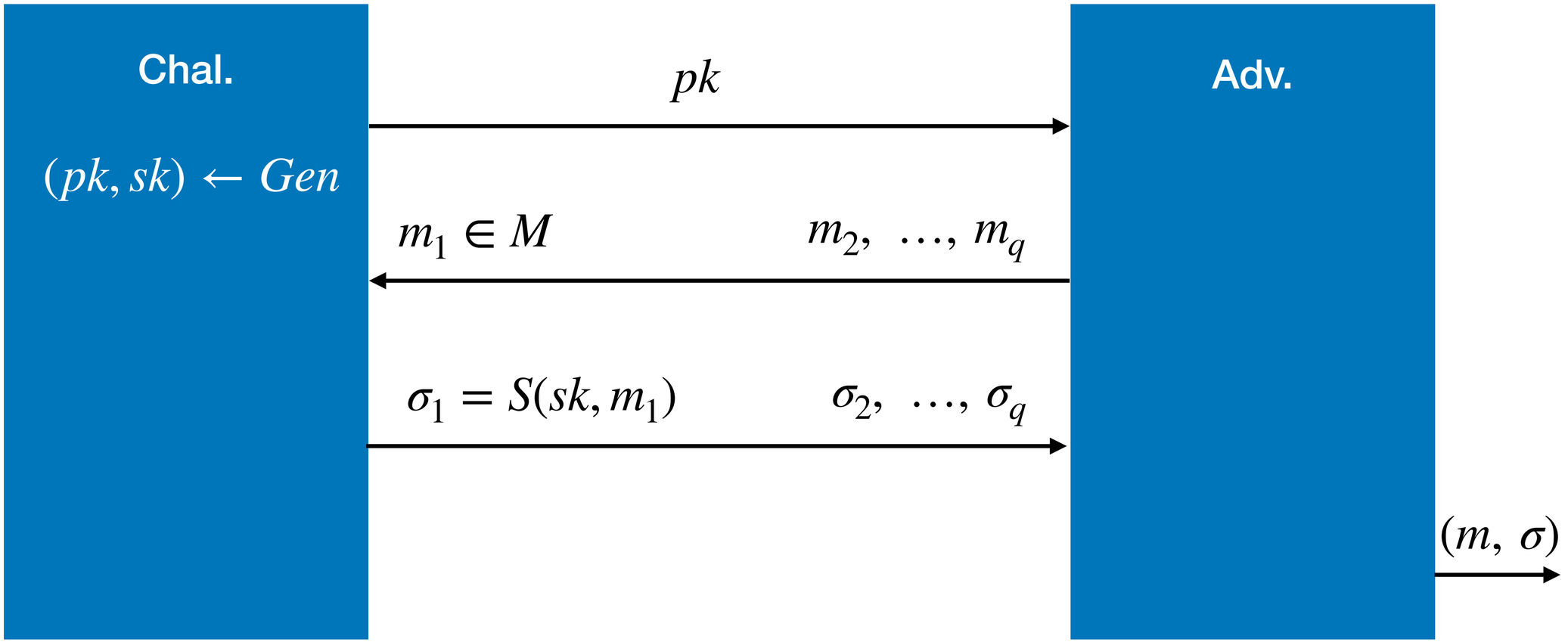
- Attacker's goal: **Existential forgery**

  produce some new valid message/sig pair (m, σ).

  $m \notin \{m_1, \ldots, m_q\}$

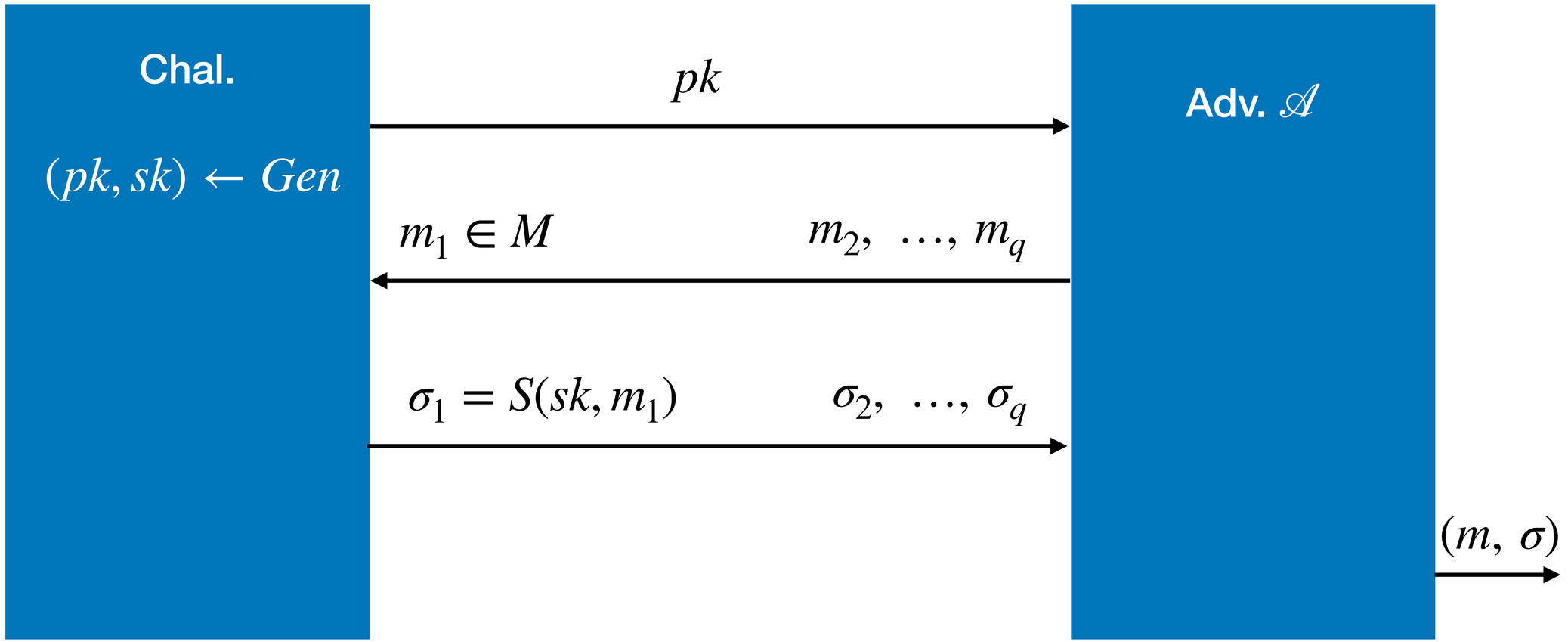
$\Rightarrow$ attacker cannot produce a valid sig. for a new message

Secure signature



Adv. **win** if $V(pk, \ m, \ \sigma) =$ '**accept**' and $m \notin \{m_1, \ldots, m_q\}$

Secure signature



Definition. A signature scheme is a secure if for all "efficient" $\mathscr{A}$:

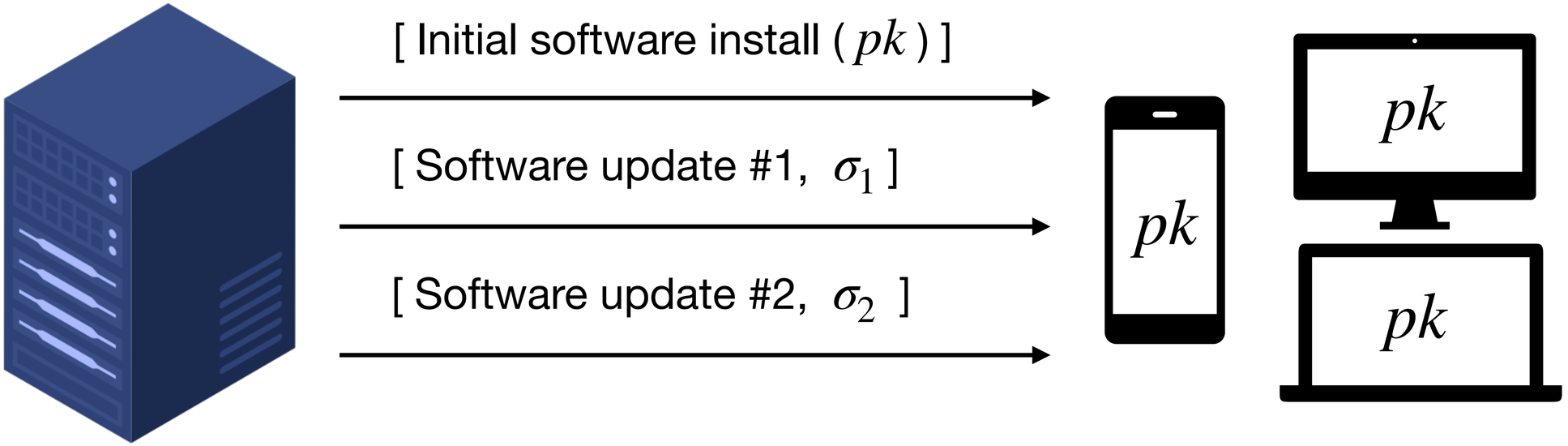
Pr[$\mathscr{A}$ **win**] is "*negligible.*"

Contents

- What is Digital Signature?
- **Applications**
- The RSA Digital Signature Scheme
- The Elgamal Digital Signature Scheme
- The Digital Signature Algorithm (DSA)

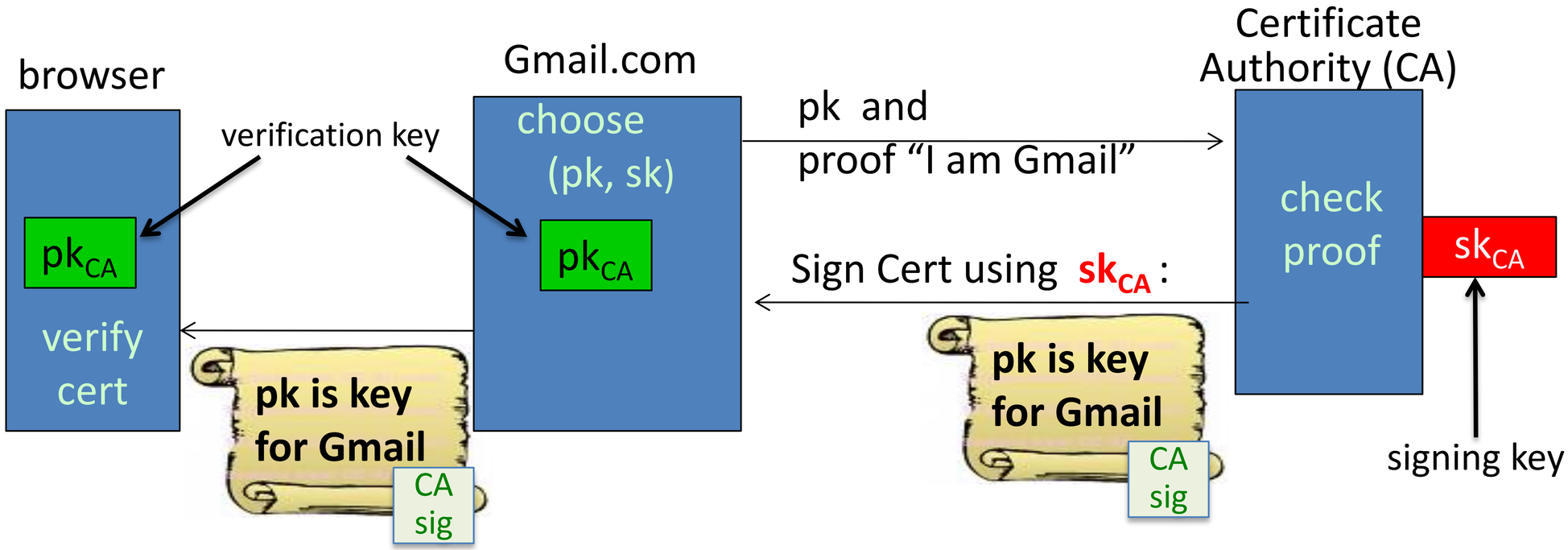
Code signing

- Software vendor signs code
- Clients have vendor's pk. Install software if signature verifies.



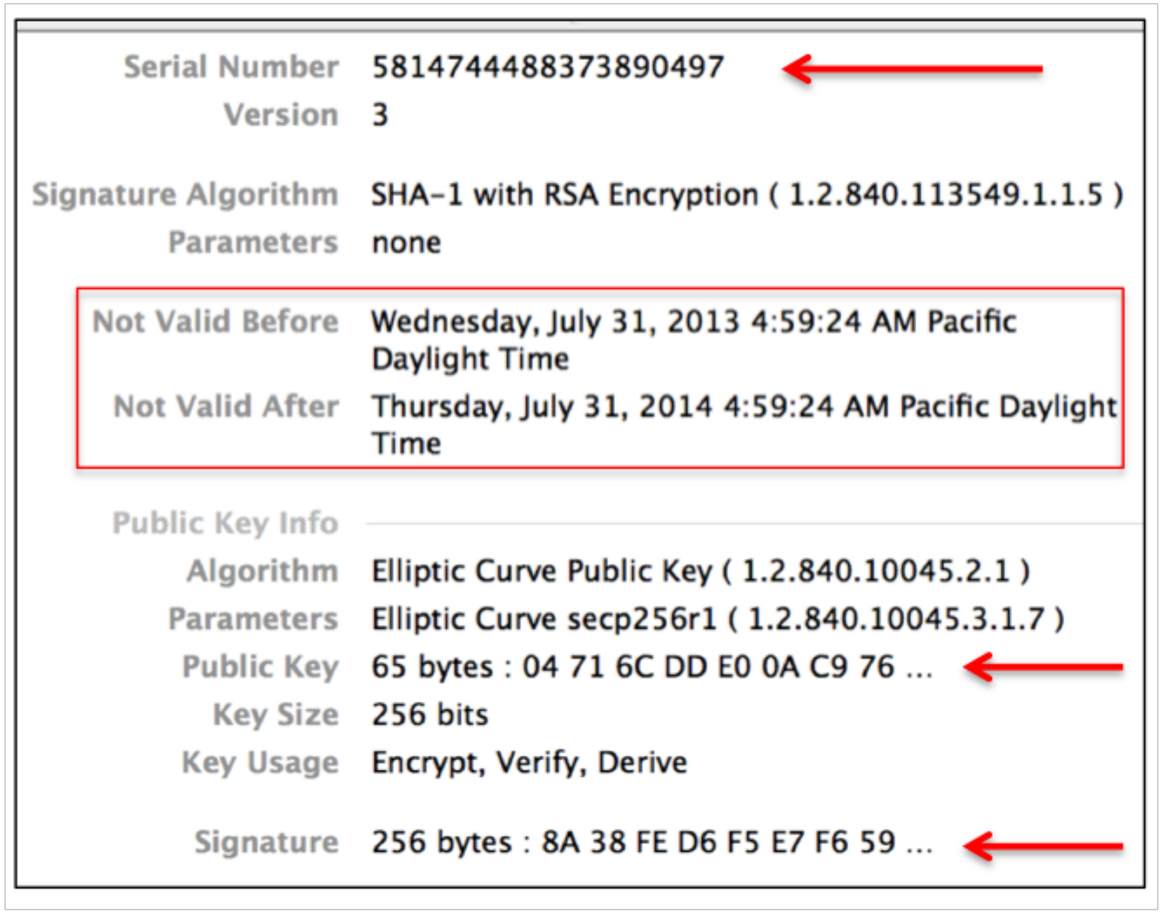
Important application: Certificates

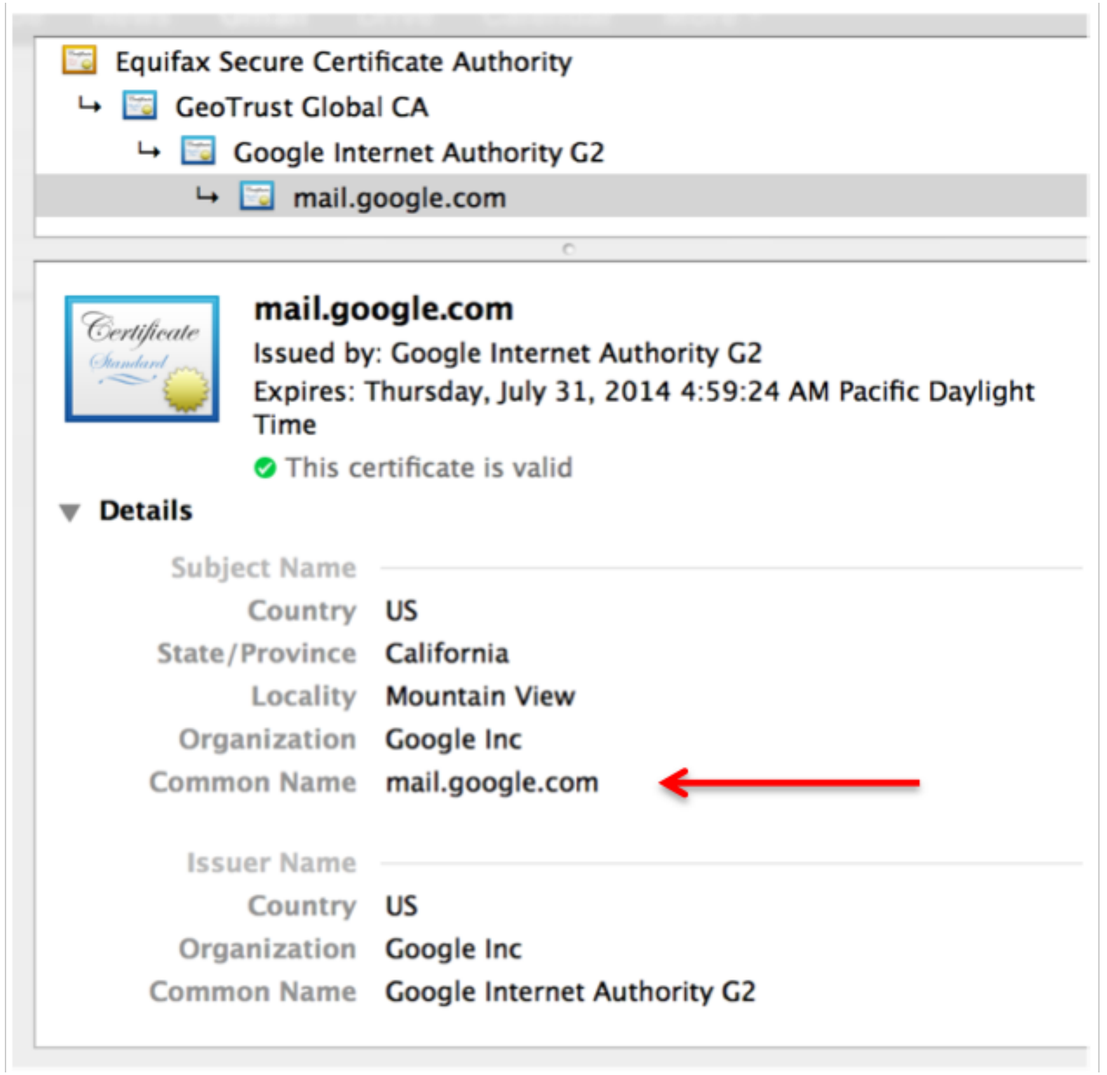
- **Problem:** browser needs server's public-key to setup a session key
- **Solution**: server asks trusted 3rd party (CA) to sign its public-key pk



Server uses Cert for an extended period (e.g. one year)

Certificates: example



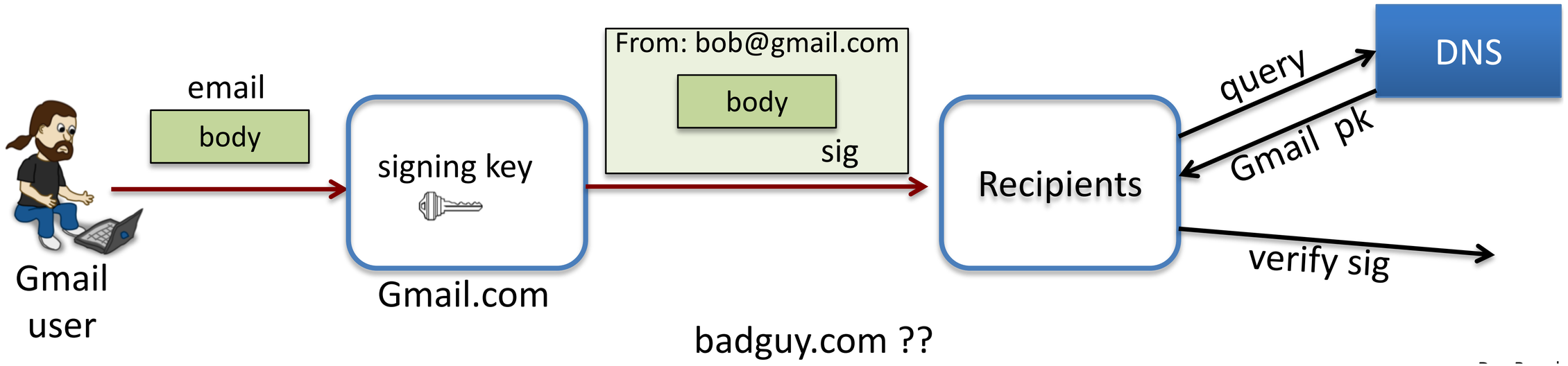


Signing email: DKIM (domain key identified mail)

- **Problem:** bad email claiming to be from someuser@gmail.com
  but in reality, mail is coming from domain badguy.com

  ⇒ Incorrectly makes gmail.com look like a bad source of email

- **Solution:** gmail.com sign every outgoing mail



When to use signatures

If **one** party signs and **one** party verifies: ***use a MAC***

- Often requires interaction to generate a shared key
- Recipient can modify the data and re-sign it before passing the data to a 3rd party

If **one** party signs and **many** parties verify: ***use a signature***

- Recipients cannot modify received data before passing data to a 3rd party (non-repudiation)

Review: three approaches to data integrity

1. **Collision resistant hashing:** need a read-only public space

2. **MACs:** vendor must compute a new MAC of software for every client
   ⇒ and must manage a long-term secret key
   (to generate a per-client MAC key)

3. **Digital signatures:** vendor must manage a long-term secret key

   - Vendor's signature on software is shipped with software

   - Software can be downloaded from an untrusted distribution site

Contents

- What is Digital Signature?
- Applications
- **The RSA Digital Signature Scheme**
- The Elgamal Digital Signature Scheme
- The Digital Signature Algorithm (DSA)

General method

- The idea was introduced by Diffie & Hellman in 1976
- Building a signature scheme from a deterministic public key crypto-system (E, D)

$$\sigma = S(sk, m) = D(sk, m)$$

$$V(pk, m, \sigma) = \begin{cases} 1 & \text{if } E(pk, \sigma) = m \\ 0 & \text{otherwise} \end{cases}$$

Textbook RSA

- Key Generation $Gen()$:
  - Choose $n = pq$ (p, q λ-bit random primes)
  - Choose e, d such that $ed = 1 \mod \varphi(n)$
  - $pk = (n, e)$ and $sk = (n, d)$
- Signature Generation: $S(sk, m) = m^d \mod n$
- Signature Verification: $V(pk, m, \sigma) = 1 \Leftrightarrow \sigma^e = m \mod n$

Example: Key generation

- Choose $p = 3$ and $q = 11$
- $n = p \cdot q = 33$
- $\varphi(n) = (3 - 1)(11 - 1) = 20$
- Choose $e = 3$
- $d = e^{-1} = 7 \mod 20$
- Output $(pk = 3,\ sk = 7)$

Example: Sign and Verify

Signature Generation

$S(sk = d = 7,\ m = 4)$:

- $\sigma = m^d \mod n$
  $= 4^7 = 16 \mod 33$

Signature Verification

$V(pk = e = 3,\ m = 4,\ \sigma = 16)$:

- $m' = \sigma^e = 16^3 = 4 \mod 33$
- Since $m = m'$ $\Rightarrow$ '**accept**'

Attack 1

- Can sign *specific* messages
  - E.g., easy to compute the e^{th} root of $m = 1$ or the cube root of $m = 8$

Attack 2

- Can sign “random” messages
  - Choose arbitrary σ ; set $m = [\sigma^e \mod n]$

Attack 3

- Can combine two signatures to obtain a third
  - Say σ_1, σ_2 are valid signatures on m_1, m_2
  - Then $\sigma = [\sigma_1 \cdot \sigma_2 \mod n]$ is a valid signature on the message $m = m_1 \cdot m_2$ because

$$(\sigma_1 \cdot \sigma_2)^e = \sigma_1^e \cdot \sigma_2^e = m_1 \cdot m_2$$

Hash & Sign

- $Gen()$: [...]
- $S(sk, m) = \mathsf{H}(m)^d \mod n$
- $V(pk, m, \sigma) = 1 \Leftrightarrow \sigma^e = \mathsf{H}(m) \mod n$

Intuition for security?

- Look at the three previous attacks…

  1. Not easy to compute the e^{th} root of $\mathrm{H}(1)$,…

  2. Choose σ but how do you find an m such that $\mathrm{H}(m) = \sigma^e \mod n$?
     $\Rightarrow$ Computing inverses of H should be hard

  3. $\mathrm{H}(m_1) \cdot \mathrm{H}(m_2) = \sigma_1^e \cdot \sigma_2^e = (\sigma_1 \cdot \sigma_2)^e \neq \mathrm{H}(m_1 \cdot m_2)$

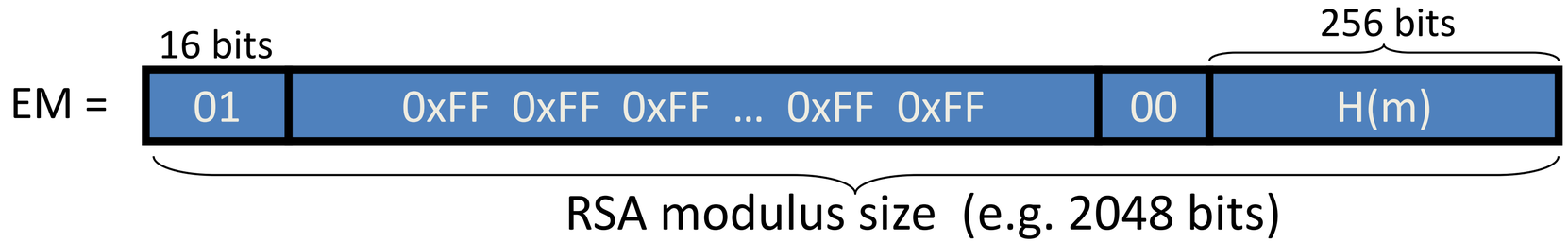
Security of RSA-FDH (Full Domain Hash)

- If the RSA assumption holds, and H is modeled as a random oracle (mapping onto $\mathbb{Z}_n^*$), then RSA-FDH is **secure**
- In practice, H is instantiated with a (modified) cryptographic hash function
  - Must ensure that the range of H is large enough!
  - A "good" hash function H :

$$\mathrm{H}(m) = n \text{ first bytes of } \mathrm{SHA256}(1\|m) \,\|\, \mathrm{SHA256}(2\|m) \,\|\, \cdots \,\|\, \mathrm{SHA256}(11\|m)$$

RSA in practice - PKCS1 v1.5

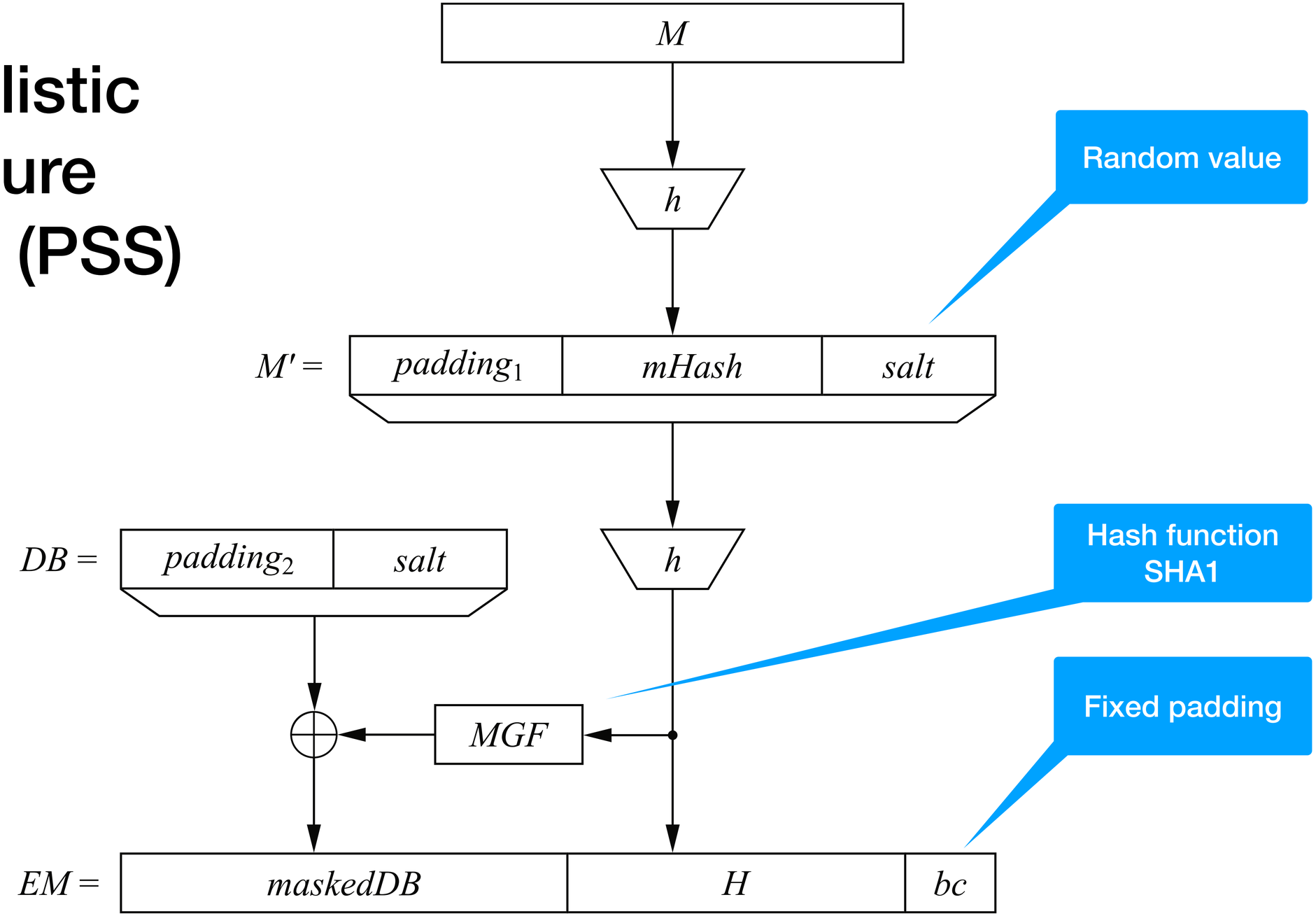
- Trapdoor permutation RSA : $pk = (n, e), sk = (n, d)$
- $S(sk, m \in M)$:



output $\sigma = (EM)^d \mod n$

- $V(pk, m \in M, \sigma)$: verify that $\sigma^e \mod n$ has correct format

Probabilistic Signature Standard (PSS)



Contents

- What is Digital Signature?
- Applications
- The RSA Digital Signature Scheme
- **The Elgamal Digital Signature Scheme**
- The Digital Signature Algorithm (DSA)

ElGamal Digital Signatures

- The paradigm

$$E(D(m)) = m$$

doesn't work for ElGamal, since ElGamal is not a trapdoor permutation (it is randomized.)

- The Elgamal digital signature is quite different from the Elgamal encryption scheme.

Textbook ElGamal: Key Generation

- $Gen()$:

    1. Choose a large prime p
    2. Choose a generator g of $\mathbb{Z}_p^*$ (or a subgroup of $\mathbb{Z}_p^*$)
    3. Choose a random integer $d \in \{2, 3, \ldots, p-2\}$;
    4. Output ($pk = g^d$, $sk = d$)

Textbook ElGamal: Signature Generation

- $S(sk = d, m)$:

1. Choose a random ephemeral key $k_E \in \{1, 2, \ldots, p-2\}$ such that

$$\gcd(k_E, p-1) = 1 \qquad (k_E^{-1} \mod p-1 \text{ exists})$$

2. Compute the signature parameters:

$$r = g^{k_E} \mod p,$$

$$s = \frac{(m - d \cdot r)}{k_E} \mod p-1$$

3. Output the signature $\sigma = (r, s)$

Textbook ElGamal: Signature Verification

- $V(pk = g^d,\ m,\ \sigma = (r, s))$:

    1. Compute the value

$$t = \left(g^d\right)^r \cdot r^s \mod p$$

    2. `if` $t = g^m \mod p$ `return ‘accept’ else ‘reject’`

Textbook ElGamal: Correctness

- $t = \left(g^d\right)^r \cdot r^s \mod p = \left(g^d\right)^r \cdot \left(g^{k_E}\right)^s \mod p$

$$= g^{d \cdot r + s \cdot k_E} \mod p$$

- According to Fermat's Little Theorem , the equality $g^m = t \mod p$ is equivalent to

$$m = \left(d \cdot r + s \cdot k_E\right) \mod p - 1$$

- from which the construction rule of the signature parameters s follows:

$$s = \frac{m - d \cdot r}{k_E} \mod p - 1$$

Example: Key and signature generation

Key generation $Gen()$

- choose $p = 29$
- choose $g = 2$
- Choose $sk = d = 12$
- $pk = g^d = 7 \mod 29$

Signature generation $S(sk = 12,\ m = 26)$**:**

- choose $k_E = 5$, note that $\gcd(5,28) = 1$
- $r = g^{k_E} = 2^5 = 3 \mod 29$
- $s = (m - d \cdot r) \cdot k_E^{-1} \mod p - 1$
  $= -10 \cdot 17 \mod 28$
  $= 26 \mod 28$
- Output signature $(3,\ 26)$

Example: Signature Verification

- $V(\ pk = g^d = 7,\ m = 26,\ \sigma = (3,26)\ )$:
  - $t = (g^d)^r \cdot r^s = 7^3 \cdot 3^{26} = 22 \mod 29$
  - $g^m = 2^{26} = 22 \mod 29$
  - Since $t = g^m \mod 29$ $\Rightarrow$ '**accept**'

Reuse of the Ephemeral Key k_E

- If the two messages m_1 and m_2 have the same ephemeral key k_E, then

$$s_1 = \frac{m_1 - dr}{k_E} \mod p - 1 \quad \text{and} \quad s_2 = \frac{m_2 - dr}{k_E} \mod p - 1$$

- Oscar can compute the ephemeral key

$$k_E = \frac{m_1 - m_2}{s_1 - s_2} \mod p - 1$$

- also compute the private key

$$d = \frac{m_1 - s_1 k_E}{r} \mod p - 1$$

Exercise

- Suppose that Oscar knows Bob's public key, which is given as:
  - $p = 29,\ g = 2,\ g^d = 7$
- Suppose that Bob signed two messages with the same ephemeral key k_E :
  - $[m_1,\ (r, s_1)] \ = \ [26,\ (3,26)]$
  - $[m_2,\ (r, s_2)] \ = \ [13,\ (3,1)]$
- Compute Bob's private key sk.

Question on Existential Forgery Attack

- In Textbook ElGamal, can you sign a “random” message? (Similar to the case of Textbook RSA)?

Existential Forgery Attack: Random message

- **Signature Generation**

  - select integers i, j where $\gcd(j, p-1) = 1$

  - compute signature:
    $r = g^i \cdot \left(g^d\right)^j \mod p$
    $s = -r \cdot j^{-1} \mod p-1$

  - compute message:
    $m = s \cdot i \mod p-1$

- **Signature Verification**

  - compute $t = (g^d)^r \cdot r^s \mod p$

  - since $t = g^m \mod p$
    $\Rightarrow$ valid signature!

Hash & Sign ElGamal

- Like RSA, the Hash and Sign scheme not only increases efficiency but also increases security.
- It helps against signature forgery attacks for “random” messages.
- The signing equation becomes:

$$s = \frac{\left(\mathrm{H}(m) \ - d \cdot r\right)}{k_E} \mod p - 1$$

Contents

- What is Digital Signature?
- Applications
- The RSA Digital Signature Scheme
- The Elgamal Digital Signature Scheme
- **The Digital Signature Algorithm (DSA)**

The Digital Signature Algorithm (DSA)

- It is a federal US government standard for digital signatures (DSS) and
- was proposed by the National Institute of Standards and Technology
- **Advantages :**
  - the signature is only 320 bit long
  - some of the attacks that can threaten the Elgamal scheme are not applicable.

Key Generation for DSA

- Generate a prime p with $2^{1023} < p < 2^{1024}$
- Find a prime divisor q of $p - 1$ với $2^{159} < q < 2^{160}$
- Find an element g with order $\text{ord}(g) = q$;
  i.e., g generates the subgroup with q elements
- Choose a random integer d with $0 < d < q$
- Compute $\beta = g^d$
- Output $pk = (p, q, g, \beta)$ and $sk = d$

DSA Signature Generation

- Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
- Compute $r = \left(g^{k_E} \mod p\right) \mod q$
- Compute $s = \left(\mathrm{H}(m) + d \cdot r\right) \cdot k_E^{-1} \mod q$

DSA Signature Verification

- Compute auxiliary value $w = s^{-1} \mod q$
- Compute auxiliary value $u_1 = w \cdot \mathrm{H}(m) \mod q$
- Compute auxiliary value $u_2 = w \cdot r \mod q$
- Compute $v = \left(g^{u_1} \cdot \left(g^{d} \right)^{u_2} \mod p \right) \mod q$
- The verification $V(pk,\ m,\ (r, s))$ follows from:

```
if  v = r mod q  return ‘accept’  else ‘reject’
```

Standardized parameter bit lengths and security levels for DSA

p	*q*	Signature	Hash output (min)	Security levels
1024	160	320	160	80
2048	224	448	224	112
3072	256	512	256	128

Example: DSA Key and Signature Generation

Key generation

- Choose $p = 59$
- Choose $q = 29$
- Choose $g = 3$
- Choose private key
  $sk = d = 7$
- Public key
  $$pk = g^d = 4 \mod 59$$

Signature generation for H$(m) = 26$

- Choose ephemeral key $k_E = 10$
- $$r = \left(3^{10} \mod 59\right) \mod 29$$
  $$= 20 \mod 29$$
- $$s = \left(26 + 7 \cdot 20\right) \cdot 3 \mod 29$$
  $$= 5 \mod 29$$

Example: DSA Signature Verification

$V(pk = 4,\ \mathbf{H}(m) = 26,\ (r = 20,\ s = 5))$**:**

- $w = 5^{-1} = 5 \mod 29$
- $u_1 = 6 \cdot 26 = 11 \mod 29$
- $u_2 = 6 \cdot 20 = 4 \mod 29$
- $v = \left(3^{11} \cdot 4^{4} \mod 59\right) \mod 29 = 20$
- Since $v = r \mod 29$ $\Rightarrow$ the signature is '**valid**'

Computation Aspects

- **Problem:**
  - Find a cyclic group $\mathbb{Z}_p^*$ with a bit length 1024 bit, and
  - which has a prime subgroup in the range of 2^{160}
- **General approach:**
  - find the 160 bit prime q and construct the larger prime p from it.

Prime Generation for DSA

- **Output**:
  - Two primes (p, q),

    where $2^{1023} < p < 2^{1024}$ and $2^{159} < q < 2^{160}$

    such that $p - 1$ is a multiple of q

Prime Generation for DSA

1. Find prime q with $2^{159} < q < 2^{160}$ using the Miller–Rabin algorithm
2. **for** $i = 1$ to 4096
   - generate random integer M with $2^{1023} < M < 2^{1024}$
   - $M_r = M \mod 2q$
   - $p - 1 = M - M_r$ // $p - 1$ is a multiple of $2q$
   - **if** p là số nguyên tố: **return** (p, q) // use Miller–Rabin primality test
3. Goto Step 1.

Quiz 1

- Let (Gen, S, V) be a signature scheme.
- Suppose an attacker is able to find $m_0 \neq m_1$ such that
  $V(pk, m_0, \sigma) = V(pk, m_1, \sigma)$ for all σ and keys. $(pk, sk) \leftarrow Gen()$
- Can this signature be secure?

  1. Yes, the attacker cannot forge a signature for either m_0 or m_1
  2. No, signatures can be forged using a chosen message attack
  3. It depends on the details of the scheme

Quiz 2

- Alice generates a (pk, sk) and gives pk to her bank.
- Later Bob shows the bank a message m=“pay Bob 100$” properly signed by Alice, i.e., $V(pk, m, \sigma) =' yes'$
- Alice says she never signed m. Is Alice lying?

1. Alice is lying: existential unforgeability means Alice signed m and therefore the Bank should give Bob 100$ from Alice’s account
2. Bob could have stolen Alice’s signing key and therefore the bank should not honor the statement
3. What a mess: the bank will need to refer the issue to the courts

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

soict.hust.edu.vn/ fb.com/groups/soict