~~Pta~~

Last time

- CRT ✓
- started ✓ public-key crypto

HW4 – Logistics
quiz 2

LectureLet 26

Nov 2, 2020

Plain RSA

Textbook

$Gen(1^n)$: ← sec param

$|Z_N^*|$

$N = pq$ → two primes of n-bits

$\phi(N) = (p-1)(q-1)$

choose $e > 1$, s.t. $\gcd(e, \phi(N)) = 1$ ← rel-prime

$1, \ldots, 1$ (n times)

$d := [e^{-1} \bmod (\phi(N))]$

$d \cdot e \equiv 1 \pmod{\phi(N)}$ ← Z_n^* (extended Euler)

public key : $pk = \langle N, e \rangle$

private key : $sk = \langle N, d \rangle$

secret

Enc: $m \in Z_N^*$ ← $\gcd(m, N) = 1$

$c := [m^e \bmod N]$ (uses $\langle N, e \rangle = pk$)

Dec: $m := [c^d \bmod N]$ (uses $\langle N, d \rangle = sk$)

Sanity check

$Dec_{sk}(Enc_{pk}(m)) = m$

$((m^e \bmod N))^d \bmod N = m$

simplify * → $m^{ed} \bmod N = m$ To prove

To prove

$$m^{ed} \bmod N = m$$

Proof:

$ed \equiv 1 \pmod{\Phi(N)}$ ← integer ← proof works

$ed = 1 + k \cdot \Phi(N)$

$m^{ed} = m^{1+k\cdot\Phi(N)}$

$= m \cdot m^{k\Phi(N)}$

$= m \cdot (m^{\Phi(N)})^k$

$m^{\Phi(N)} \equiv 1 \pmod N$ [see lecturelet #25]

$N \nmid m$

$m^{ed} \pmod N = m \pmod N$ ✓

Some attacks on RSA

- Attacker can factor N ($=pq$) (maybe p, q are small)
  - $\mathcal{A}$ - complete break
  - $\mathcal{A}$ can find $\Phi(N)$ ($=(p-1)(q-1)$) and thus d $\langle N, d\rangle$

NIST

Remedy: pick p, q so that factoring $N = pq$ is hard.

- If messages $m < B$ (50)
  - exhaustive attack

Remedy: msg space should not be too small (use padding)

More attacks in the book.

- why study these attacks? | informs how to choose params for RSA.

RSA problem

$\mathcal{A}$: adversay

Given: $\langle N, e \rangle$ (from RSA)

$c = (m^e \bmod N)$

Goal:

Find m

Turn into PPTA definition game

Is textbook RSA secure?

deterministic

Remedy: have to use randomization during encryption.

Padded RSA

- Gen ✓ (same as before) ← textbook RSA
- Enc: $\|N\|$ = size of N in num-of-bits $= pq$

$$\hat{m} = \underbrace{m}_{\|N\| - \ell(n) - 2} \| \underbrace{r}_{\ell(n)}$$

(m: msg; r: random #; $\ell(n)$: padding size; $r = 11\ldots1$ / 10)

$$c := [\hat{m}^e \bmod N]$$

- Dec:

$$\hat{m} := [c^d \bmod N]$$

↑ discard

m is first $N - \ell(n) - 2$ bits

Other ways to pad.

PKCS # v1.5 (standard on padding)

Cyclic groups

$Z_p^* = \{1, \ldots, p-1\}$ ← cyclic (prime) (Lecturelet 24)

$|Z_p^*| = p-1$. $2 \mid p-1$

$g \in Z_p^*$ $\quad \{g, g^2, g^3, \ldots, g^{p-1}\} = Z_p^*$

$g^{p-1} = 1$

But we need cyclic group of prime order? ↑ size

$p-1$ is not prime

Ex: $|Z_7^*| = 6$.

$p=7$ $\quad 2 \mid 6$

$|\mathbb{Z}_p^*| = (p-1) = qr$ $(6 = 3 \cdot 2)$

qr ↑ prime

consider $\{g^r, g^{2r}, \ldots, g^{qr}\} = \{g^r, (g^r)^2, \ldots, (g^r)^q\}$

$g^{(p-1)} = 1$ (FLT)

generator

size q

group, cyclic (gen g^r)

Ex: generator

$3 \to 3^2 \to 3^3 \to 3^4 \Rightarrow 3^5 \to 3^6$

$\mathbb{Z}_7^*$

$3^2 = 2$, $3^3 = 6$, $3^4 = 4$, $3^5 = 5$, $3^6 = 1$

$6 = 3 \cdot 2 = r$

g, $3^4 = 4$, $3^6 = 1$

$3^2 \to 3^4 \to 3^6$

group of order 3

$\mathbb{Z}_p^*$ $p - 1 = qr$

cyclic size q

$\mathbb{Z}_q^*$ ← prime, cyclic

Elliptic curves

Last time

- Textbook RSA
- Padded RSA
- How to generate cyclic group of prime order (size)

Logistics

Nov 15 2020

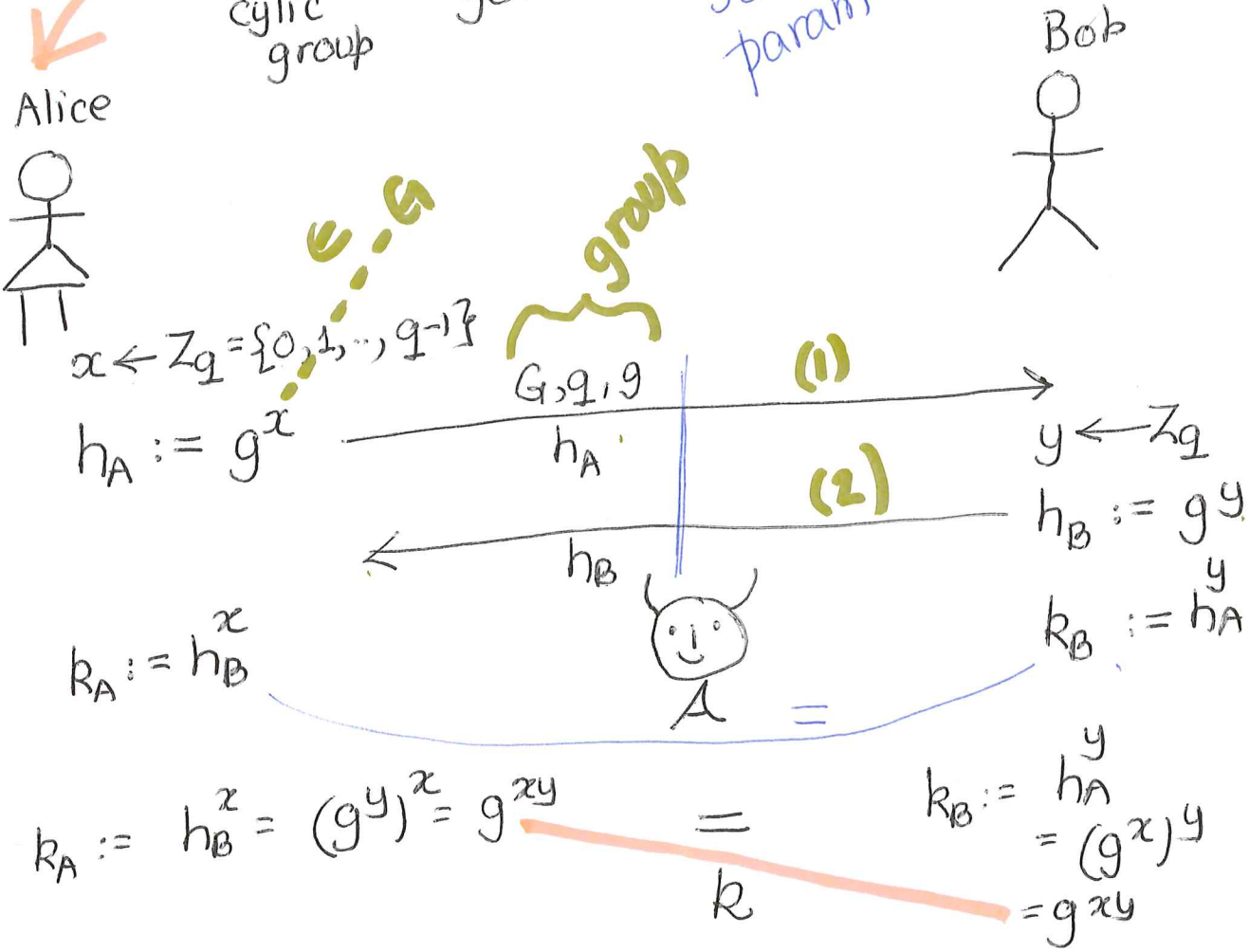
Lecture Let 27

Diffie Hellman key exchange

DH

$(G, q, g) \leftarrow Gen(1^n)$ — order prime (size), $\|q\| = n$ (# of bits in q), generators, cyclic group; sec. param

(covered in previous lecture_let)



$k_A := h_B^x = (g^y)^x = g^{xy} = k$

$k_B := h_A^y = (g^x)^y = g^{xy}$

A

knows:

G, q, g

$h_A = g^x \quad h_B = g^y$

Goal:

$k_A := k := k_B = g^{xy}$

Hard

known as the diffie-hellman problem (DHP)

Discrete log problem (DLP)

A knows:

G, q, g] same as before

g^x

Goal: x ; log

harder than

$DLP \geq_a DHP$

A_{DLP} algorithm to ~~fin~~ solve DLP

$A_{DLP}(G, q, g, g^x) = x$

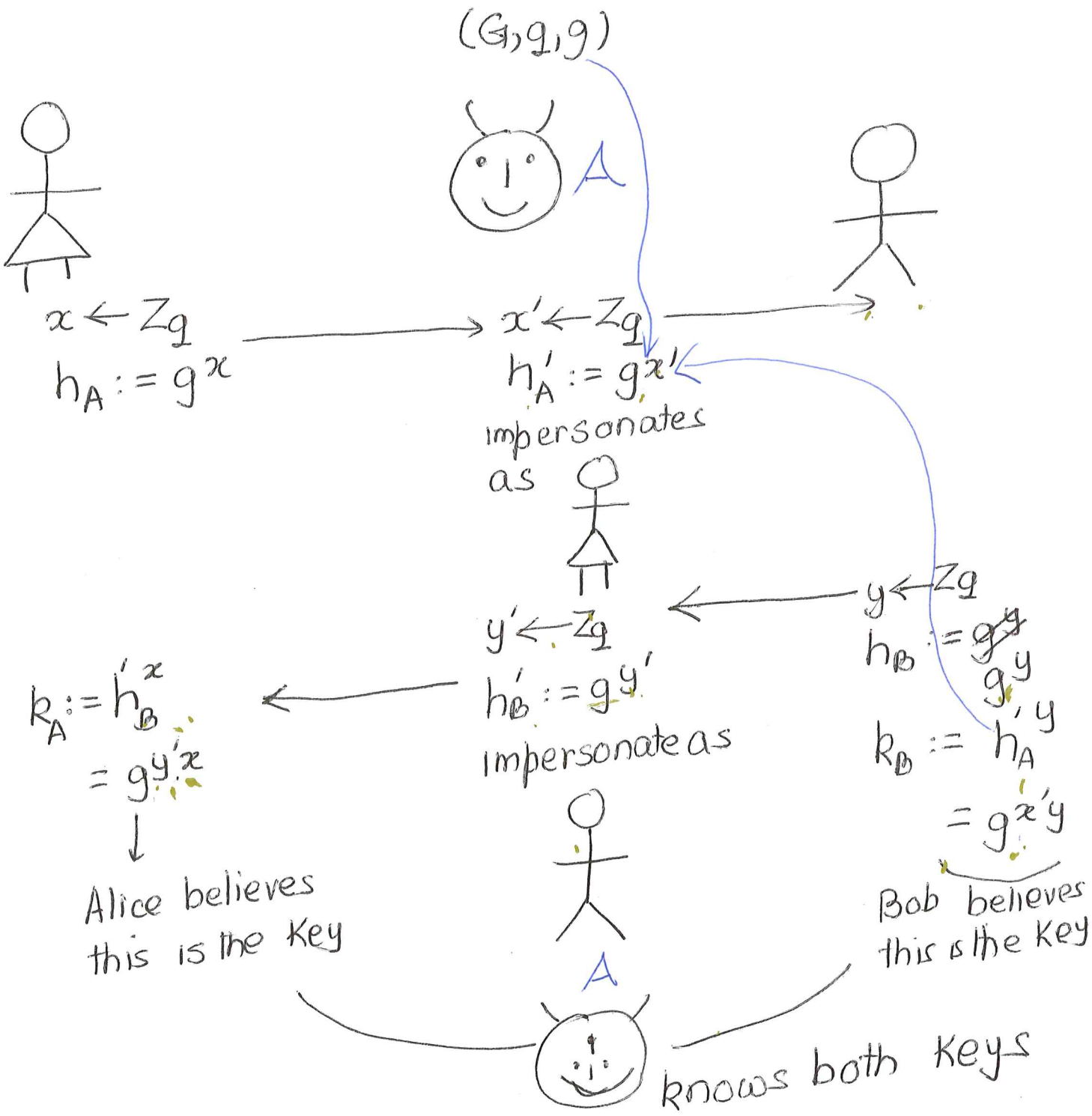
solve DHP

Given: G, q, g

g^x, g^y

$(g^y)^x = g^{xy}$ ✓ ✓

Use A_{DLP} to find x



Man-in-the-middle attack (MITM)

Digital sigs.

El Gamal encryption

- Gen

$(G, q, g) \leftarrow Gen(1^n)$

- order of the group/prime
- cyclic group
- generator
- elliptic curves
- same as DH Key exchange

$x \leftarrow Z_q$ → $\{0, \dots, q-1\}$

$h := g^x$

$pk = \langle G, q, g, h \rangle$

$sk = \langle G, q, g, x \rangle$

Keys

- Enc

$m \in G$ (msg in a group)

$y \leftarrow Z_q$ → $0 \dots q-1$

$c = \langle g^y, h^y \cdot m \rangle$

c_1 c_2

Uses pk $\langle G, q, g, h \rangle$

- Dec

$c = \langle c_1, c_2 \rangle$

$\langle G, q, g, x \rangle$

Uses sk

$\hat{m} = \frac{c_2}{c_1^x} = c_2 \cdot (c_1^x)^{-1}$

inverse of c_1^x in G

Sanity check

$\hat{m} = \frac{c_2}{c_1^x} = \frac{h^y \cdot m}{(g^y)^x} = \frac{(g^x)^y \cdot m}{g^{xy}} = m$ ✓

g^x

$\mathcal{A}$: Knows: $pk = \langle G, q, g, h \rangle$ $= g^x$ $(x \leftarrow Z_q)$

$c = \langle g^y, h^y \cdot m \rangle$ $y \leftarrow Z_q$ randomized randomly each time

Goal:

Find m

- $h^y = g^{xy}$

$\mathcal{A}$ Know: $h = g^x$, $c_1 = g^y$

DHP ($\mathcal{A}$ knows how to solve)

$g^{xy} = h^y = (g^x)^y$

$(h^y)^{-1} \cdot c_2 = m$ ✓ $c_2 = h^y \cdot m$

- $m \leftarrow \mathcal{A}$ knows.

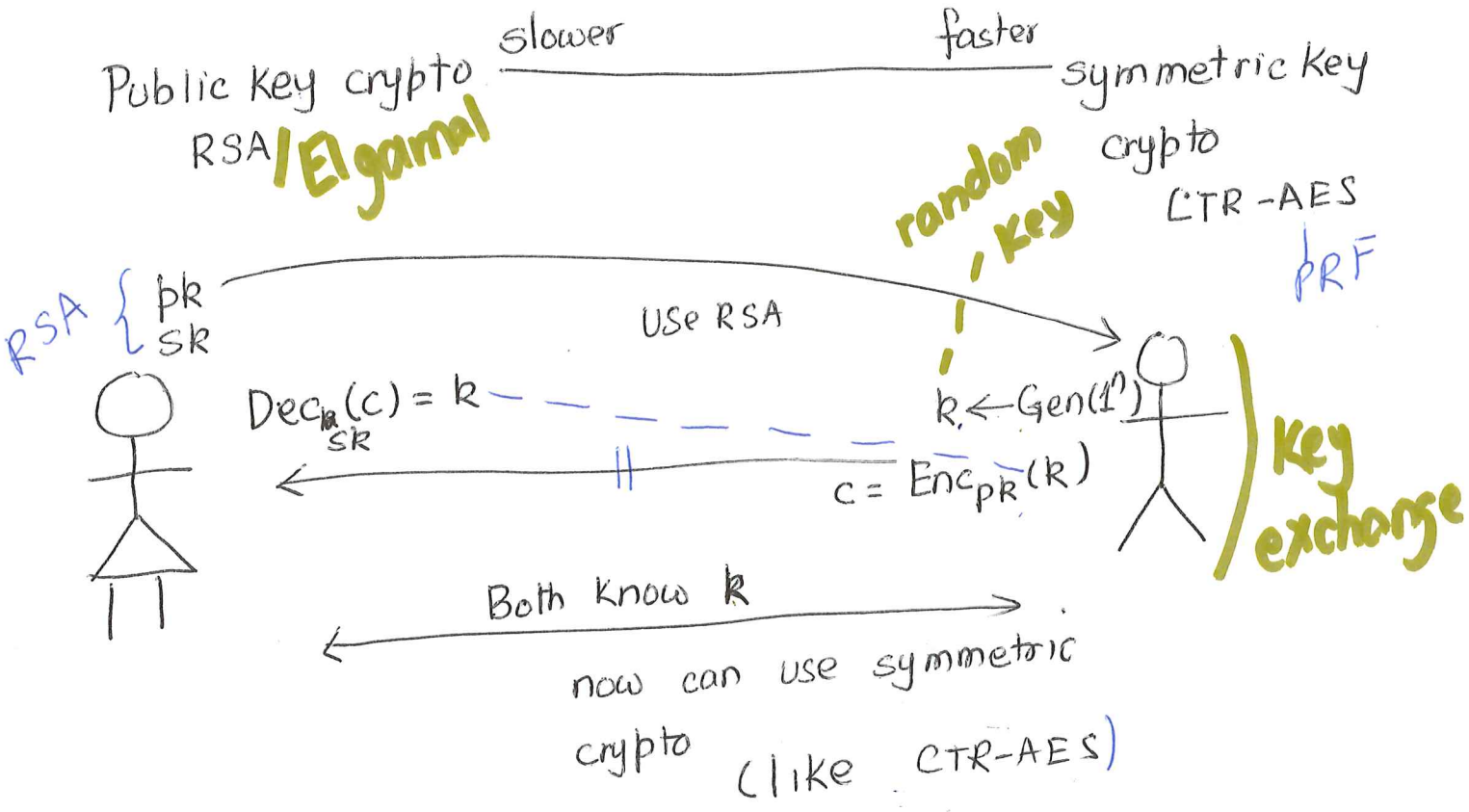
$(h^y \cdot m) \cdot m^{-1} = h^y = g^{xy}$ $= DHP$

So security of $\mathcal{D}$ Elgamal is equivalent to DHP

-Book makes this formal.

≠ deterministic

Hybrid scheme



Last time

- Diffie-Hellman key exchange
- El gamal encryption

Logistics

LectureLet 28/28

Nov 21 2020

L11

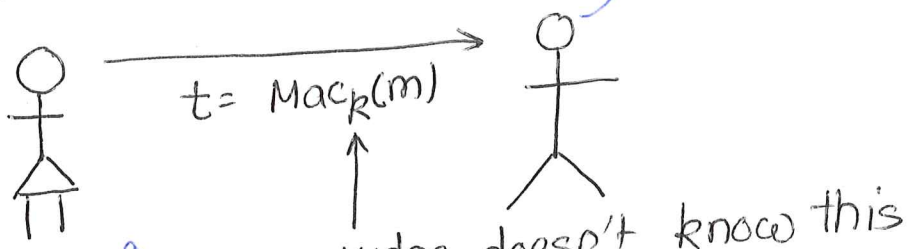
Digital signatures

Equivalent to physical signature

John Doe

- How are digital signatures different from msg authentication? (MAC)

- signatures are publically verifiable (e.g by a judge during a dispute)
  - non repudiation
- MACs
  - don't have the property



Definition

signing → (Gen, Sign, Vrfy) ← verifying

RSA, Elgamal

- (pk, sk) ← Gen(1^n)
  - pk: public key
  - sk: private key
  - sec. param
  - n=1000
- <similar to public key cryptography>
  - length of pk, sk atleast n-bits
  - length of pk, sk depend on n

2. Sign

$\sigma \leftarrow Sign_{sk}(m)$ ← $\{0,1\}^*$ (arbitrary size)

<Uses secret key>

3. Vrfy

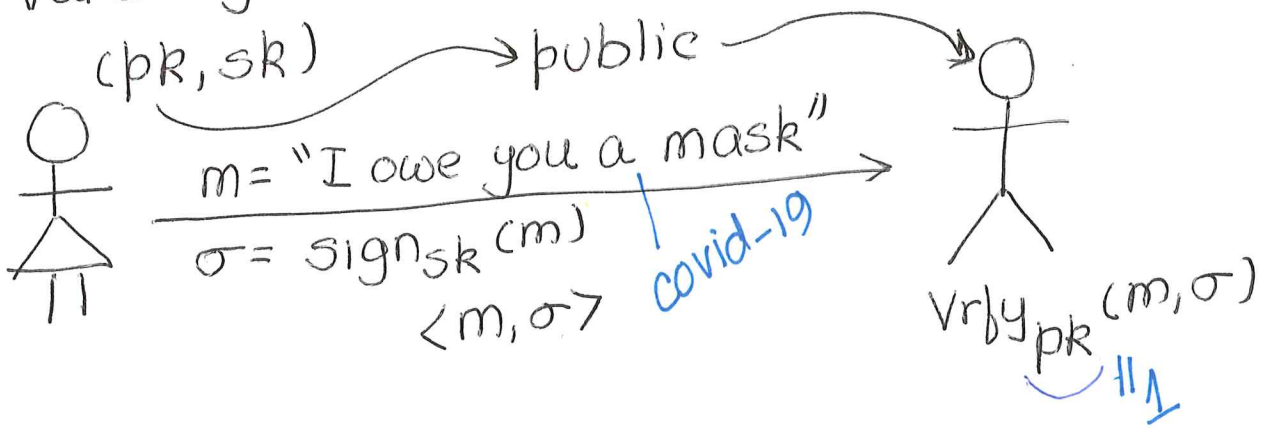
message, signature on m

bit ↘ $b := Vrfy_{pk}(m, \sigma)$

$=1$ → valid signature

$=0$ → invalid

<Uses public key>



Sanity check

$$Vrfy_{pk}(m, Sign_{sk}(m)) = 1$$

Sometimes signature schemes only work for messages of fixed lengths ($\{0,1\}^{\ell(n)}$ → {e.g. n^2})

all msgs of length $\ell(n)$

MAC "similarities"

Security

$\Pi = (Gen, Sign, Vrfy)$

$Sig\text{-}forge_{A,\Pi}(n)$

A

$(pk, sk) \leftarrow Gen(1^n)$

pk

oracle access to $Sign_{sk}(\cdot)$

secret key

$m_1, \ldots, m_k$

$\sigma_1 \ldots \sigma_k$

Q: set of all queries A asks oracle $Sign_{sk}(\cdot)$

forgery (m, σ)

valid forgery

$b = 1$ (1) $Vrfy_{pk}(m, \sigma) = 1$

(2) $m \notin Q$

$b = 0$ otherwise

for all PPT A

$$P[Sig\text{-}forge_{A,\Pi}(n) = 1] \leq negl(n)$$

Similar to Mac-forge

@Go back and review it

Hash-and-sign paradigm

Similar to hash-and-mac

- Signature scheme $\Pi = (Gen, Sign, Vrfy)$

$\ell(n) = n$ ↓ can only handle msgs of size $\ell(n)$ $[\{0,1\}^{\ell(n)}]$ (128-bits)

- Hash function

$H^s : \{0,1\}^* \longrightarrow \{0,1\}^{\ell(n)}$

$\{0,1\}^*$: arbitrary size

$\{0,1\}^{\ell(n)}$: len $\ell(n)$, 128-bits

want to sign arbitrary sized msgs

$$\sigma \leftarrow Sign_{sk}(H^s(m))$$

$$Vrfy_{pk}(H^s(m), \sigma)$$

Basically Hash the msg before signing and verification

Plain RSA

$(pk, sk) \leftarrow Gen(1^n)$; $pk = \langle N, e \rangle$; $sk = \langle N, d \rangle$; $= pq$

$ed \equiv 1 \pmod{\phi}$; $\phi = (p-1)(q-1)$

Enc, Sign: pk, $\langle N, e \rangle$

Dec, Vrfy: sk, $\langle N, d \rangle$

$\langle$same as RSA$\rangle$

$\gcd(m, N) = 1$; $m \in Z_N^* = \gcd(m, N) = 1$

$\sigma := [m^d \bmod N]$

$m \stackrel{?}{=} [\sigma^e \bmod N]$

uses secret key

uses public key

Sanity check

$$\sigma^e = (m^d)^e = m \pmod N$$

same proof for as sanity check for RSA - encryption

No-message attack

$\sigma \in Z_N^*$

$m := [\sigma^e \bmod N]$ ← public, ← garbage

(m, σ) valid forgery
- oracle not used $Q = \emptyset$
  $m \notin Q$
- $Vrfy_{pk}(m, \sigma)$
  $m \stackrel{?}{=} [\sigma^e \bmod N]$ ✓ by definition.

Multiplicative attack

A: $m \in Z_N^*$

sig-forge.

choose m_1, m_2 s.t $m = m_1 \cdot m_2 \pmod N$?

ask oracle for $sign_{sk}(m_1)$ $sign_{sk}(m_2)$

σ_1 σ_2

$\sigma = [\sigma_1 \cdot \sigma_2 \bmod N]$

(m, σ) valid forgery

✓ - m was never a query to the oracle.

mod(N) → ✓ - $\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1^{ed} \cdot m_2^{ed}$

$= m_1 \cdot m_2$

$= m \bmod N$

$Vrfy_{pk}(m, \sigma) = 1.$

Hash-then-~~MAC~~ paradigm
RSA-FDH sign

- Gen ✓ same as before — RSA

$H^s: \{0,1\}^* \to Z_N^*$

- Sign

$\sigma := [\underbrace{H^s(m)}_{Z_N^*}{}^d \bmod N]$ $\quad \{0,1\}^*$

Uses $sk = \langle N, d \rangle$

- Vrfy

$Vrfy_{pk}(m,\sigma) = \begin{cases} \sigma^e \stackrel{?}{=} H^s(m) \bmod N \end{cases}$

Uses $pk = \langle N, e \rangle$

No-msg attack

$\hat{m} := [\sigma^e \bmod N]$

σ

need to find m s.t $H(m) = \hat{m}$ ← Hard

↑ pre-image resistant

Multiplicative attack

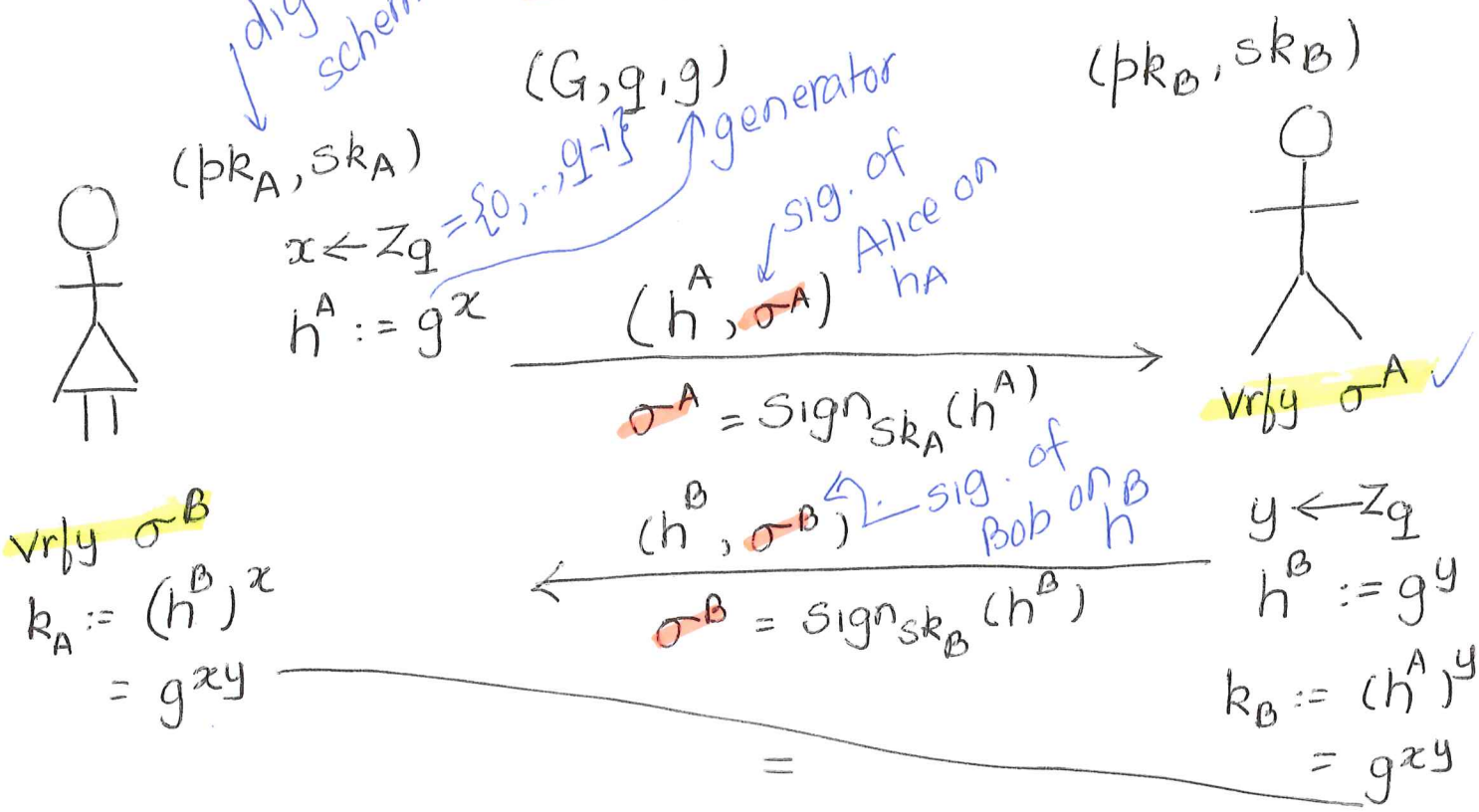
$\underbrace{H(m) = H(m_1) \cdot H(m_2) \bmod N}$

hard for hash functions

$H(m) = H(m')$

- Book formal proof

Authenticate Diffie-Hellman key exchange



MITM does not work

- A does not know sk_A and sk_B
- cannot generate signatures σ^A, σ^B
- cannot impersonate  and 

Certificate chains