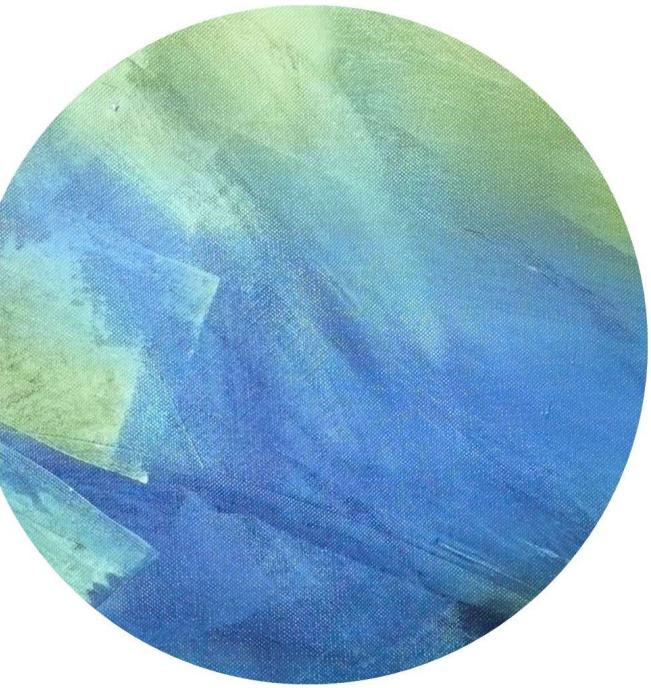
Lecture ~~10~~ 11



Outline

Bleichenbacher attack

CCA Secure Encryption

RSA OAEP

Public Key Encryption

Textbook RSA System

Gen(.): choose random primes $p,q \approx 1024$ bits. Set **N=pq**.

choose integers **e , d** s.t. $\mathbf{e \cdot d = 1 \pmod{\varphi(N)}}$

output pk = (N, e) , sk = (N, d)

RSA-Enc (pk, x) = $\mathbf{x^e}$ (in Z_N)

RSA-Dec (pk, y) = $\mathbf{y^d}$ (in Z_N)

$$y^d = x^{ed} = x^{k\varphi(N)+1} = \left(x^{\varphi(N)}\right)^k \cdot \mathbf{x} = (1)^k \cdot \mathbf{x} = x$$

Trapdoor Permutation:

Trapdoor Permutation

Three algorithms: (G, F, F^{-1})

- G: outputs pk, sk. pk defines a function $F(pk, \cdot): X \to X$
- $F(pk, x)$: evaluates the function at x
- $F^{-1}(sk, y)$: inverts the function at y using sk Gives x s.t. $F(pk, x) = y$.

Secure trapdoor permutation:

The function $F(pk, \cdot)$ is one-way without the trapdoor sk

$$\Pr\left[\mathcal{A}(pk, y) \to x\right] = \text{negl}$$

$y = F(pk, x)$

$(pk, sk) \leftarrow G$

$x \leftarrow X$

The RSA assumption

RSA assumption: RSA is one-way permutation

For all efficient algs. A:

$$\Pr\left[\ A(N,e,y) = \underbrace{y^{1/e}}_{y^d}\ \right] < \text{negligible}$$

where $p,q \xleftarrow{R}$ n-bit primes, $N \leftarrow pq$, $y \xleftarrow{R} Z_N^*$

Textbook RSA System

Gen(.): **pk** = (N, e), **sk** = (N, d) such that **e·d = 1 (mod φ(N))**

RSA-Enc (pk, x) = x^e (in Z_N)

RSA-Dec (pk, y) = y^d (in Z_N)

Is not semantically secure

How would you make it semantically secure?

$$\mathrm{Enc}(pk, x; r) = (\underline{r^e}, \underbrace{H(r)}_{\text{extractor}} \oplus m)$$

Speeding up RSA

x^e x^{21} ...

$\{ x \quad x^2 \quad x^4 \quad x^8 \quad x^{16} \ldots \}$

x^{16}

To speed up RSA use a small e: $c = m^e \pmod N$

- Recommended value: **e >= 65537 = $\boxed{2^{16}+1}$** ($\gcd(e, \varphi(N)) = 1$)

Encryption: 17 multiplications

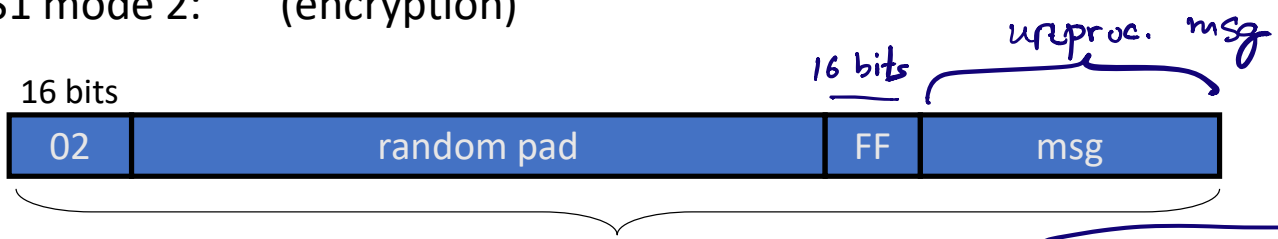
<u>Asymmetry of RSA:</u> fast enc. / slow dec.

- ElGamal (~~next module~~): approx. same time for both.

$(gh)^a$

RSA in practice: PKCS1 v1.5

PKCS1 mode 2: (encryption)



RSA modulus size (e.g. 2048 bits)

processed msg

- Resulting value is RSA encrypted $(\text{processed msg})^e$.
- Widely deployed, e.g. in HTTPS
- Suffered from a CCA attack

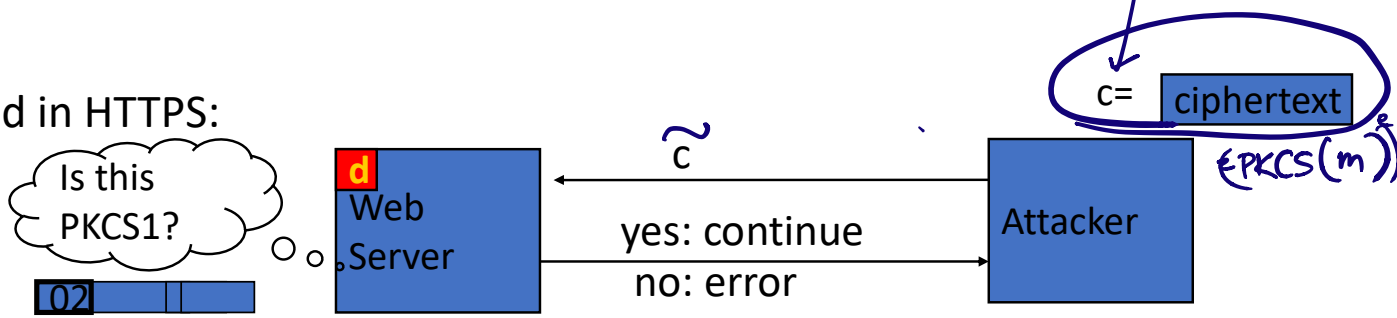
$e \cdot d = 1 \mod \varphi(N)$

$(x^e)^d = x \mod N.$

CCA Attack on PKCS1 v1.5

(Bleichenbacher 1998)

PKCS1 used in HTTPS:



⇒ attacker can test if 16 MSBs of plaintext = '02'

Chosen-ciphertext attack: to decrypt a given ciphertext **c** do:

- Choose $r \in Z_N$. Compute $c' \leftarrow r^e \cdot c = (r \cdot \mathrm{PKCS1}(m))^e$
- Send c' to web server and use response

$= (r^e \cdot (\text{proc. msg})^e)$

$= (r \cdot (\text{proc. msg}))^e$

02	random pad	FF	msg

$c = x^e \bmod N$

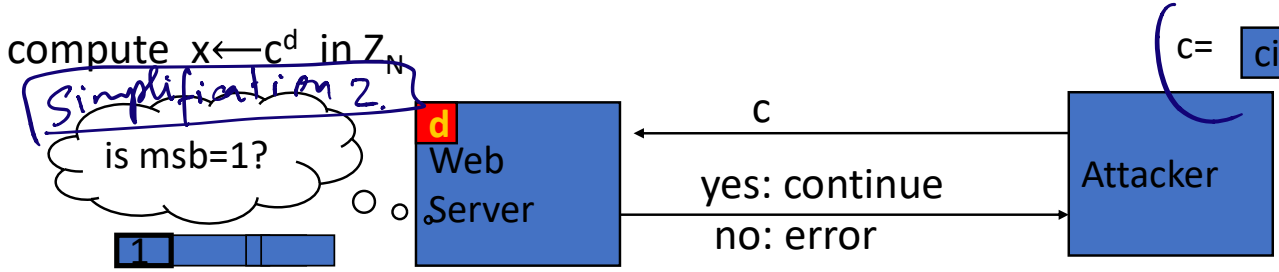
$\tilde{c} = ((2^{-1})^e \cdot c) \bmod N$

Baby Bleichenbacher

$\boxed{02}$

map: If first bit of x is 1, then $x' = \boxed{02}\cdots$

first bit of x is 0, $x' \neq \boxed{02}\cdots$



(Simplification 1)

Suppose N is $\boxed{N = 2^n}$ (an invalid RSA modulus). Then:

- Sending c reveals msb(x)
- Sending $2^e \cdot c = (2x)^e$ in Z_N reveals msb(2x mod N) = $\mathbf{msb_2(x)}$
- Sending $4^e \cdot c = (4x)^e$ in Z_N reveals msb(4x mod N) = $\mathbf{msb_3(x)}$
- … and so on to reveal all of x

Chosen Ciphertext Security for Public Key Encryption

Challenger $\xrightarrow{pk}$ Attacker

$\xleftarrow{\tilde{c}}$

$\xrightarrow{Dec_{sk}(\tilde{c})}$

$\xleftarrow{\tilde{\tilde{c}}}$

$\xrightarrow{Dec_{sk}(\tilde{\tilde{c}})}$

$\xleftarrow{m_0, m_1}$

$\xrightarrow{c = Enc_{pk}(m_b)}$

$\xleftarrow{c'}$

$\xrightarrow{Dec_{sk}(c')}$

$\vdots$

set of Adv. queries

$c \notin \{\tilde{c}, \tilde{\tilde{c}}, c' \ldots\}$

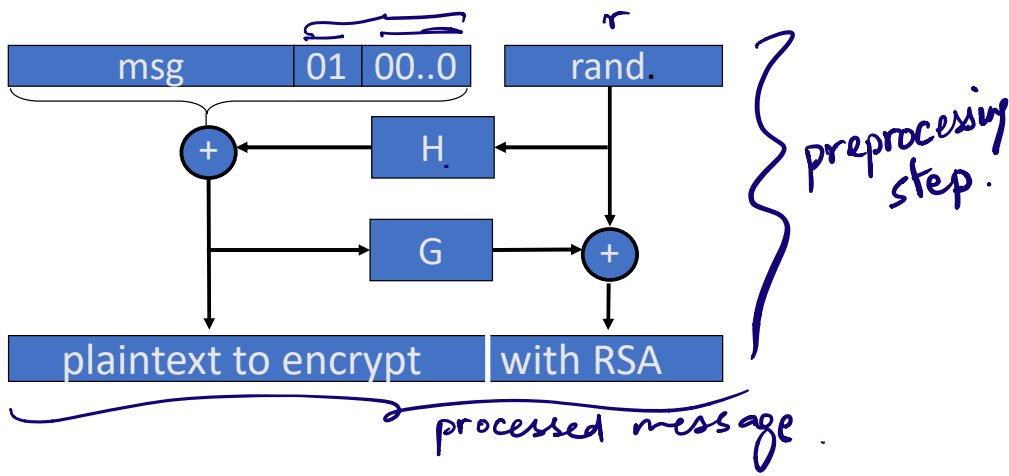
$|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = negl$

b'?

PKCS1 v2.0: OAEP

New preprocessing function: OAEP [BR94]

check pad
on decryption.
reject CT if invalid.



Thm [FOPS'01] : RSA is a trap-door permutation $\Rightarrow$
RSA-OAEP is CCA secure when H,G are *random oracles*

in practice: use SHA-256 for H and G

What is a random oracle?

H: Input	Output
00 0	uniform random string s_1
00 1	uniform random s_2
⋮	
111 . . . 1	uniform random s_3

- H: Truly random function
- "Observable"

No attacker can output $z = H(y)$ without first querying oracle on y.

STRONGER: For any attacker $\forall y$, $H(y)$ is unif. random unless attacker queries R.O. on y.

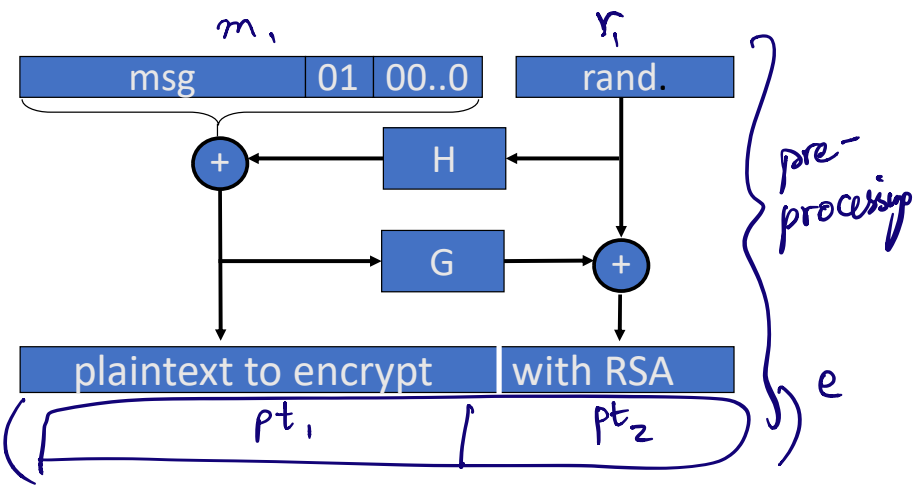
- "Programmable"

Stronger than observability.

"Proof of security / Challenger" can set the output of RO on any given input x.

Fujisaki - Okamoto
Pointcheval - Stern

Analyze OAEP



$pt_1 = m_1 \oplus H(r_1)$

$pt_2 = G(pt_1) \oplus r_1$

$Dec(\tilde{c}) = pt_1 \mid pt_2$

$r_1 = G(pt_1) \oplus pt_2$

$m_1 = H(r_1) \oplus pt_1$

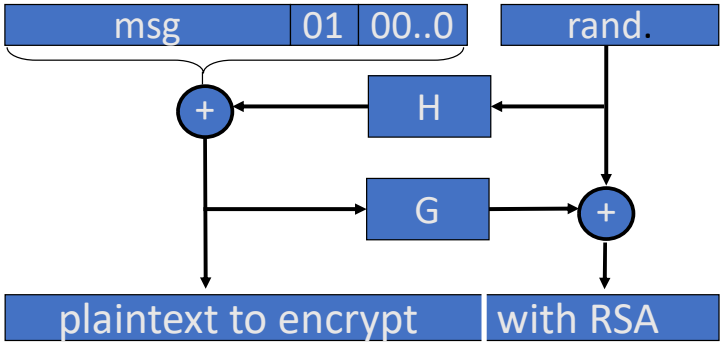
"Plaintext - awareness".

Goal: Argue plaintext-awareness.

Claim: If Attacker "knows" (r_1, pt_1), they also "know" m_1.

Challenger MUST HAVE recorded queries (pt_1, r_1) s.t. $\tilde{ct}$ can be recomputed given (pt_1, r_1)

Analyze OAEP



"Plaintext-awareness" $\Rightarrow$

Any attacker that queries CCA Dec. oracle on $\boxed{\tilde{ct}}$, must "know" underlying (m_1, r_1).

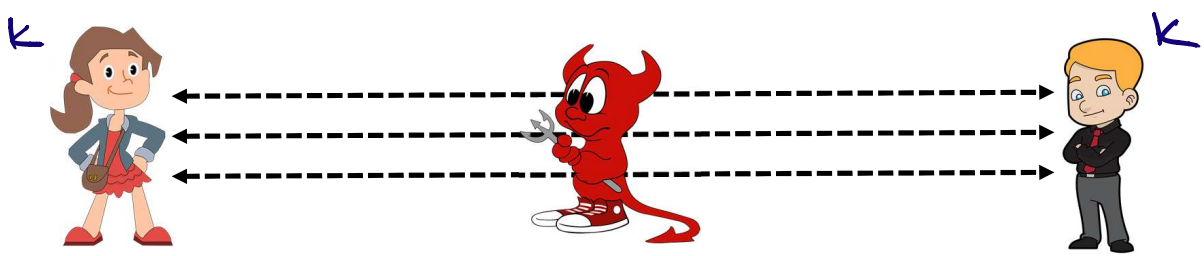
$\Rightarrow$ For any attacker that makes CCA Dec. oracle query on $\boxed{\tilde{ct}}$,

Challenger sitting inside RO, must have "recorded" a query that would allow challenger to compute (m_1, r_1) w/o using sk at all.

Authenticated Key Exchange

Authenticated Key Exchange

- Recall key exchange



- Adversary eavesdrops on the network

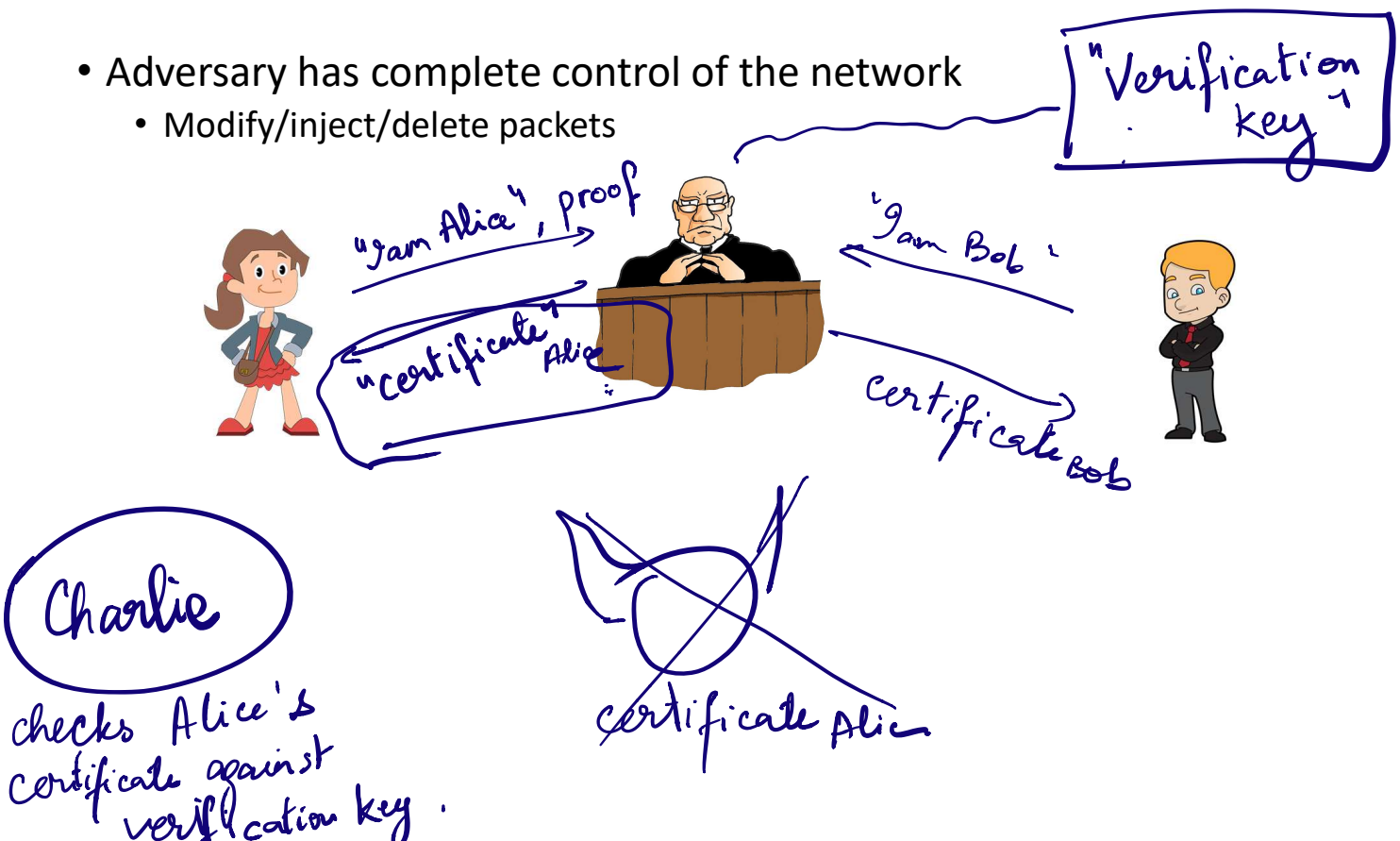
Authenticated Key Exchange

- Adversary has complete control of the network
  - Modify/inject/delete packets

Man-in-the-middle attack.

Trusted Third Parties

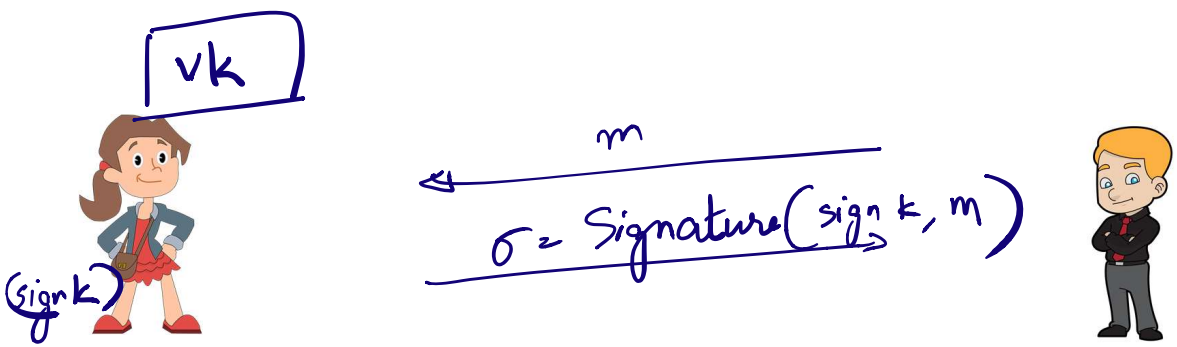
- Adversary has complete control of the network
  - Modify/inject/delete packets



Authenticated Key Exchange

(Next lecture)

Important Component: Signature Scheme



Digital Signatures

= "Public-key" MAC

= Verification is "public" instead of private.

Challenger (vk, sk) — Attacker

$\xrightarrow{vk}$

$\xleftarrow{m_1}$

$\xrightarrow{\sigma = \mathrm{Sign}(sk, m_1)}$

$\vdots$

$\xleftarrow{m_i}$

$\xrightarrow{\sigma = \mathrm{Sign}(sk, m_i)}$

$\vdots$

$\xrightarrow{(\tilde{m}, \tilde{\sigma})}$

END OF LECTURE

Wins if $\mathrm{Verify}(\tilde{m}, \tilde{\sigma}, vk) = 1$

Digital Signatures

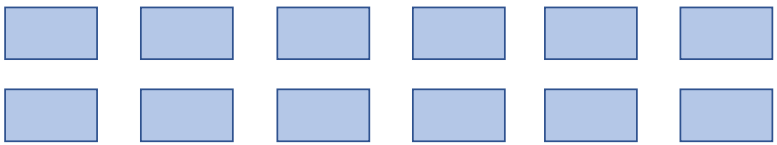
- Public-key version of “MACs”

One-Time Signatures

- Definition of security

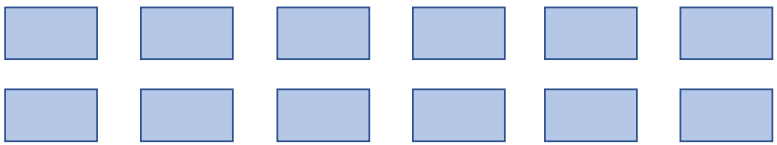
Lamport One-Time Signatures

- vk =



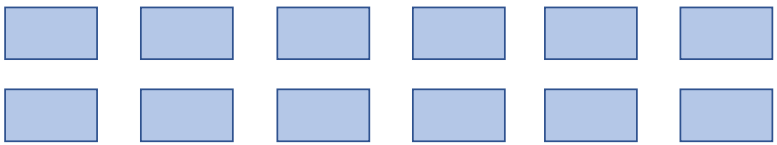
- sk =

Lamport One-Time Signatures



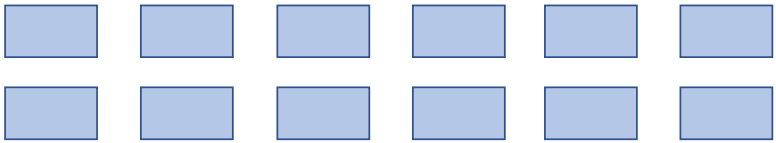
- One-time security:

Lamport One-Time Signatures



- One-time security:

Lamport One-Time Signatures



- What about two-time security?

Summary

- RSA review
- Bleichenbacher attack
- PKCS 1 v2.0
- Authenticated Key Exchange and Signatures