CSE 484 / CSE M 584: Public Key Encryption + Digital Signatures

Winter 2025

Nirvan Tyagi
tyagi@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner, and Nirvan Tyagi. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, Nickolai Zeldovich and many others.

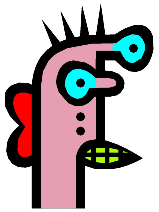
Announcements

- Things due
  - Homework 2: Next Wednesday

Applications of Public Key Cryptography

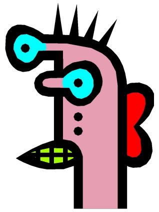
- Encryption for confidentiality
- Digital signatures for integrity
- Session key establishment / “Key exchange”

Public Key Encryption



Alice

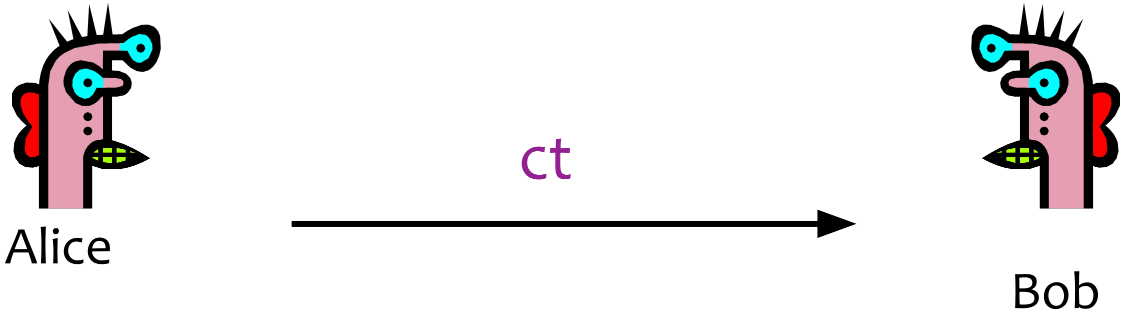
pk_B



Bob

pk_B, sk_B

Public Key Encryption

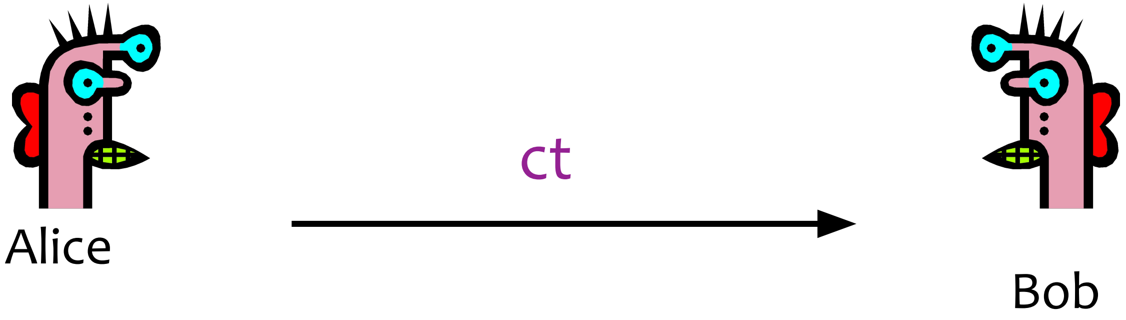


pk_B

pk_B, sk_B

$\text{Encrypt}(pk_B\ , m) \rightarrow ct$

Public Key Encryption



pk_B

pk_B, sk_B

$\text{Encrypt}(pk_B\,, m) \longrightarrow ct$

$\text{Decrypt}(sk_B\,, ct) \longrightarrow m$

Public Key Encryption from Diffie-Hellman

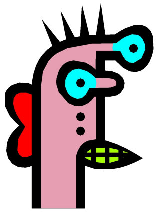
pk_B pk_B, sk_B

Alice Bob

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

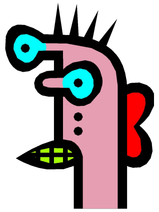
Public Key Encryption from Diffie-Hellman

pk_B



Alice

pk_B, sk_B



Bob

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

<u>Sample one-time key:</u>

$sk_A \leftarrow r$
$pk_A \leftarrow g^r$

<u>Compute DH shared secret:</u>

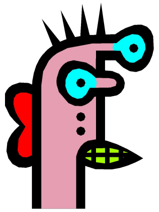
$K = H(g^{ry})$

<u>Encrypt with authenticated symmetric encryption:</u>

$ct_{SE} = SE.Enc(K, m)$

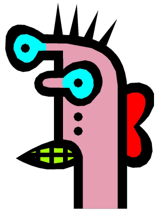
Public Key Encryption from Diffie-Hellman

pk_B



Alice

pk_B, sk_B



Bob

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

<u>Sample one-time key:</u>

$sk_A \leftarrow r$
$pk_A \leftarrow g^r$

$$ct = (g^r, ct_{SE}) \longrightarrow$$

<u>Compute DH shared secret:</u>

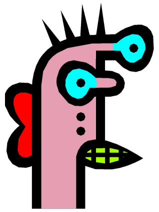
$K = H(g^{ry})$

<u>Encrypt with authenticated symmetric encryption:</u>

$ct_{SE} = SE.Enc(K, m)$

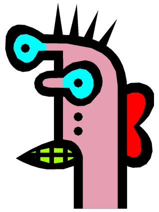
Public Key Encryption from Diffie-Hellman

pk_B



Alice

pk_B, sk_B



Bob

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$

<u>Sample one-time key:</u>

$sk_A \leftarrow r$
$pk_A \leftarrow g^r$

$ct = (g^r, ct_{SE})$ →

<u>Compute DH shared secret:</u>

$K = H(g^{ry})$

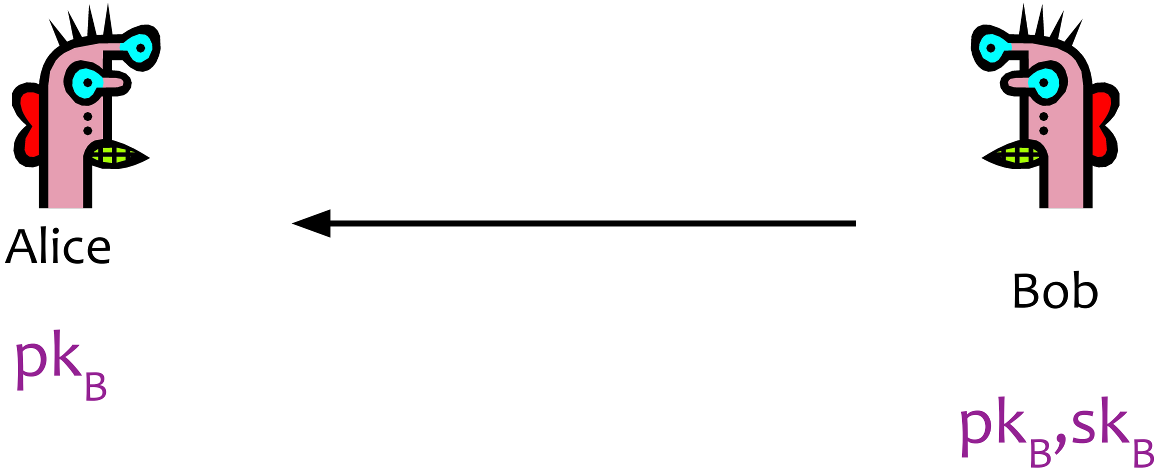
<u>Encrypt with authenticated symmetric encryption:</u>

$ct_{SE} = SE.Enc(K, m)$

$K = H(g^{ry})$
$m = SE.Dec(K, ct_{SE})$

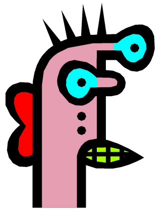
Digital Signatures

- No one should be able to forge signatures from Bob’s public key without Bob’s secret key



Digital Signatures

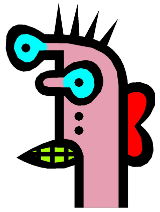
- No one should be able to forge signatures from Bob's public key without Bob's secret key



Alice

pk_B

$\xleftarrow{\sigma}$



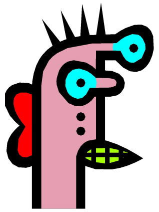
Bob

pk_B, sk_B

$\mathrm{Sign}(sk_B, m) \rightarrow \sigma$

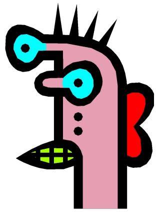
Digital Signatures

- No one should be able to forge signatures from Bob's public key without Bob's secret key



Alice

σ



Bob

pk_B

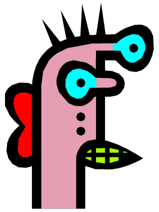
pk_B, sk_B

$\text{Verify}(pk_B, m, \sigma) \rightarrow 0/1$

$\text{Sign}(sk_B, m) \rightarrow \sigma$

Digital Signatures

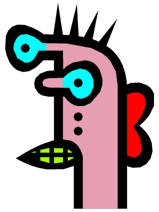
- No one should be able to forge signatures from Bob’s public key without Bob’s secret key



Alice

pk_B

$\xleftarrow{\sigma}$



Bob

pk_B, sk_B

$\mathrm{Verify}(pk_B, m, \sigma) \rightarrow 0/1$

$\mathrm{Sign}(sk_B, m) \rightarrow \sigma$

In-Class Activity 2/5: What benefit do signatures have over MACs?

Assume prime-order group

Schnorr Signature

<u>Sample one-time key:</u>

r, g^r

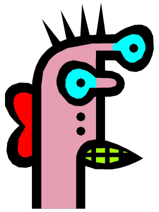
<u>Compute random challenge:</u>

$c = H(g^y, g^r, m)$

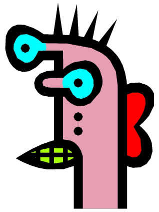
<u>Prove "knowledge" of y:</u>

$z = r + yc$

$sk_B \leftarrow y$
$pk_B \leftarrow g^y$



Bob



Alice

Assume prime-order group

Schnorr Signature

Sample one-time key:

r, g^r

Compute random challenge:

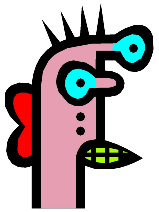
$c = H(g^y , g^r , m)$

Prove "knowledge" of y:

$z = r + yc$

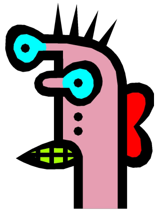
$sk_B \leftarrow y$

$pk_B \leftarrow g^y$



Bob

$\sigma = (g^r , z)$



Alice

Assume prime-order group

Schnorr Signature

<u>Sample one-time key:</u>

r, g^r

<u>Compute random challenge:</u>

$c = H(g^y, g^r, m)$

<u>Prove "knowledge" of y:</u>

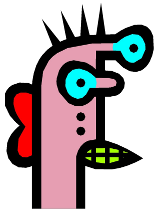
$z = r + yc$

$c = H(g^y, g^r, m)$

$g^z \stackrel{?}{=} g^r \oplus (g^y)^c$

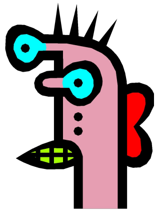
$sk_B \leftarrow y$

$pk_B \leftarrow g^y$



Bob

$\sigma = (g^r, z)$



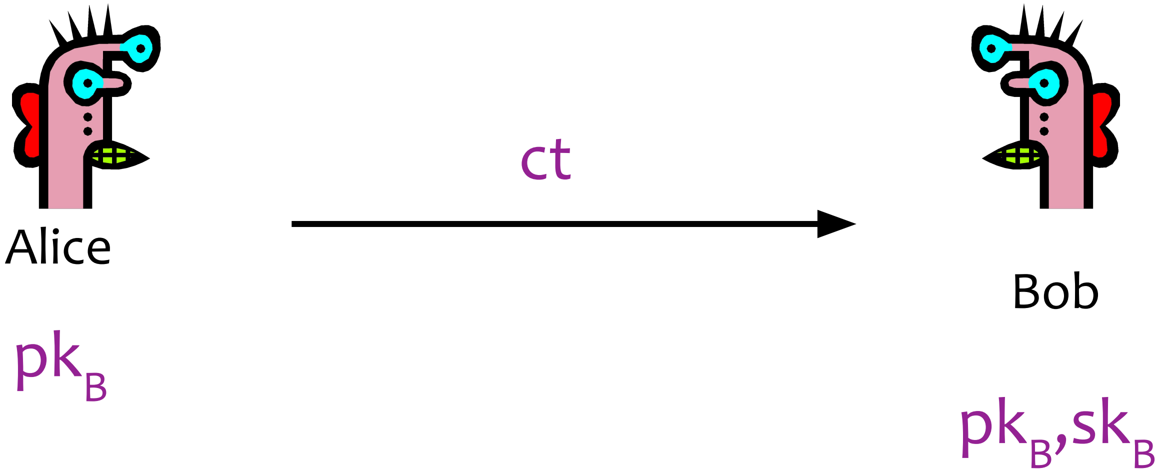
Alice

RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- Operates over Z_n^* for n = pq (product of 2 primes)

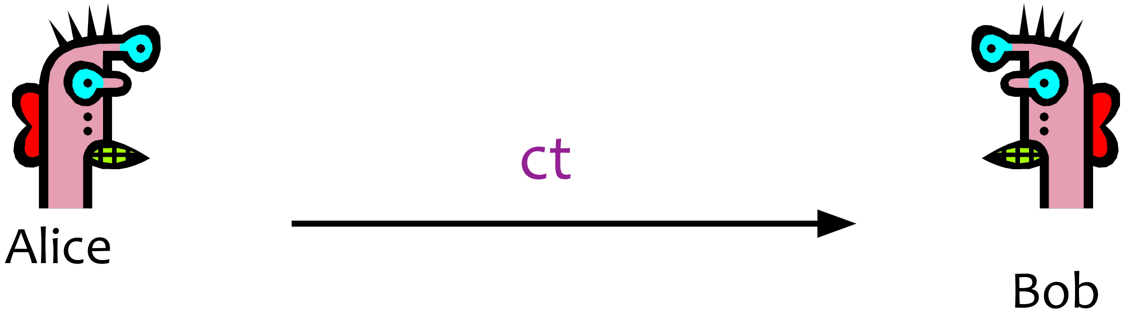
- Background: Helpful number theory facts about Z_n^*
  - Order = $\varphi(n) = (p-1)(q-1)$
    - $\varphi(n)$: Euler's Totient Function: # of integers in $[1,n)$ relatively prime to n
  - Euler's Theorem:
    - For every $a \in Z_n^*$, $a^{\varphi(n)} = 1 \pmod n$

RSA Cryptosystem [Rivest, Shamir, Adleman 1977]



- Key generation:
  - Generate large primes p, q
  - Compute **n**=pq and $\varphi(\mathbf{n})=(p-1)(q-1)$
  - Choose small **e**, relatively prime to $\varphi(n)$
  - Compute modular inverse **d**: $ed \equiv 1 \bmod \varphi(n)$
  - $pk_B = (e,n)$; $sk_B = (d,n)$

RSA Cryptosystem [Rivest, Shamir, Adleman 1977]



pk_B

pk_B, sk_B

- Encryption: $ct = m^e \bmod n$

- Decryption:

$ct^d \bmod n = (m^e)^d \bmod n = m$

- Key generation:
  - Generate large primes p, q
  - Compute **n**=pq and φ(**n**)=(p-1)(q-1)
  - Choose small **e**, relatively prime to $\varphi(n)$
  - Compute modular inverse **d**: $ed \equiv 1 \bmod \varphi(n)$
  - $pk_B = (e,n)$; $sk_B = (d,n)$

Why is RSA Secure?

- RSA problem: given c, $n=pq$, and e such that $\gcd(e, \varphi(n))=1$, find m such that $m^e=c \bmod n$
- Factoring problem: given positive integer n, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e1}p_2^{e2}\ldots p_k^{ek}$

- If factoring is easy, then RSA problem is easy
  (knowing factors means you can compute d = inverse of e mod $(p-1)(q-1)$)

Why is RSA Secure?

- RSA problem: given c, $n=pq$, and e such that $\gcd(e, \phi(n))=1$, find m such that $m^e=c \bmod n$
- Factoring problem: given positive integer n, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e1}p_2^{e2}\ldots p_k^{ek}$

- If factoring is easy, then RSA problem is easy
  (knowing factors means you can compute d = inverse of e mod $(p-1)(q-1)$)

- Other RSA Caveats
  - If m is small, can brute force
  - Not randomized!
  - Requires n ~ 2048-4096 bits for 128-bits of security
  - Largely being phased out for efficient elliptic curve group cryptography

What do Quantum Computers mean for Cryptography?

What do Quantum Computers mean for Cryptography?

1. Implications for existing cryptography
   - Quantum algorithms exist to solve “hard” assumptions quickly
     - Shor’s algorithm can solve factoring and discrete logarithm
   - “Post-quantum” cryptography
     - Build asymmetric cryptography for classical computers based on assumptions that we think are “hard” even for quantum computers
     - “Lattice-based” cryptography

What do Quantum Computers mean for Cryptography?

1. Implications for existing cryptography
   - Quantum algorithms exist to solve “hard” assumptions quickly
     - Shor’s algorithm can solve factoring and discrete logarithm
   - “Post-quantum” cryptography
     - Build asymmetric cryptography for classical computers based on assumptions that we think are “hard” even for quantum computers
     - “Lattice-based” cryptography
2. Implications for future cryptography
   - Quantum computing offers new hardness assumptions and new functionality from which to build cryptography