CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2023 Lecture 10

Announcements

1. HW 4 is due today!
2. HW 5 is due in 1.5 weeks!

Last time:

- Diffie-Hellman and elementary discrete log algorithms.

This time:

- More number theory: rings, fields, CRT, Euler ϕ
- A bit more about discrete logs
- Public-key cryptography: RSA, factoring assumptions, ElGamal

Math review: Algebraic rings

- A ring is a set closed under two operations: $+$, $\times$
- $+$ has identity, inverses, associative, commutative
- $\times$ has identity, associative, commutative, might not have inverses
- Distributive law for $+$, $\times$

Canonical examples to remember:

- $\mathbb{Z}$
- $\mathbb{Z}_N = \mathbb{Z}/N\mathbb{Z}$, the integers modulo N
- $\mathbb{F}[x]$, the ring of polynomials with coefficients in a field

Algebraic fields

All the properties of a ring, except that $\times$ also has inverses.

- A field is a set closed under two operations: $+$, $\times$
- $+$ has identity, inverses, associative, commutative
- $\times$ has identity, inverses, associative, commutative
- Distributive law for $+$, $\times$

Examples to remember:

- $\mathbb{Q}$ the field of rational numbers
- $\mathbb{C}$ the field of complex numbers
- $\mathbb{F}_p$ the field of integers modulo p (also written $\mathbb{Z}_p$ or $GF(p)$ sometimes)
- $\mathbb{F}(x)$ the field of rational functions

Euler Totient Function

Recall: $\mathbb{Z}_N^* = \{z \in \mathbb{Z}_N \text{ s.t. } \gcd(z, N) = 1\}$
This is an abelian group under $\times$.

$\phi(N) = |\mathbb{Z}_N^*| =$ order of $\mathbb{Z}_N^*$ (Euler ϕ) = totient

- $\phi(p) = p - 1$
- $\phi(p^k) = p^{k-1}(p-1)$
- $\phi(N = \prod_i p_i^{e_i}) = \prod_i p_i^{e_i - 1}(p_i - 1)$

Theorem (Euler)

$a \in \mathbb{Z}_N^*$, $a^{\phi(N)} \equiv 1 \bmod N$

Proof.

Let $G = \langle a \rangle$. By Lagrange's theorem, $a^{|G|} = 1$ and $|G| \mid |\mathbb{Z}_N^*|$. $\square$

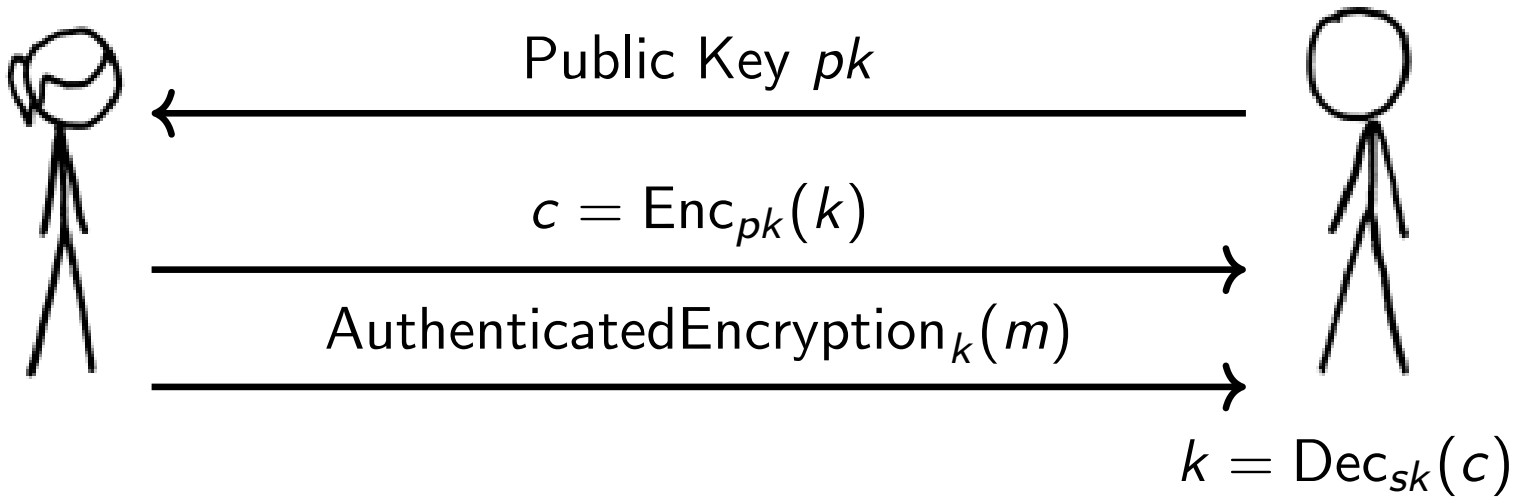
Idea # 2: Public-key encryption

Public Key

pk

Secret Key

sk



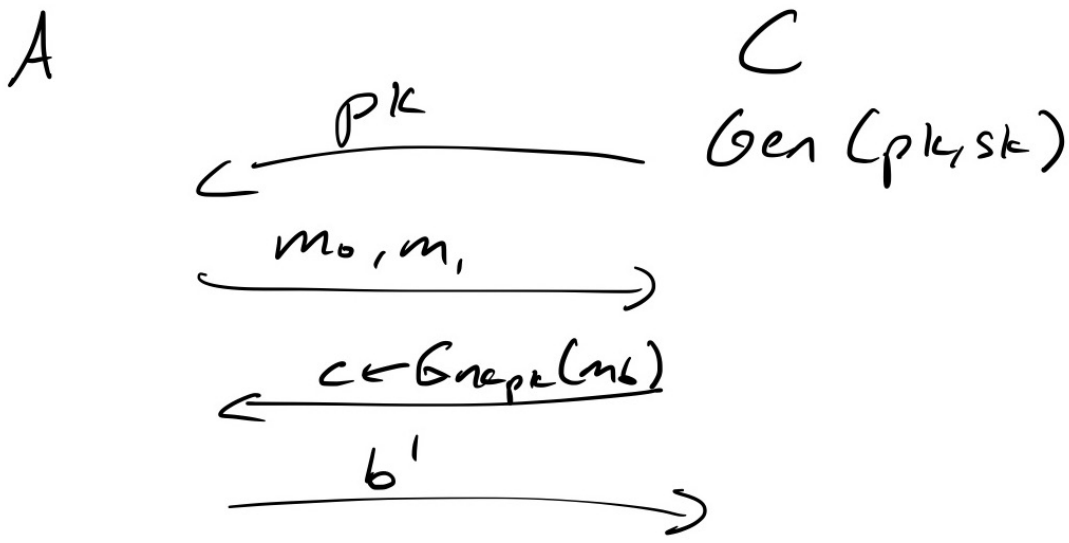
Keys come in pairs (pk, sk); pk can only be used to encrypt and only sk can be used to decrypt.

Formally specifying public-key encryption

- Key generation: Generate public key pk and secret key sk.
- Encryption: $c \leftarrow \mathsf{Enc}_{pk}(m)$.
- Decryption: $m \leftarrow \mathsf{Dec}_{sk}(c)$.

Correctness: $\Pr[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m] = 1$

Semantic security for public-key encryption



Definition

A public-key encryption scheme is semantically secure if

$$|\Pr[\text{A outputs } 1 | b = 1] = \Pr[\text{A outputs } 1 | b = 0]| \text{ is negligible}$$

- Randomized encryption is required for semantic security
- In the public-key setting, semantic security implies CPA security

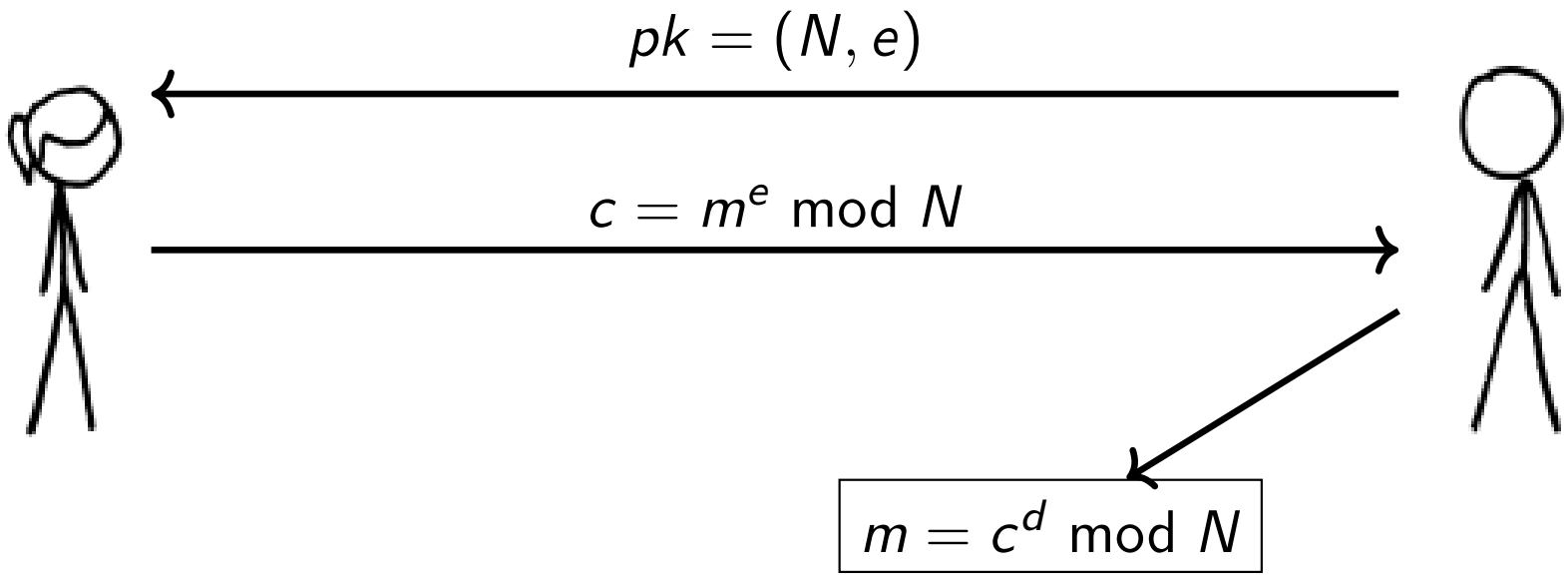
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R.L. Rivest, A. Shamir, and L. Adleman*

Textbook RSA Encryption

[Rivest Shamir Adleman 1977]

- Key Generation:
  1. Generate random primes p, q
  2. $N = pq$
  3. Choose odd e s.t. $\gcd(e, (p-1)(q-1)) = 1$
  4. $d = e^{-1} \bmod (p-1)(q-1)$
  5. $pk = (N, e)$, $sk = (N, d)$.
- Encryption: $c = m^e \bmod N$
- Decryption: $m = c^d \bmod N$



RSA Decryption works

- Key Generation:
  1. $N = pq$
  2. $d = e^{-1} \bmod (p-1)(q-1)$
  3. $pk = (N, e)$, $sk = (N, d)$.
- Encryption: $c = m^e \bmod N$
- Decryption: $m = c^d \bmod N$

Theorem

RSA Decryption is correct: $\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(m)) = m$

Proof.

$$m^{ed} = m^{1+k\phi(N)} \bmod N$$

Case 1: $m \in \mathbb{Z}_N^* \implies m^{\phi(N)} \equiv 1 \bmod N$

Case 2: $m \notin \mathbb{Z}_N^* \implies p|m$ or $q|m$

$$\left.\begin{aligned} m \equiv 0 \bmod p &\implies m^{ed} \equiv m \bmod p \\ m^{ed} = m^{1+k'(q-1)} \bmod q &\equiv m \bmod q \end{aligned}\right\} \text{CRT} \to m \bmod N$$

Factoring assumption

Assumption: Given $N = pq$, hard to compute p, q efficiently.

- Widely assumed to be hard for properly chosen p, q larger than 1024 bits.
- Equivalent to computing secret key d: compute

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

and then secret key

$$d = e^{-1} \bmod (p-1)(q-1)$$

- Factoring is in NP $\cap$ coNP so not likely NP-complete.
- Polynomial time factoring with quantum computers.

RSA assumption: eth roots mod N

Input: N, e, y
Output: x st.t $x^e \equiv y \bmod N$.

- Equivalent to decrypting RSA ciphertext.
- Factoring is easy $\implies$ RSA is easy to break
- RSA assumption is easy to break $\stackrel{?}{\implies}$ factoring

Semantically secure encryption from RSA

Textbook RSA as we described it is not semantically secure: it is deterministic!

Semantically secure encryption from RSA

Textbook RSA as we described it is not semantically secure: it is deterministic!

Fix: Use a symmetric cipher $(\mathsf{SymEnc}, \mathsf{SymDec})$ and a hash function H.

- Key Generation:
  1. Generate primes p, q; $N = pq$
  2. Choose odd e s.t. $\gcd(e, \phi(N)) = 1$
  3. $d = e^{-1} \bmod \phi(N)$
  4. $pk = (N, e)$, $sk = (N, d)$.

- Encryption: Choose random x, $y = x^e \bmod N$; $k = H(x)$;
  $$c = \mathsf{SymEnc}_k(m), \text{ send } (y, c)$$

- Decryption: Input (y, c), compute $x = y^d \bmod N$; $k = H(x)$; $m = \mathsf{SymDec}_k(c)$

Textbook ElGamal public-key encryption

Global parameters: A cyclic group G with generator g of prime order q

- Key Generation: Choose $a \in \mathbb{Z}_q$ and compute $u = g^a$. Then $pk = u$ and $sk = a$.
- Encryption: Input public key $pk = u$. Choose $b \in \mathbb{Z}_q$. Compute $v = g^b$. $\mathrm{Enc}_{pk}(m) = (v, u^b \cdot m)$.
- Decryption: Input a ciphertext (v, e), secret key $sk = a$. $\mathrm{Dec}_{sk}((v, e)) = e/v^a$.

This is semantically secure, but easy to implement poorly in practice.

Textbook RSA is insecure

Small e attack:
If $e = 3$ and $m < N^{1/3}$, $m = c^{1/3}$ over $\mathbb{Z}$.

Textbook RSA is insecure

Small e attack:
If $e = 3$ and $m < N^{1/3}$, $m = c^{1/3}$ over $\mathbb{Z}$.

Cube roots over $\mathbb{Z}$: polynomial time

Cube roots over $\mathbb{Z}/N\mathbb{Z}$: not efficient unless factorization of N is known

Textbook RSA is insecure

Small e with Chinese Remainder Theorem

Assume three parties have RSA keys with $e = 3$:

$$(3, N_1) \quad (3, N_2) \quad (3, N_3)$$

And the same (full-length) message is encrypted to each:

$$c_1 = m^3 \bmod N_1 \quad c_2 = m^3 \bmod N_2 \quad c_3 = m^3 \bmod N_3$$

Use the Chinese remainder theorem to reconstruct

$$c \equiv m^3 \bmod N_1 N_2 N_3$$

Now since $m^3 < N_1 N_2 N_3$ we are in the same situation as before.

Textbook RSA is insecure

Meet-in-the-middle attack for random m

Input ciphertext c.

- Compute $x_i = c/r^e \bmod N$ for r from $1, \ldots, \sqrt{m}$.
- Compute $y_i = s^e \bmod N$ for s from $1, \ldots, \sqrt{m}$.
- If $y_i = x_j$ return $r \cdot s$.

If $m = r \cdot s$, $m^e = r^e \cdot s^e$.

For a randomly chosen m, good probability it has no large factors.

- HW 5 due in 1.5 weeks!