CSE 291-E: Applied Cryptography

Nadia Heninger

UCSD

Fall 2020 Lecture 10

Legal Notice

The Zoom session for this class will be recorded and made available asynchronously on Canvas to registered students.

Announcements

1. HW 4 is due today!
2. HW 5 is due in one week!

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 $|G| \mid |\mathbb{Z}_N^*|$. $\square$

Chinese remainder theorem

Theorem

$N = pq$ p, q relatively prime, not necessarily prime.

$$\mathbb{Z}_N \simeq \mathbb{Z}_p \times \mathbb{Z}_q \qquad \mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$$

$f : x \to (x \bmod p, x \bmod q)$ is an explicit isomorphism.

Theorem (Alternative Statement)

Given r_1, r_2 there is a unique $x \in \mathbb{Z}_N$ satisfying $x \equiv r_1 \bmod p$, $x \equiv r_2 \bmod q$.

Chinese Remainder Theorem, constructively

Want to solve for x with $x \equiv r_p \bmod p \quad x \equiv r_q \bmod q$.

$$\gcd(p, q) = 1 \qquad \Longrightarrow \qquad ap + bq = 1$$

$$bq \equiv 1 \bmod p \quad bq \equiv 0 \bmod q$$
$$ap \equiv 0 \bmod p \quad ap \equiv 1 \bmod q$$

Think of these as "basis vectors".

Then to solve $x \equiv r_p \bmod p \quad x \equiv r_q \bmod q$, multiply the "magnitudes" by the "basis vectors":

$$x = (1, 0) \cdot r_p + (0, 1) \cdot r_q = bq \cdot r_p + ap \cdot rq$$

Chinese Remainder Theorem in more "dimensions"

Want to solve for x with $x \equiv r_1 \bmod p_1 \ldots x \equiv r_k \bmod p_k$.

Let $N = \prod_i p_i$ and then set $N_i = N/p_i$.

$$\gcd(p_i, N_i) = 1 \quad \Longrightarrow \quad a_i p_i + b_i N_i = 1$$

Our "basis vectors":

$$b_i N_i \equiv 1 \bmod p_i \quad b_i N_i \equiv 0 \bmod p_j \quad j \neq i$$

Then to solve $x \equiv r_i \bmod p_i$,

$$x = \sum_i r_i b_i N_i$$

Pohlig-Hellman Algorithm for Discrete Logs

How to compute discrete logs when the group order is composite

Want to solve $g^x = y \bmod p$. Know $\text{ord}(g) | p - 1$.

Say $\text{ord}(g) = m = p_1 p_2 \dots p_r$ so is composite.

Pohlig-Hellman Algorithm for Discrete Logs

How to compute discrete logs when the group order is composite

Want to solve $g^x = y \bmod p$. Know $\text{ord}(g) | p - 1$.

Say $\text{ord}(g) = m = p_1 p_2 \dots p_r$ so is composite.

Algorithm:

1. Solve the discrete log in each prime-order subgroup.
2. Use the Chinese Remainder Theorem to reconstruct the discrete log for the whole group.

Solving the discrete log in a prime-order subgroup

Want to solve $g^x = y \bmod p$ where
$\mathrm{ord}(g) = m = p_1 p_2 \ldots p_r \mid p - 1$.

Solving the discrete log in a prime-order subgroup

Want to solve $g^x = y \bmod p$ where
$\text{ord}(g) = m = p_1 p_2 \ldots p_r \mid p - 1$.

By Lagrange's theorem, $g^m = 1 \bmod p$.

So g^{m/p_i} generates group of order $p_i \bmod p$.

We know:

$$\begin{aligned} g^x &\equiv y \bmod p \\ (g^x)^{m/p_i} &\equiv y^{m/p_i} \bmod p \\ \left(g^{m/p_i}\right)^x &\equiv y^{m/p_i} \bmod p \end{aligned}$$

Solving the discrete log in a prime-order subgroup

Want to solve $g^x = y \bmod p$ where
$\text{ord}(g) = m = p_1 p_2 \dots p_r \mid p - 1$.

By Lagrange's theorem, $g^m = 1 \bmod p$.

So g^{m/p_i} generates group of order $p_i \bmod p$.

We know:

$$\begin{aligned} g^x &\equiv y \bmod p \\ (g^x)^{m/p_i} &\equiv y^{m/p_i} \bmod p \\ \left(g^{m/p_i}\right)^x &\equiv y^{m/p_i} \bmod p \end{aligned}$$

Solving this discrete log problem, we learn a relation

$$x \equiv x_i \bmod p_i$$

Applying the Chinese Remainder Theorem

Solve discrete logs as above, we get many equivalences:

$$x \equiv x_1 \bmod p_1$$
$$x \equiv x_2 \bmod p_2$$
$$\vdots$$
$$x \equiv x_r \bmod p_r$$

Use CRT to construct $x \bmod \prod_i p_i$.

Total algorithm time dominated by discrete log:
$O(\text{polylog}(p) \cdot max_i \sqrt{p_i})$

Countermeasures

Always do Diffie-Hellman and other discrete log-based cryptography over large prime order subgroups.

Common choices for p:

- "Safe" primes $p = 2q + 1$, with q prime, choose g so it generates group of order q.
- Some implementations choose q much smaller (e.g. 256 bits) and construct $p = qh + 1$ and choose g so it generates group of order q.

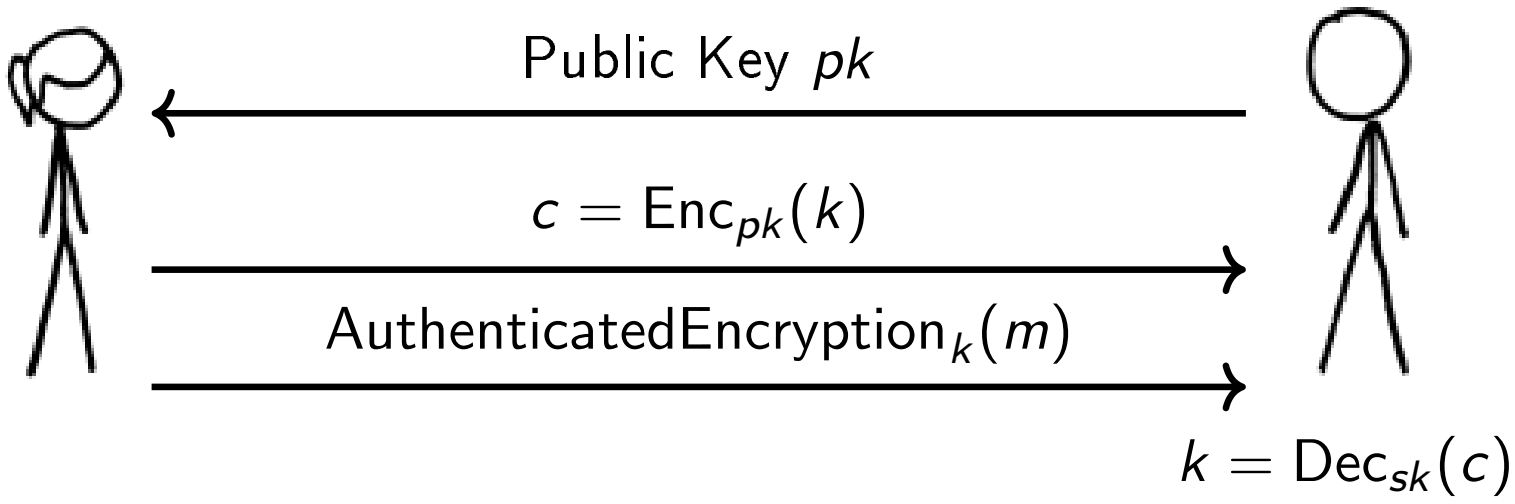
Idea # 2: Public-key encryption

Public Key

pk

Secret Key

sk



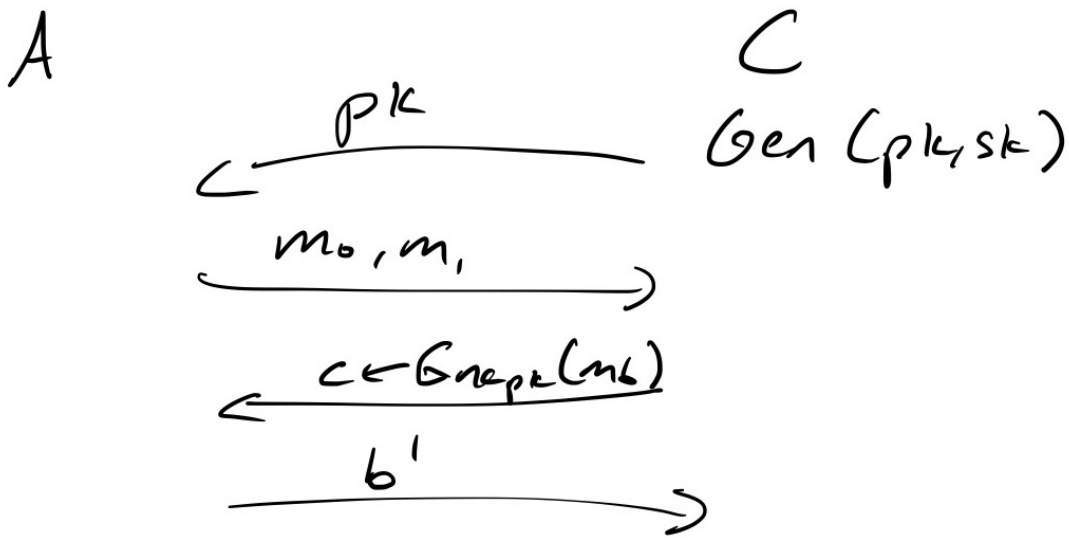
Keys come in pairs (pk, sk); pk can only be used to encrypt and only sk can be used to decrypt.

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

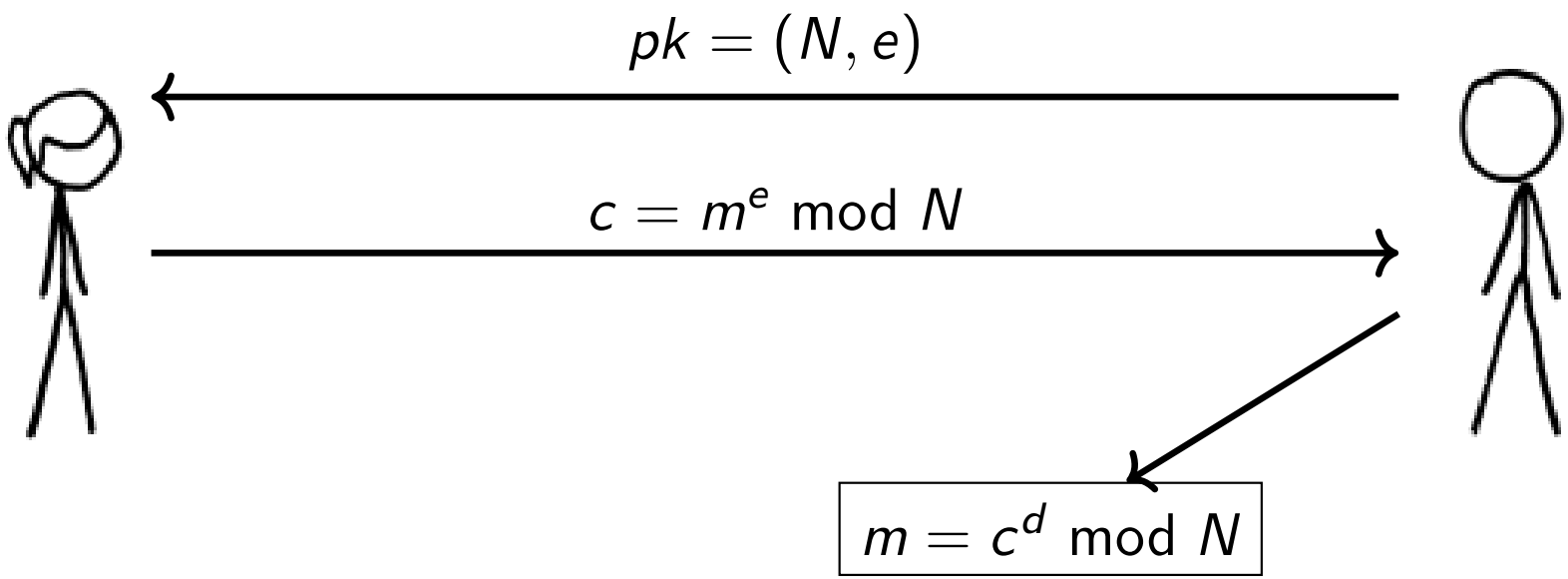
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

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: $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$

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 $q|m$

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

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 $\overset{?}{\implies}$ 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.

- HW 5 due in one week!