CSE 107: Introduction to Modern Cryptography

Nadia Heninger

UCSD

Winter 2025 Lecture 13

Last time:

- Diffie-Hellman.

This time:

- Elementary discrete log algorithms
- More number theory: rings, fields, Euler ϕ
- Public-key cryptography: RSA, factoring assumptions, KEMs, hybrid encryption

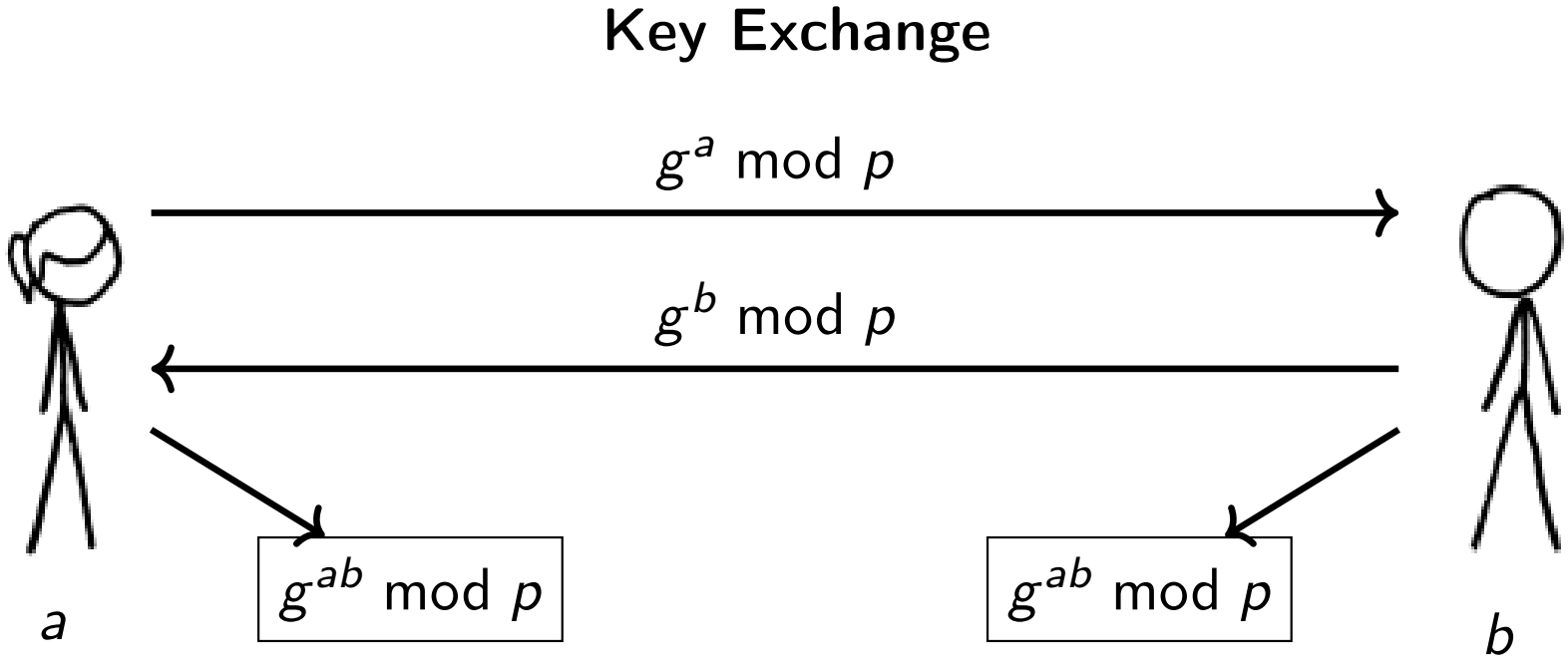
Last Time: Prime Field Diffie-Hellman Key Exchange

Public Parameters

p a prime

q a subgroup order; $q \mid (p-1)$

$g \in \mathbb{F}_p^*$ a generator of subgroup of order q



Why is Diffie-Hellman considered secure?

The best algorithms we have for breaking it are to compute the discrete logarithm of the public key exchange value.

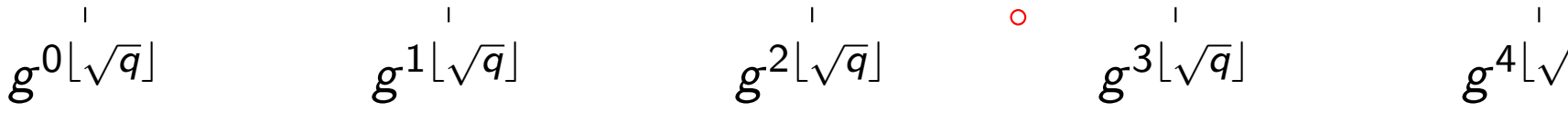
Computing Discrete Logs in $O(\sqrt{q})$ time

Input: Target t, prime p, group gen g, order q

Goal: Find ℓ where $g^\ell \equiv t \bmod p$.

Baby-Step Giant-Step Algorithm

1. "Giant steps": Compute $g^0, g^{\lfloor\sqrt{q}\rfloor}, g^{2\lfloor\sqrt{q}\rfloor}, \ldots g^{\lfloor\sqrt{q}\rfloor^2} \bmod p$.



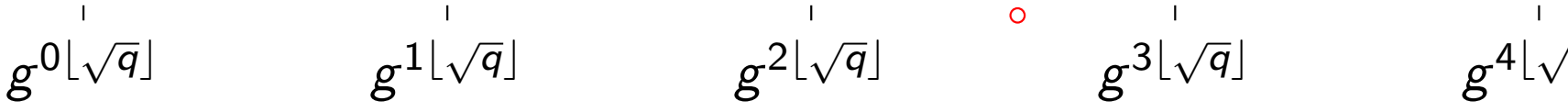
Computing Discrete Logs in $O(\sqrt{q})$ time

Input: Target t, prime p, group gen g, order q

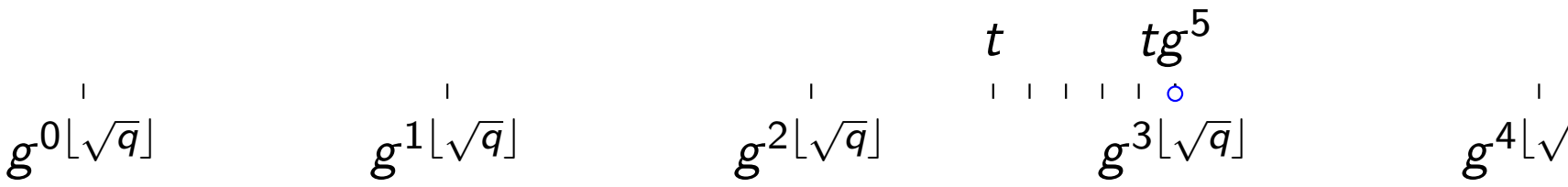
Goal: Find ℓ where $g^\ell \equiv t \bmod p$.

Baby-Step Giant-Step Algorithm

1. "Giant steps": Compute $g^0, g^{\lfloor\sqrt{q}\rfloor}, g^{2\lfloor\sqrt{q}\rfloor}, \ldots g^{\lfloor\sqrt{q}\rfloor^2} \bmod p$.



2. "Baby steps": Compute $tg^1, tg^2, \ldots \bmod p$ until collision.



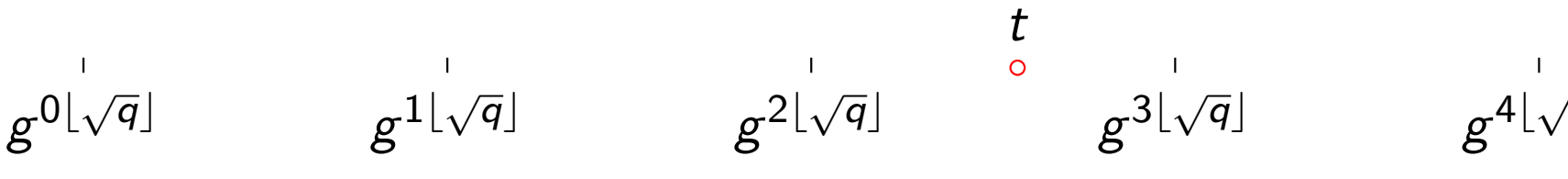
Computing Discrete Logs in $O(\sqrt{q})$ time

Input: Target t, prime p, group gen g, order q

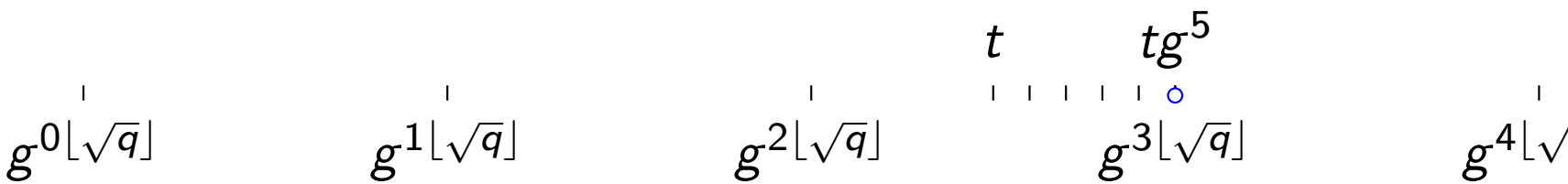
Goal: Find ℓ where $g^\ell \equiv t \bmod p$.

Baby-Step Giant-Step Algorithm

1. "Giant steps": Compute $g^0, g^{\lfloor\sqrt{q}\rfloor}, g^{2\lfloor\sqrt{q}\rfloor}, \ldots g^{\lfloor\sqrt{q}\rfloor^2} \bmod p$.



2. "Baby steps": Compute $tg^1, tg^2, \ldots \bmod p$ until collision.



3. Solve for $\log t$ from collision:

$$tg^j = g^{i\lfloor\sqrt{q}\rfloor} \bmod p$$

$$t = g^{i\lfloor\sqrt{q}\rfloor - j} \bmod p$$

$$\log t = i\lfloor\sqrt{q}\rfloor - j \bmod q$$

Baby Step Giant Step running time

- Storage: $O(\sqrt{q})$ group elements
- Running time: $O(\sqrt{q})$ group multiplications
- Running time: $O(\sqrt{q})$ table lookups

Total: $O(\sqrt{q}\,\mathrm{polylog}\,p)$

Baby Step Giant Step running time

- Storage: $O(\sqrt{q})$ group elements
- Running time: $O(\sqrt{q})$ group multiplications
- Running time: $O(\sqrt{q})$ table lookups

Total: $O(\sqrt{q}\,\mathrm{polylog}\,p)$

Idea: Reduce storage using random walk. This leads to the Pollard rho algorithm, which also has $O(\sqrt{q})$ running time but constant storage.

Selecting good Diffie-Hellman parameters

To protect against these discrete logarithm algorithms, we must 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.

To protect against number field sieve attacks, we have to choose p with 2048 bits or more.

Idea # 2: Public-key encryption

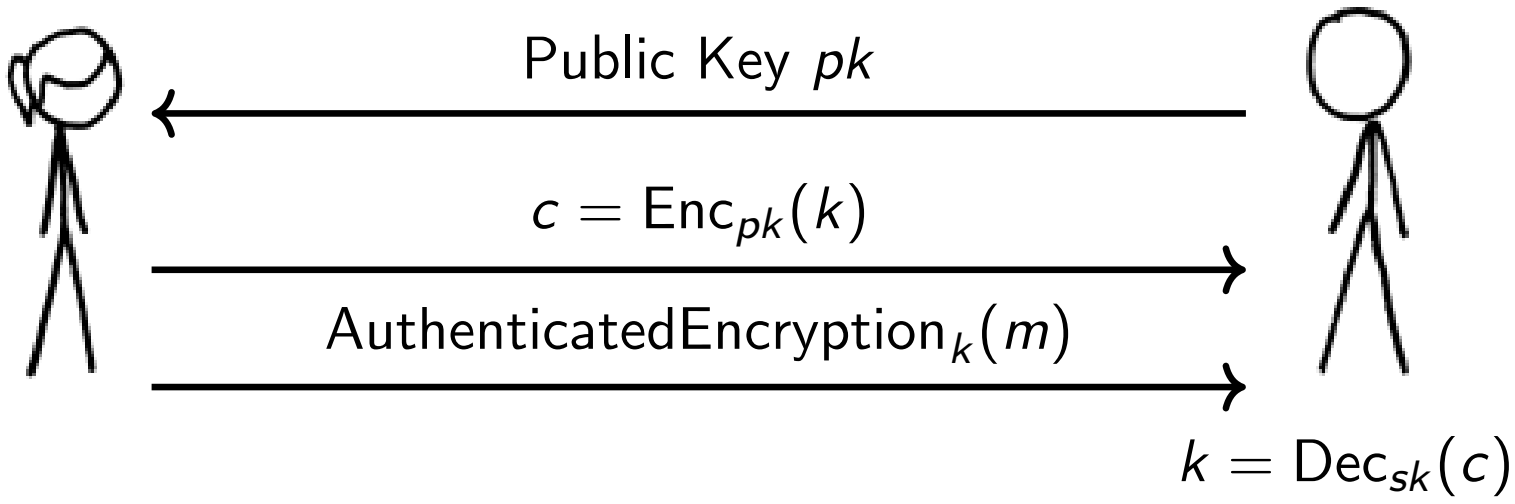
Keys come in pairs (pk, sk); pk can only be used to encrypt and only sk can be used to decrypt.

Public Key

pk

Secret Key

sk



Formally specifying public-key encryption

A public-key (or asymmetric) encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms that operate as follows:

- Key generation: $(ek, dk) \xleftarrow{\$} \mathcal{K}$
- Encryption: $C \xleftarrow{\$} \mathcal{E}_{ek}(M)$: encrypt message M under encryption key ek to get a ciphertext C. Algorithm $\mathcal{E}$ may be randomized.
- Decryption: $M' \leftarrow \mathcal{D}_{dk}(C)$ — decrypt ciphertext C under decryption key dk to get an output $M' \in \{0,1\}^* \cup \{\perp\}$.

Correct Decryption Requirement: $\Pr[\mathcal{D}_{dk}(\mathcal{E}_{ek}(M)) = M] = 1$

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

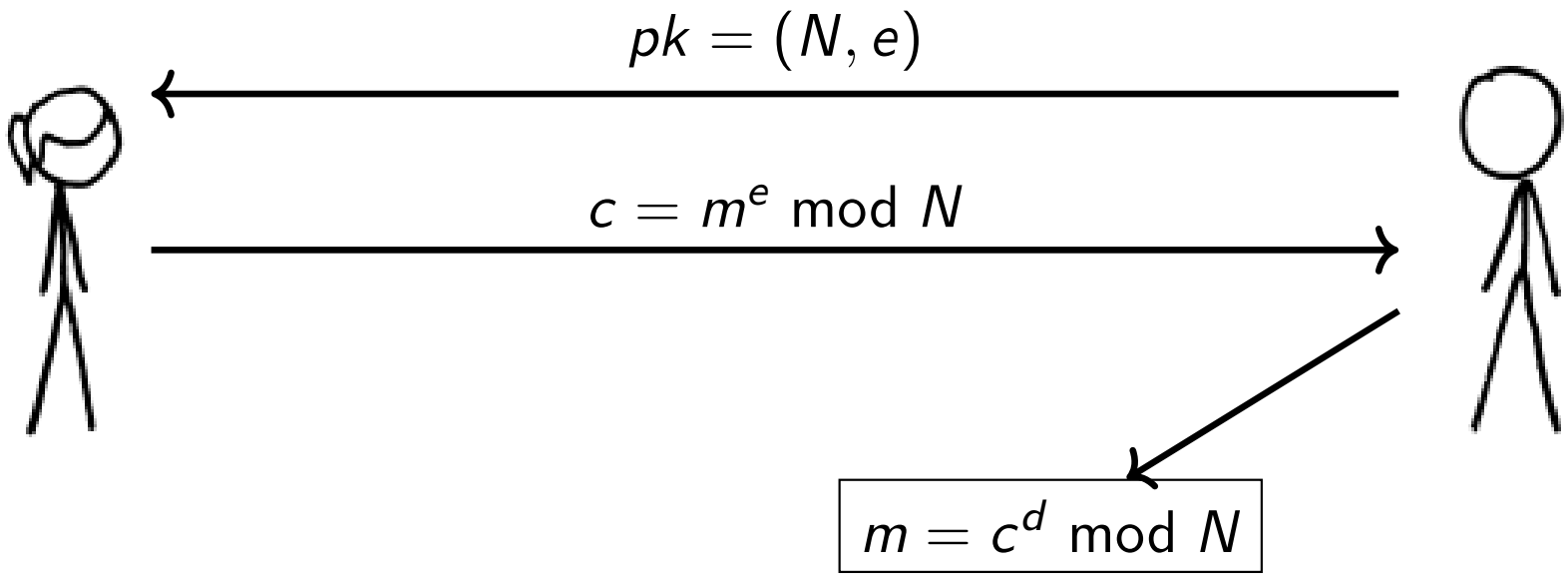
A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

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 N larger than 2048 bits.
- Equivalent to computing RSA 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.
- Factoring is polynomial time factoring with a quantum computer.

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

Security of PKE Schemes

We formalize two goals:

- **IND-CPA**: Indistinguishability under chosen-plaintext attack. Just like for symmetric encryption, except that adversary needs to be given the encryption key.
- **IND-CCA**: Indistinguishability under chosen-ciphertext attack. A stronger goal in which the adversary has (limited) access to a decryption oracle.

IND-CPA security games

Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a PKE scheme and A an adversary.

Game $\mathrm{Left}_{\mathcal{AE}}$	Game $\mathrm{Right}_{\mathcal{AE}}$
procedure Initialize	**procedure Initialize**
$(ek, dk) \xleftarrow{\$} \mathcal{K}$; return ek	$(ek, dk) \xleftarrow{\$} \mathcal{K}$; return ek
procedure LR(M_0, M_1)	**procedure LR**(M_0, M_1)
Return $C \xleftarrow{\$} \mathcal{E}_{ek}(M_0)$	Return $C \xleftarrow{\$} \mathcal{E}_{ek}(M_1)$

Associated to $\mathcal{AE}, A$ are the probabilities

$$\Pr\left[\mathrm{Left}^{A}_{\mathcal{AE}} \Rightarrow 1\right] \qquad \Pr\left[\mathrm{Right}^{A}_{\mathcal{AE}} \Rightarrow 1\right]$$

that A outputs 1 in each world. The ind-cpa advantage of A is

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE}}(A) = \Pr\left[\mathrm{Right}^{A}_{\mathcal{AE}} \Rightarrow 1\right] - \Pr\left[\mathrm{Left}^{A}_{\mathcal{AE}} \Rightarrow 1\right]$$

PKE IND-CPA: Explanations

The "return ek" statement in Initialize means the adversary A gets the encryption key ek as input. It does not get dk.

It can call **LR** with any equal-length messages M_0, M_1 of its choice to get back an encryption $C \xleftarrow{\$} \mathcal{E}_{ek}(M_b)$ of M_b under ek, where $b = 0$ in game $\mathrm{Left}_{\mathcal{AE}}$ and $b = 1$ in game $\mathrm{Right}_{\mathcal{AE}}$. Notation indicates encryption algorithm may be randomized.

A is not allowed to call **LR** with messages M_0, M_1 of unequal length. Any such A is considered invalid and its advantage is undefined or 0.

It outputs a bit, and wins if this bit equals b.

Textbook RSA is not CPA-secure

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

Exercise: Prove this. There is a 2-query attack that is nearly identical to the ones you wrote before.

Textbook RSA is insecure

Some variants of RSA have even easier attacks.

Small e attack:

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

Textbook RSA is insecure

Some variants of RSA have even easier attacks.
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