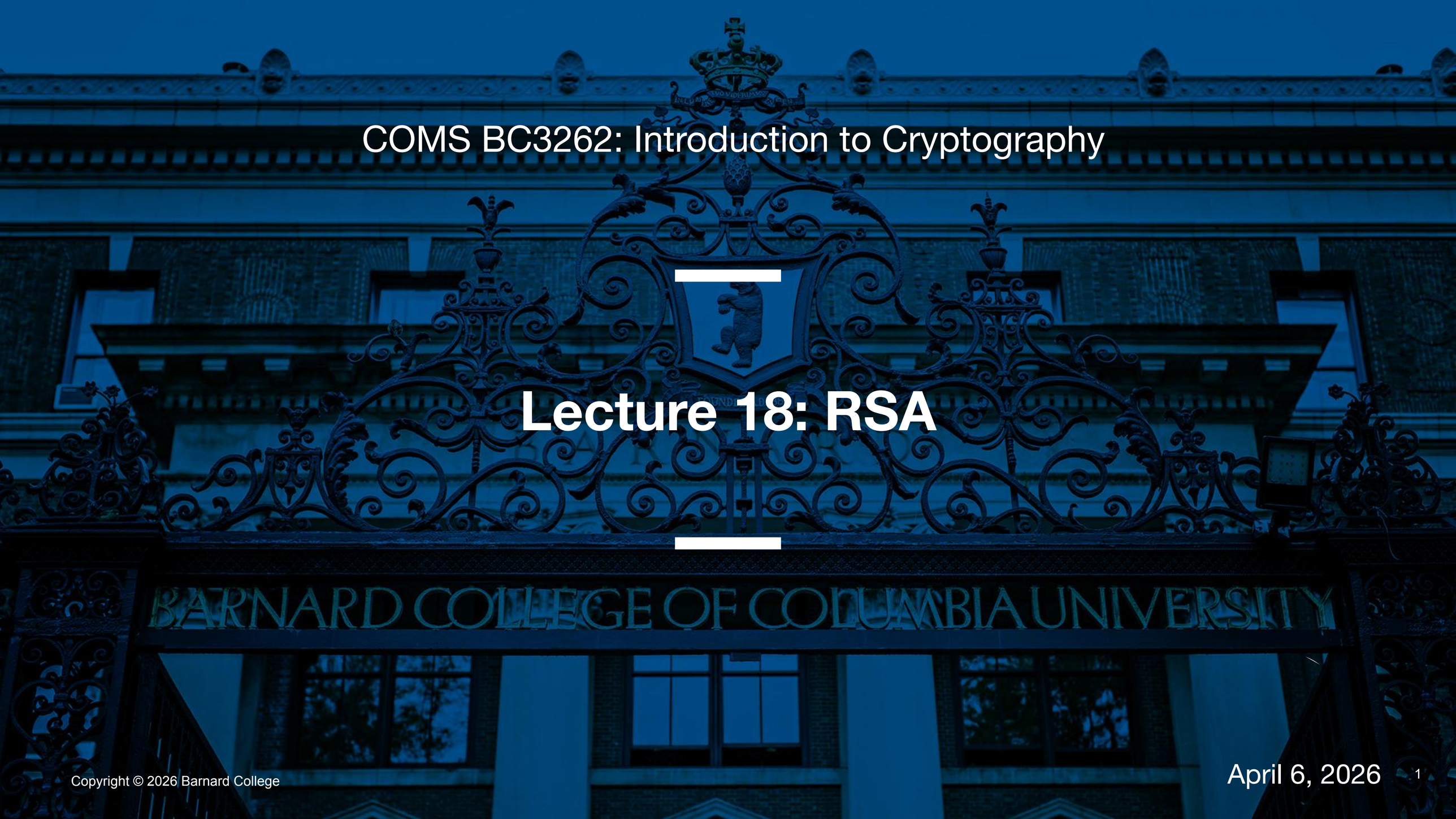


COMS BC3262: Introduction to Cryptography

Lecture 18: RSA

April 6, 2026

Logistics

- **Extra credit opportunity**: Attend Olive's talk Thursday, April 16 at 12pm in Milstein 402

  - Olive Franzese-McLaughlin will be giving a talk titled "Cryptographically Verified AI/ML Audits"

- PS 4 was released last week

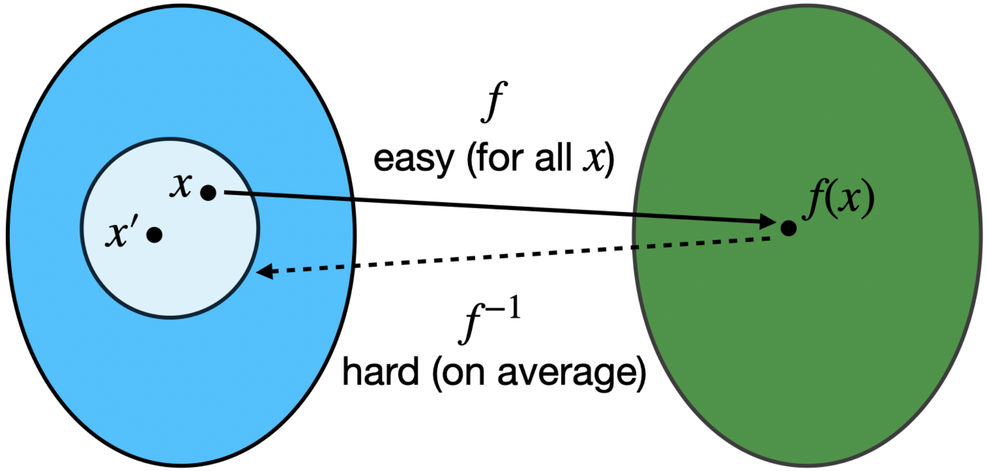
Last Time: Factoring Assumption

Factoring Assumption

Informally, “multiplying two large primes is a OWF”:

Given a composite N, it’s hard to find p, q such that $pq = N$

A function $f : \{0,1\}^* \to \{0,1\}^*$ that is easy to compute but hard to invert



See Lecture 8 for introduction of OWFs

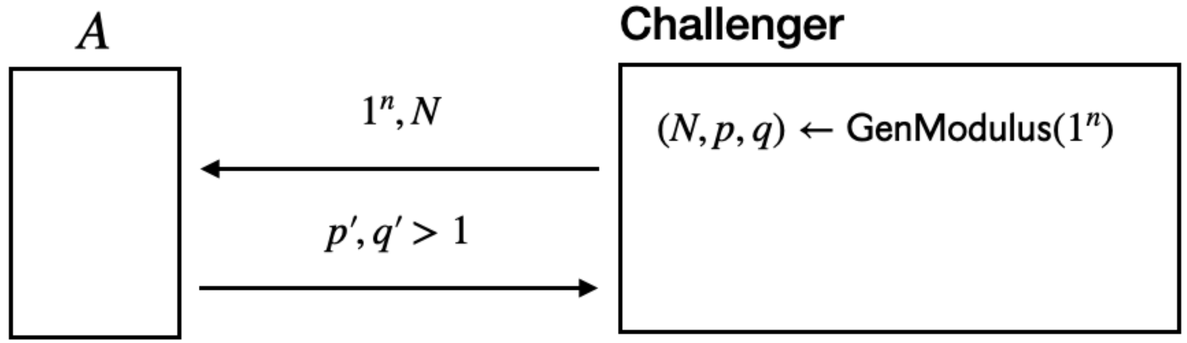
Factoring Assumption

Suppose we had an algorithm GenModulus that on input 1^n outputs (N, p, q) such that $N = p \cdot q$ and p, q are n-bit primes

Definition:

The Factoring is hard relative to GenModulus if for all PPT adversaries A there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr[\mathsf{Factor}_{A,\mathsf{GenModulus}}(n) = 1] \leq \mathsf{negl}(n)$$



$$\mathsf{Factor}_{A,\mathsf{GenModulus}}(n) = \begin{cases} 1 & N = p'q' \\ 0 & \text{otherwise} \end{cases}$$

Number Theory

Recall: The Group $\mathbb{Z}_N^*$

For $N > 1$, define $\mathbb{Z}_N^* = \{a \in \{1,\ldots,N-1\} \mid \gcd(a,N) = 1\}$ with the group operation $ab = [ab \bmod N]$

- Example:
  - $\mathbb{Z}_{15}^* = \{1,2,4,7,8,11,13,14\}$
  - $\mathbb{Z}_{11}^* = \{1,2,3,4,5,6,7,8,9,10\}$

Recall: The Group $\mathbb{Z}_N^*$

Definition: Euler's Totient Function $\phi(N)$ is the order of $\mathbb{Z}_N^*$

Theorem: Let $N = \Pi_i p_i^{e_i}$, where p_i are distinct primes and $e_i \geq 1$. Then

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

Example:

- $\phi(15) = (3-1)(5-1) = 8$
- $\mathbb{Z}_{15}^* = \{1,2,4,7,8,11,13,14\}$

Recall: The Group $\mathbb{Z}_N^*$

Theorem: Let $\mathbb{G}$ be a finite group of order $m = |\mathbb{G}|$. Then, for every $a \in \mathbb{G}$, it holds that $a^m = 1$

Corollary: Let $N > 1$ and let $a \in \mathbb{Z}_N^*$. Then

$$a^{\phi(N)} = 1 \mod N$$

If $N = p$ is prime and $a \in \{1, \ldots, p-1\}$ then

$$a^{p-1} = 1 \mod p$$

Permutation on $\mathbb{Z}_N^*$

Corollary: Let $N > 1$ and let $a \in \mathbb{Z}_N^*$. Then $a^{\phi(N)} = 1 \mod N$

If $N = p$ is prime and $a \in \{1, \ldots, p-1\}$ then $a^{p-1} = 1 \mod p$

Corollary: Let $N > 1$ and let $e > 0$ be an integer. Define $f_{N,e} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ by

$$f_{N,e}(x) = [x^e \mod N]$$

If $\gcd(e, \phi(N)) = 1$, then $f_{N,e}$ is a permutation.

Further, if $d = e^{-1} \mod \phi(N)$ then $f_{N,d}$ is the inverse of $f_{N,e}$

Permutation on $\mathbb{Z}_N^*$

Corollary: Let $N > 1$ and let $a \in \mathbb{Z}_N^*$. Then $a^{\phi(N)} = 1 \mod N$

Corollary: Let $N > 1$ and let $e > 0$ be an integer. Define $f_{N,e} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ by

$$f_{N,e}(x) = [x^e \mod N]$$

If $\gcd(e, \phi(N)) = 1$, then $f_{N,e}$ is a permutation.

Further, if $d = e^{-1} \mod \phi(N)$ then $f_{N,d}$ is the inverse of $f_{N,e}$

Proof sketch:

$$f_{N,d}(y) = f_{N,e}^{-1}(y) = y^d = (x^e)^d \mod N = x^{1 \mod \phi(N)} = x \mod N$$

The RSA Assumption

Let $N = pq$ where p, q are primes, $p \neq q$ and $|p| = |q|$, and let $e > 2$ be such that $\gcd(e, \phi(N)) = 1$

The RSA function is the permutation $f_{N,e}(x) = [x^e \bmod N]$

Informally, RSA Assumption states given y, e, and N, finding $[y^{1/e} \bmod N]$ is hard

- Note we work $\bmod N$ in the group and $\bmod \phi(N)$ in the exponent
- What is $\phi(N)$? (i.e., the order of the group $\mathbb{Z}_N^*$)
  - $\phi(N) = (p-1)(q-1)$

The RSA Assumption

Consider a PPT alg GenRSA that outputs (N, e, d) s.t. $ed = 1 \mod \phi(N)$

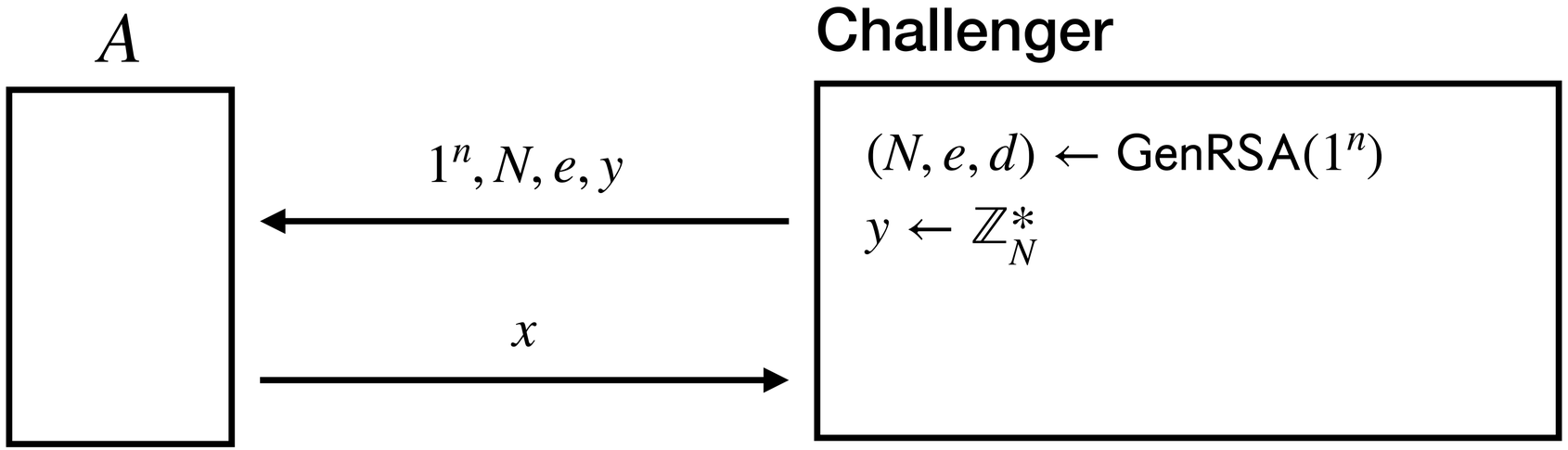
For example:

GenRSA(1^n) :

1. Compute $(N, p, q) \leftarrow$ GenModulus(1^n)
2. Choose $e > 2$ s.t. $\gcd(e, \phi(N)) = 1$
3. Find d s.t. $ed = 1 \mod \phi(N)$ (e.g., using extended GCD on e and $\phi(N)$)
4. Output (N, e, d)

The RSA Assumption

Consider a PPT alg GenRSA that outputs (N, e, d) s.t. $ed = 1 \mod \phi(N)$



$$\mathsf{RSAInv}_{A,\mathsf{GenRSA}}(n) = \begin{cases} 1 & x^e = y \mod N \\ 0 & \text{otherwise} \end{cases}$$

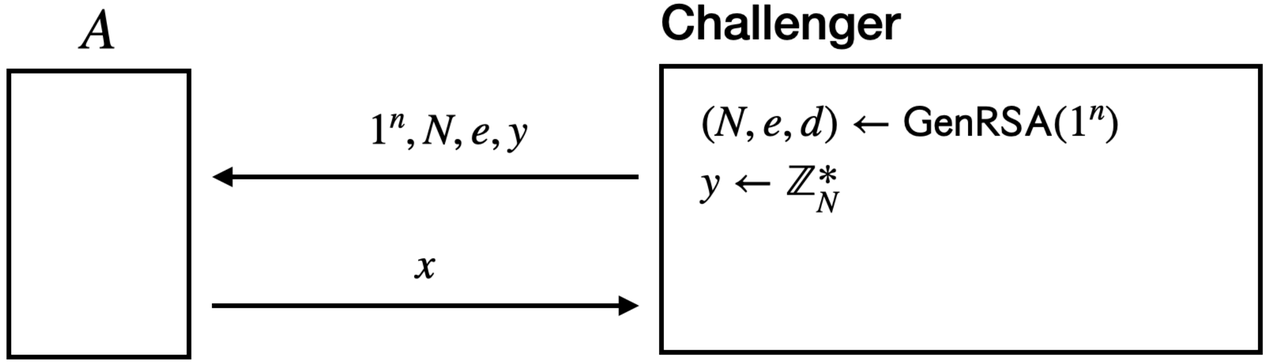
The RSA Assumption

Consider a PPT alg GenRSA that outputs (N, e, d) s.t. $ed = 1 \mod \phi(N)$

Definition:

The RSA Assumption holds relative to GenRSA if for all PPT adversaries A there exists a negligible function $\text{negl}(\cdot)$ such that

$$\Pr[\text{RSAInv}_{A,\text{GenRSA}}(n) = 1] \leq \text{negl}(n)$$



$$\text{RSAInv}_{A,\text{GenRSA}}(n) = \begin{cases} 1 & x^e = y \mod N \\ 0 & \text{otherwise} \end{cases}$$

The RSA Assumption

Consider a PPT alg GenRSA that outputs (N, e, d) s.t. $ed = 1 \mod \phi(N)$

Definition:

The RSA Assumption holds with respect to $\text{GenRSA}(1^n)$ if for all PPT adversaries A there exists a negligible function ϵ s.t.

$$\Pr_{\substack{(N,e,d) \leftarrow \text{GenRSA}(1^n) \\ y \leftarrow \mathbb{Z}_N^*}} \left[A(N, e, y) = x \text{ where } x^e = y \mod N\right] \leq \epsilon(n)$$

Equivalent way of writing it

RSA Example

- Say GenModulus outputs $(N = 143,\ p = 11,\ q = 13)$
  - What's $\phi(143)$? $(11 - 1)(13 - 1) = 120$
- Say we choose $e = 7$
  - Using extended GCD you can solve for its inverse: $d = 103$
- GenRSA outputs $(N = 143,\ e = 7,\ d = 103)$

RSA Example

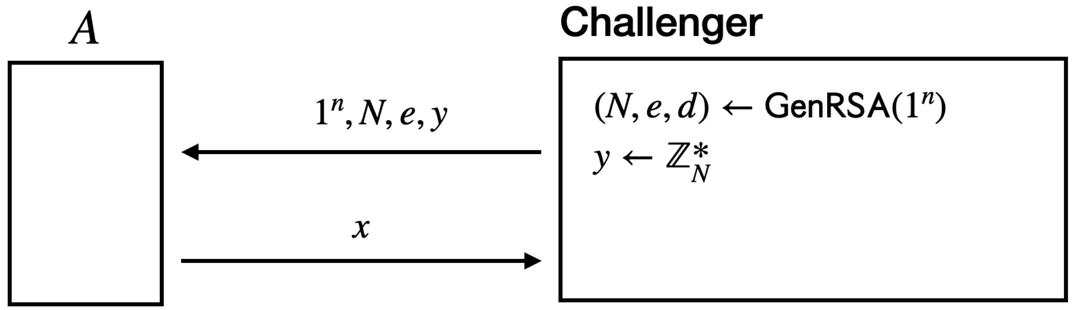
- GenRSA outputs $(N = 143,\ e = 7,\ d = 103)$
- RSA Assumption says given a random element from $\mathbb{Z}_N^*$, it should be hard to find the e-th root without knowing d or the factorization of N
  - Example: What's the 7th root of $y = 64$?
  - If given d, it's very easy: $64^d = 64^{103} = 25 \mod 143$

RSA vs Factoring

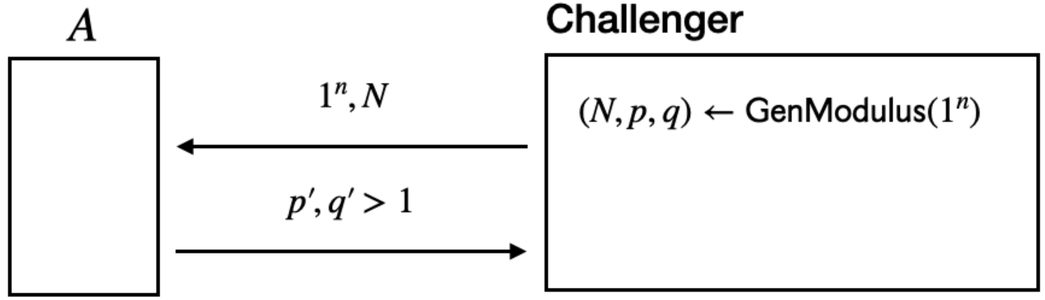
If factoring is easy, then RSA is easy

- Given (N, e, y), factor $N = pq$
- Compute $\phi(N) = (p-1)(q-1)$
- Extended GCD using e and $\phi(N)$ to compute d
- Compute $y^d \bmod N$

What about the other direction?



$$\text{RSAInv}_{A,\text{GenRSA}}(n) = \begin{cases} 1 & x^e = y \bmod N \\ 0 & \text{otherwise} \end{cases}$$

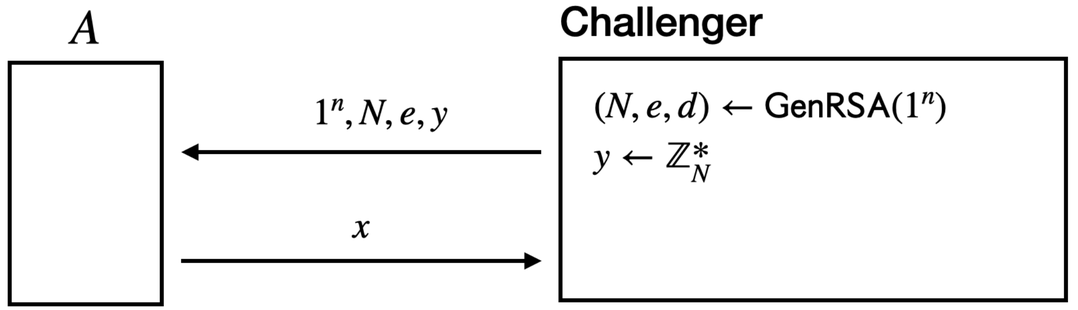


$$\text{Factor}_{A,\text{GenModulus}}(n) = \begin{cases} 1 & N = p'q' \\ 0 & \text{otherwise} \end{cases}$$

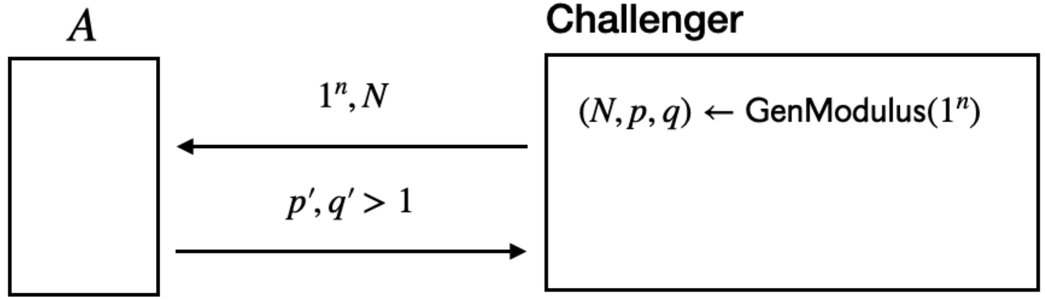
RSA vs Factoring

Other direction is a big open problem: it's possible factoring is hard but RSA isn't!

- Finding $\phi(N)$ from N is equivalent to factoring
- Finding d from (N, e) is equivalent to factoring
- Is it possible to find $y^{1/e} \bmod N$ without $\phi(N)$ or d?



$$\text{RSAInv}_{A,\text{GenRSA}}(n) = \begin{cases} 1 & x^e = y \bmod N \\ 0 & \text{otherwise} \end{cases}$$



$$\text{Factor}_{A,\text{GenModulus}}(n) = \begin{cases} 1 & N = p'q' \\ 0 & \text{otherwise} \end{cases}$$

RSA vs Discrete Log

RSA

- Group $\mathbb{Z}_N^*$ of unknown order $\phi(N)$
- Find x from $x^e \mod N$
  - e is known
- Trapdoor $d = e^{-1} \mod \phi(N)$
  - “Trapdoor permutation”

Discrete Log

- Cyclic group of known order q
- Find x from g^x
  - g is known
- No known trapdoor

RSA Encryption

Textbook RSA Encryption

Very straightforward way of encrypting

- Gen(1^n): Run $(N, e, d) \leftarrow$ GenRSA(1^n). Output the public key as $pk = (N, e)$ and the secret key as $sk = (N, d)$

- Enc$(pk, m) = m^e \bmod N$

- Dec$(sk, c) = c^d \bmod N$

Satisfies correctness, **but it's not secure** (there are MANY attacks)

Brief Detour: Hard-Core Predicates

- One-way functions say given $y = f(x)$ it's hard to find x in its entirety
- Hard-core predicates say given $y = f(x)$ it's hard to compute $\mathsf{hc}(x)$ where $\mathsf{hc} : \{0,1\}^* \rightarrow \{0,1\}$ with significantly greater than 1/2 probability
  - We can map x to a single bit that is hard to guess given $f(x)$
- **Claim**: The least-significant bit of x is a hard-core predicate for RSA
  - That is, given $N,\ e,\ y = [x^e \bmod N]$, it is hard for an adversary to guess the least significant bit of x if RSA assumption holds

Single-bit RSA Encryption

- Gen(1^n): Run $(N, e, d) \leftarrow \text{GenRSA}(1^n)$.

  Output the public key as $pk = (N, e)$ and the secret key as $sk = (N, d)$

- Enc $\left(pk, m \in \{0,1\}\right)$: Choose a random $r \leftarrow \mathbb{Z}_N^*$.

  Output $(r^e \bmod N, \text{lsb}(r) \oplus m)$ where lsb(r) returns the least-significant bit of r

- Dec(sk, c): Parse $c = (c_1, c_2)$ and compute $r = c_1^d \bmod N$.

  Output the message as $\text{lsb}(r) \oplus c_2$

CPA-Secure under the RSA assumption

Extends to longer messages by encrypting each bit of the message separately

Padded-RSA

- Gen(1^n): Run $(N, e, d) \leftarrow$ GenRSA(1^n).
  Output the public key as $pk = (N, e)$ and the secret key as $sk = (N, d)$

- Enc $\left(pk, m\right)$: To encrypt a message $m \in \{0,1\}^{|N|-\ell(n)-1}$,
  Choose a random $r \leftarrow \{0,1\}^{\ell(n)}$ and parse $\hat{m} = r||m$ as an element of $\mathbb{Z}_N^*$.
  Output $\hat{m}^e \mod N$

- Dec(sk, c): Compute $\hat{m} = c^d \mod N$ and
  output the $|N| - \ell(n) - 1$ least-significant bits of m

Intuitively, if $\ell(n)$ is very long, then $\hat{m}$ is close to random

CPA-secure under RSA Assumption if $|m| = O(\log(n))$

Hash-Based RSA

- Gen(1^n): Run $(N, e, d) \leftarrow \text{GenRSA}(1^n)$.

  Output the public key as $pk = (N, e)$ and the secret key as $sk = (N, d)$

- Enc $\left(pk, m\right)$: Choose a random $r \leftarrow \mathbb{Z}_N^*$.

  Output $(r^e \mod N, H(r) \oplus m)$ where H is a hash function

- Dec(sk, c): Parse $c = (c_1, c_2)$.

  Compute $r = c_1^d \mod N$ and output $m = H(r) \oplus c_2$

CPA-secure in the random oracle model under RSA Assumption

Next Time

- Digital Signatures!!