

COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Spring 2018

Number Theory

\mathbb{Z}_N : integers mod N

\mathbb{Z}_N^* : integers mod N that are relatively prime to N

- $x \in \mathbb{Z}_N^*$ iff x has an “inverse” y s.t. $xy \bmod N = 1$
- For prime N , $\mathbb{Z}_N^* = \{1, \dots, N-1\}$

$$\Phi(N) = |\mathbb{Z}_N^*|$$

Euler’s theorem: for any $x \in \mathbb{Z}_N^*$, $x^{\Phi(N)} \bmod N = 1$

Groups

A group is a set \mathbf{G} together with a binary operation \otimes

- $\mathbf{g} \otimes \mathbf{h} \in \mathbf{G}$
- \exists identity $\mathbf{1}$ s.t. $\mathbf{g} \otimes \mathbf{1} = \mathbf{1} \otimes \mathbf{g} = \mathbf{g}$
- $\mathbf{f} \otimes (\mathbf{g} \otimes \mathbf{h}) = (\mathbf{f} \otimes \mathbf{g}) \otimes \mathbf{h}$ (Associativity)
- For all \mathbf{g} , $\exists \mathbf{g}^{-1}$ s.t. $\mathbf{g} \otimes \mathbf{g}^{-1} = \mathbf{g}^{-1} \otimes \mathbf{g} = \mathbf{1}$

In this class, we will always work with finite commutative groups

- $|\mathbf{G}| < \infty$
- $\mathbf{g} \otimes \mathbf{h} = \mathbf{h} \otimes \mathbf{g}$

Examples of Groups

Additive group \mathbb{Z}_N

- $\mathbf{g \otimes h = g + h \bmod N}$

Multiplicative group \mathbb{Z}_N^*

- $\mathbf{g \otimes h = g \times h \bmod N}$

Cyclic Groups

A group \mathbf{G} of size \mathbf{N} is cyclic if:

$$\exists \mathbf{g} \text{ s.t. } \mathbf{G} = \{1, \mathbf{g}, \mathbf{g}^2, \dots, \mathbf{g}^{N-1}\}$$

(we call such a \mathbf{g} a generator)

Examples:

- Additive group \mathbb{Z}_N (generator?)
- Multiplicative group \mathbb{Z}_p^* for prime p

Non-example: \mathbb{Z}_{15}^*

Increasing Difficulty

DLog:

- Given (g, g^a) , compute a

CDH:

- Given (g, g^a, g^b) , compute g^{ab}

DDH:

- Distinguish (g, g^a, g^b, g^c) from (g, g^a, g^b, g^{ab})

Stronger Assumptions

G cyclic, order q


(G, t, ϵ) -Discrete Log:

For any algorithm  running in time at most t ,


$$\Pr[g^a \leftarrow \text{candlestick figure}(g, g^a): g \leftarrow G, a \leftarrow \mathbb{Z}_q] \leq \epsilon$$

(G, t, ϵ) -Computational Diffie Hellman:

For any algorithm  running in time at most t ,

$$\Pr[g^{ab} \leftarrow \text{}(g, g^a, g^b): g \leftarrow G, a, b \leftarrow \mathbb{Z}_q] \leq \epsilon$$

(G, t, ϵ) -Decisional Diffie Hellman:

For any algorithm  running in time at most t ,

$$\left| \Pr[1 \leftarrow \text{candlestick} (g, g^a, g^b, g^{ab}): g \leftarrow G, a, b \leftarrow \mathbb{Z}_q] \right. \\ \left. - \Pr[1 \leftarrow \text{candlestick} (g, g^a, g^b, g^c): g \leftarrow G, a, b, c \leftarrow \mathbb{Z}_q] \right| \leq \epsilon$$

Hardness of DLog

Over \mathbb{Z}_p^* :

- Brute force: $O(p)$
- Better algs based on birthday paradox: $O(p^{1/2})$
- Even better heuristic algorithms:
 $\exp(C (\log p)^{1/3} (\log \log p)^{2/3})$

- Therefore, plausible assumption:

$$(\mathbb{Z}_p^*, t=2^{(\log p)^{1/3}}, \epsilon=2^{-(\log p)^{1/3}})$$

Naor-Reingold PRF

Domain: $\{0,1\}^n$

Key space: \mathbb{Z}_q^{n+1}

Range: \mathbf{G}

$$F((a, b_1, b_2, \dots, b_n), x) = g^{a b_1^{x_1} b_2^{x_2} \dots b_n^{x_n}}$$

Theorem: If $(\mathbf{G}, t, \epsilon)$ -DDH holds, then the Naor-Reingold PRF is $(t-t', r, 2rn\epsilon)$ -secure

Proof by Hybrids

Hybrids **0**: $H(x) = g^{a \cdot b_1^{x_1} b_2^{x_2} \dots b_n^{x_n}}$

Hybrid **i**: $H(x) = H_i(x_{[1,i]}) b_{i+1}^{x_{i+1}} \dots b_n^{x_n}$

- H_i is a random function from $\{0,1\}^i \rightarrow G$

Hybrid **n**: $H(x)$ is truly random

Proof

Suppose adversary can distinguish Hybrid **$i-1$** from Hybrid **i** for some **i**

Easy to construct adversary that distinguishes:

$$\mathbf{x} \rightarrow \mathbf{H}_i(\mathbf{x}) \text{ from } \mathbf{x} \rightarrow \mathbf{H}_{i-1}(\mathbf{x}_{[1,i-1]})^{\mathbf{b}^{\mathbf{x}_i}}$$

with advantage **$2r\epsilon$**

Proof

Suppose adversary makes **$2r$** queries

- Assume wlog that queries are in pairs **$x||0, x||1$**

What does the adversary see?

- **$H_i(x)$** : **$2r$** random elements in **G**
- **$H_{i-1}(x_{[1,i-1]})^{b_i^{x_i}}$** : **$r$** random elements in **G** , **h_1, \dots, h_r**
as well as **h_1^b, \dots, h_r^b**

Lemma: Assuming (G, t, ϵ) -DDH the following distributions are indistinguishable except with advantage $s\epsilon$:

$$(g, g^{x_1}, g^{y_1}, \dots, g^{x_s}, g^{y_s}) \text{ and } (g, g^{x_1}, g^{b \cdot x_1}, \dots, g^{x_s}, g^{b \cdot x_s})$$

Suffices to finish proof of NR-PRF

Proof of Lemma

Hybrids **0**: $(g, g^{x_1}, g^{b \cdot x_1}, \dots, g^{x_s}, g^{b \cdot x_s})$

Hybrid **i**:

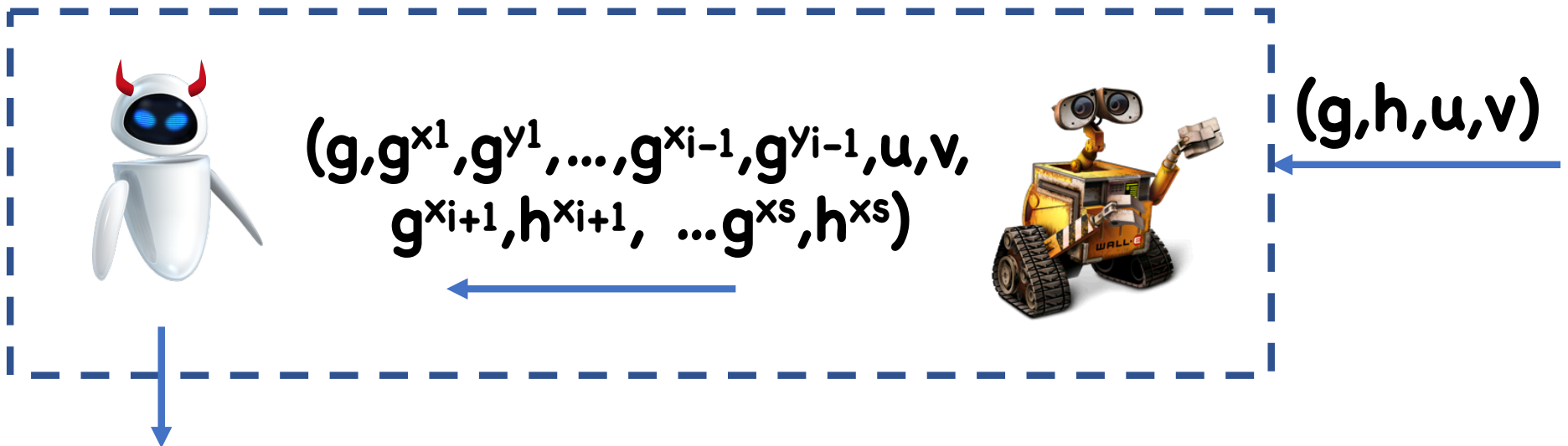
$(g, g^{x_1}, g^{y_1}, \dots, g^{x_i}, g^{y_i}, g^{x_{i+1}}, g^{b \cdot x_{i+1}}, \dots, g^{x_s}, g^{b \cdot x_s})$

Hybrid **s**: $(g, g^{x_1}, g^{y_1}, \dots, g^{x_s}, g^{y_s})$

Proof of Lemma

Suppose adversary distinguishes Hybrid **$i-1$** from Hybrid **i**

Use adversary to break DDH:



Proof of Lemma

$(g, g^{x_1}, g^{y_1}, \dots, g^{x_{i-1}}, g^{y_{i-1}}, u, v, g^{x_{i+1}}, h^{x_{i+1}}, \dots, g^{x_s}, h^{x_s})$

If $(g, h, u, v) = (g, g^b, g^{x_i}, g^{b \cdot x_i})$, then Hybrid $i-1$

If $(g, h, u, v) = (g, g^b, g^{x_i}, g^{y_i})$, then Hybrid i

Therefore, 's advantage is the same as 's

Further Applications

From NR-PRF can construct:

- CPA-secure encryption
- Block Ciphers
- MACs
- Authenticated Encryption

Parameter Size in Practice?

G = subgroup of \mathbb{Z}_p^* of order **q** , where **$q \mid p-1$**

- In practice, best algorithms require **$p \geq 2^{1024}$** or so

- **G** = "elliptic curve groups"
- Can set **$p \approx 2^{256}$** to have security
 \Rightarrow best attacks run in time **2^{128}**

Therefore, elliptic curve groups tend to be much more efficient \Rightarrow shift to using in practice


Integer Factorization


Integer Factorization

Given an integer **N**, find it's prime factors

Studied for centuries, presumed difficult

- Grade school algorithm: $O(N^{1/2})$
- Better algorithms using birthday paradox: $O(N^{1/4})$
- Even better assuming G. Riemann Hyp.: $O(N^{1/5})$
- Still better heuristic algorithms:
 $\exp(C (\log N)^{1/3} (\log \log N)^{2/3})$
- However, all require super-polynomial time in bit-length of **N**

(λ, t, ϵ) -Factoring Assumption: For any factoring algorithm  running in time at most t ,

$\Pr[(p, q) \leftarrow \text{}(N):$
 $N = pq$ and p, q random λ -bit primes] $\leq \epsilon$

Plausible assumption: $(\lambda, t = 2^{\lambda^{1/3}}, \epsilon = 2^{-\lambda^{1/3}})$

Sampling Random Primes

Prime Number Theorem: A random λ -bit number is prime with probability $\approx 1/\lambda$

Primality Testing: It is possible in polynomial time to decide if an integer is prime

Fermat Primality Test (randomized, some false positives):

- Choose a random integer $a \in \{0, \dots, N-1\}$
- Test if $a^N = a \bmod N$
- Repeat many times

Chinese Remainder Theorem

Let $N = pq$ for distinct prime p, q

Let $x \in \mathbb{Z}_p$, $y \in \mathbb{Z}_q$

Then there exists a unique integer $z \in \mathbb{Z}_N$ such that

- $x = z \bmod p$, and
- $y = z \bmod q$


Proof: $z = [py(p^{-1} \bmod q) + qx(q^{-1} \bmod p)] \bmod N$

Quadratic Residues

Definition: y is a quadratic residue mod N if there exists an x such that $y = x^2 \bmod N$. x is called a “square root” of y

Ex:

- Let p be a prime, and $y \neq 0$ a quadratic residue mod p . How many square roots of y ?
- Let $N=pq$ be the product of two primes, y a quadratic residue mod N . Suppose $y \neq 0 \bmod p$ and $y \neq 0 \bmod q$. How many square roots?

(λ, t, ϵ) -QR Assumption: For any factoring algorithm  running in time at most t ,

$\Pr[y^2 = x^2 \bmod N]:$

$y \leftarrow \text{fact}(N, x^2)$

$N = pq$ and p, q random λ -bit primes

$x \leftarrow \mathbb{Z}_N \quad] \leq \epsilon$

Theorem: If the (λ, t, ϵ) -factoring assumption holds, then the $(\lambda, t - t', 2\epsilon)$ -QR assumption holds

Proof

To factor **N**:

- $x \leftarrow \mathbb{Z}_N$
- $y \leftarrow \text{👤}(N, x^2)$
- Output **GCD(x-y, N)**

Analysis:

- Let $\{a, b, c, d\}$ be the 4 square roots of x^2
- 👤 has no idea which one you chose
- With probability $\frac{1}{2}$, **y** will not be in $\{+x, -x\}$
- In this case, we know **$x = y \pmod p$** but **$x = -y \pmod q$**

Collision Resistance from Factoring

Let $N=pq$, y a QR mod N

Suppose -1 is not a QR mod N

Hashing key: (N,y)

Domain: $\{1,\dots,(N-1)/2\} \times \{0,1\}$

Range: $\{1,\dots,(N-1)/2\}$

$H((N,y), (x,b))$: Let $z = y^b x^2 \bmod N$

- If $z \in \{1,\dots,(N-1)/2\}$, output z
- Else, output $-z \bmod N \in \{1,\dots,(N-1)/2\}$

Theorem: If the $(\lambda, t, \varepsilon)$ -factoring assumption holds, H is $(t - t', 2\varepsilon)$ -collision resistant

Proof:

- Collision means $(x_0, b_0) \neq (x_1, b_1)$ s.t.
$$y^{b_0} x_0^2 = \pm y^{b_1} x_1^2 \pmod{N}$$
- If $b_0 = b_1$, then $x_0 \neq x_1$, but $x_0^2 = \pm x_1^2 \pmod{N}$
 - $x_0^2 = -x_1^2 \pmod{N}$ not possible. Why?
 - $x_0 \neq -x_1$ since $x_0, x_1 \in \{1, \dots, (N-1)/2\}$
- If $b_0 \neq b_1$, then $(x_0/x_1)^2 = \pm y^{\pm 1} \pmod{N}$
 - $-y$ case not possible. Why?
 - (x_0/x_1) or (x_1/x_0) is a square root of y

Choosing **N**

How to choose **N** so that **-1** is not a QR?

By CRT, need to choose **p,q** such that **-1** is not a QR mod **p** or mod **q**

Fact: if **p = 3 mod 4**, then **-1** is not a QR mod **p**

Fact: if **p = 1 mod 4**, then **-1** is a QR mod **p**

Is Composite **N** Necessary for SQ to be hard?

Let **p** be a prime, and suppose **$p = 3 \bmod 4$**

Given a QR **$x \bmod p$** , how to compute square root?

Hint: recall Fermat: **$x^{p-1} = 1 \bmod p$** for all **$x \neq 0$**

Hint: what is **$x^{(p+1)/2} \bmod p$** ?

Solving Quadratic Equations

In general, solving quadratic equations is:

- Easy over prime moduli
- As hard as factoring over composite moduli

Other Powers?

What about $x \rightarrow x^4 \bmod N$? $x \rightarrow x^6 \bmod N$?

The function $x \rightarrow x^3 \bmod N$ appears quite different

- Suppose **3** is relatively prime to **p-1** and **q-1**
- Then $x \rightarrow x^3 \bmod p$ is injective for $x \neq 0$
 - Let **a** be such that $3a = 1 \bmod p-1$
 - $(x^3)^a = x^{1+k(p-1)} = x(x^{p-1})^k = x \bmod p$
- By CRT, $x \rightarrow x^3 \bmod N$ is injective for $x \in \mathbb{Z}_N^*$

$x^3 \bmod N$

What does injectivity mean?

Cannot base of factoring:

Adapt alg for square roots:

- Choose a random $z \bmod N$
- Compute $y = z^3 \bmod N$
- Run inverter on y to get a cube root x
- Let $p = \text{GCD}(z-x, N)$, $q = N/p$

RSA Problem

Given

- **$N = pq$,**
- **e such that $\text{GCD}(e, p-1) = \text{GCD}(e, q-1) = 1$,**
- **$y = x^e \bmod N$ for a random x**

Find **x**

Injectivity means cannot base hardness on factoring,
but still conjectured to be hard

(e,t,ε)-RSA Assumption: For any factoring algorithm  running in time at most t ,

$\Pr[x \leftarrow \text{ wizard } (N, x^3 \bmod N)$

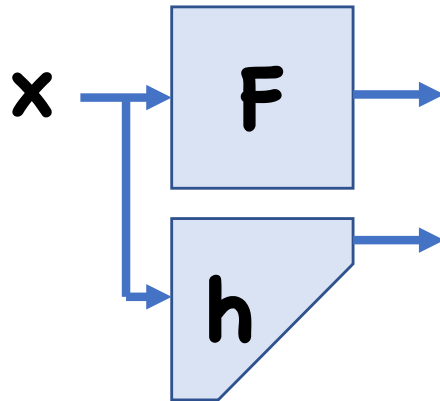
$N=pq$ and p,q random λ -bit primes s.t.

$\text{GCD}(3,p-1)=\text{GCD}(3,q-1)=1$

$x \leftarrow \mathbb{Z}_N^*] \leq \epsilon$

Application: PRGs

Let $F(x) = x^3 \bmod N$, $h(x)$ = least significant bit



Theorem: If (e,t,ϵ) -RSA Assumption holds, then $G(x) = (F(x), h(x))$ is a $(t-t',\epsilon')$ -secure PRG

Crypto from Minimal Assumptions

Many ways to build crypto

We've seen many ways to build crypto

- SPN networks
- LFSR's
- Discrete Log
- Factoring

Questions:

- Can common techniques be abstracted out as theorem statements?
- Can every technique be used to build every application?

One-way Functions


The minimal assumption for crypto

Syntax:

- Domain **\mathcal{D}**
- Range **\mathcal{R}**
- Function **$F: \mathcal{D} \rightarrow \mathcal{R}$**

No correctness properties other than deterministic

Security?

Definition: F is (t, ϵ) -One-Way if, for all  running in time at most t ,


$$\Pr[x \leftarrow \text{pirate}(F(x)), x \leftarrow D] < \epsilon$$

Trivial example:

$F(x)$ = parity of x

Given $F(x)$, impossible to predict x

Security

Definition: F is (t, ϵ) -One-Way if, for all  running in time at most t ,

$$\Pr[F(x) = F(y) : y \leftarrow \text{pirate}(F(x)), x \leftarrow D] < \epsilon$$

Examples

Any PRG

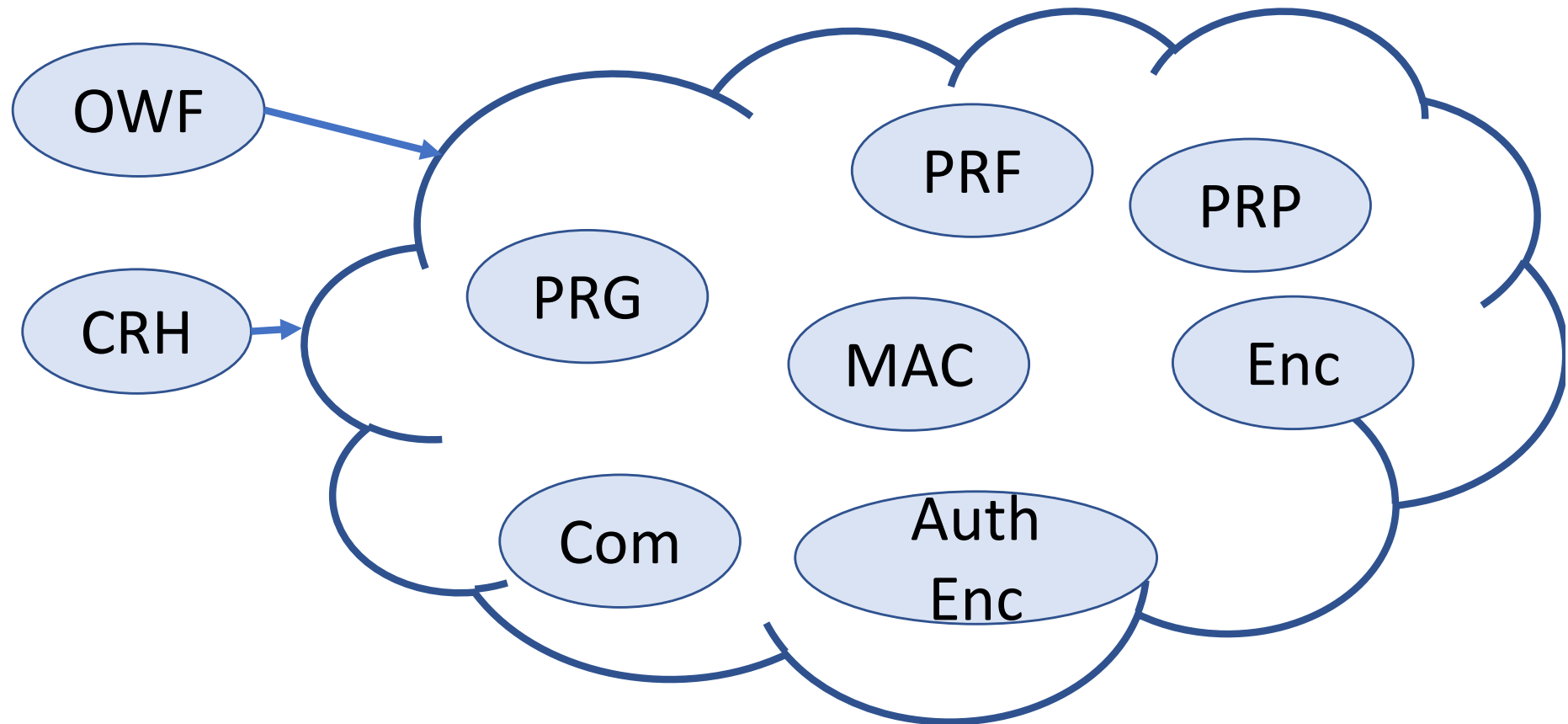
Any Collision Resistant Hash Function (with sufficient compression)

$$F(p,q) = pq$$

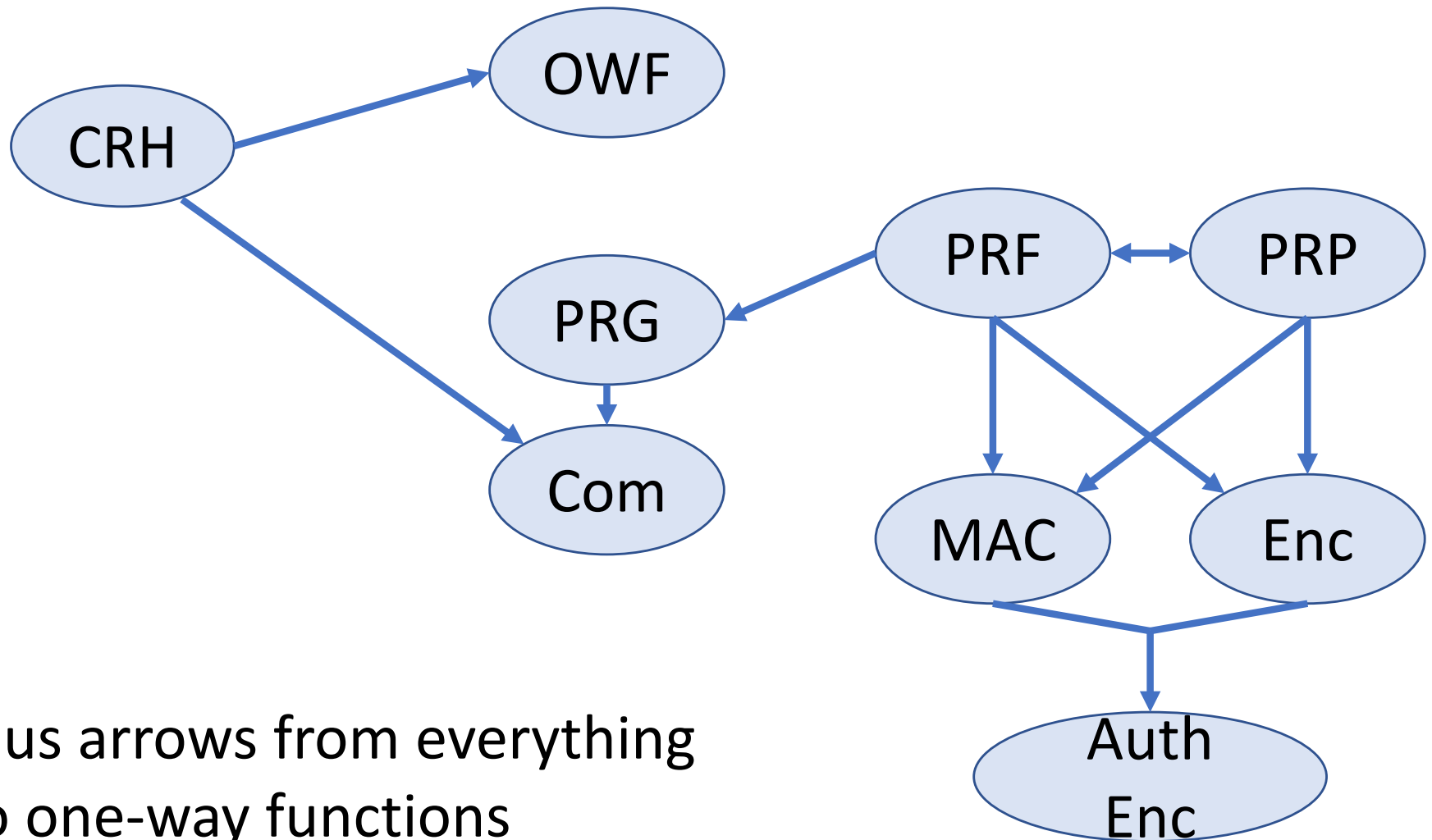
$$F(g,a) = (g,g^a)$$

$$F(N,x) = (N,x^3 \bmod N) \text{ or } F(N,x) = (N,x^2 \bmod N)$$

What's Known




So Far



Our Goal: Fill in Remaining Arrows

Hardcore Bits

Let \mathbf{F} be a one-way function with domain \mathbf{D} , range \mathbf{R}

Definition: A function $\mathbf{h}:\mathbf{D}\rightarrow\{0,1\}$ is a (t,ϵ) -hardcore bit for \mathbf{F} if, for any  running in time at most t ,

$$| \Pr[1 \leftarrow \text{robot}(F(x), h(x)), x \leftarrow D]$$

$$- \Pr[1 \leftarrow \text{robot}(F(x), b), x \leftarrow D, b \leftarrow \{0,1\}] | \leq \epsilon$$

In other words, even given $\mathbf{F}(\mathbf{x})$, hard to guess $\mathbf{h}(\mathbf{x})$

Examples of Hardcore Bits

Define **lsb(x)** as the least significant bit of **x**

For **x** $\in \mathbf{Z}_N$, define **Half(x)** as **1** iff **0** $\leq x < N/2$

Theorem: Let p be a prime, and $F: \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$ be $F(x) = g^x \bmod p$, for some generator g

Half is a hardcore bit for F (assume F is one-way)

Theorem: Let N be a product of two large primes p, q , and $F: \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ be $F(x) = x^e \bmod N$ for some e relatively prime to $(p-1)(q-1)$

Lsb and Half are hardcore bits for F (assuming RSA)

Theorem: Let N be a product of two large primes p, q , and $F: \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ be $F(x) = x^2 \bmod N$

Lsb and Half are hardcore bits for F (assuming factoring)

Goldreich Levin Hardcore Bit

Let \mathbf{F} be a OWF with domain $\{0,1\}^n$ and range \mathbf{R}

Let $\mathbf{F}':\{0,1\}^{2n} \rightarrow \{0,1\}^n \times \mathbf{R}$ be:

$$\mathbf{F}'(r,x) = r, \mathbf{F}(x)$$

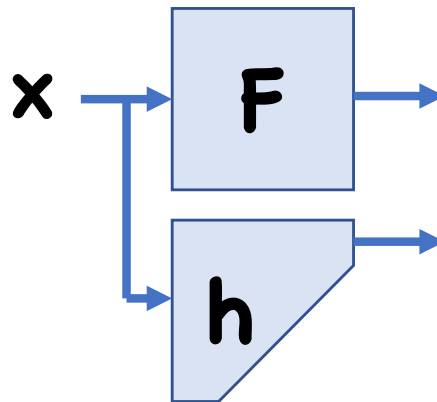
Define $\mathbf{h}(r,x) = \langle r,x \rangle = \sum r_i x_i \bmod 2$

Theorem (Goldreich-Levin): If \mathbf{F} is (t,ϵ) -one-way, then \mathbf{h} is a $(\text{poly}(t,1/\epsilon), \text{poly}(\epsilon))$ -hc bit for \mathbf{F}'

Application: PRGs

Suppose \mathbf{F} was a permutation ($\mathbf{D}=\mathbf{R}$ and \mathbf{F} is one-to-one)

Let \mathbf{F}' , \mathbf{h} be from Goldreich-Levin



Hardcore Bits

A hc bit for any OWF

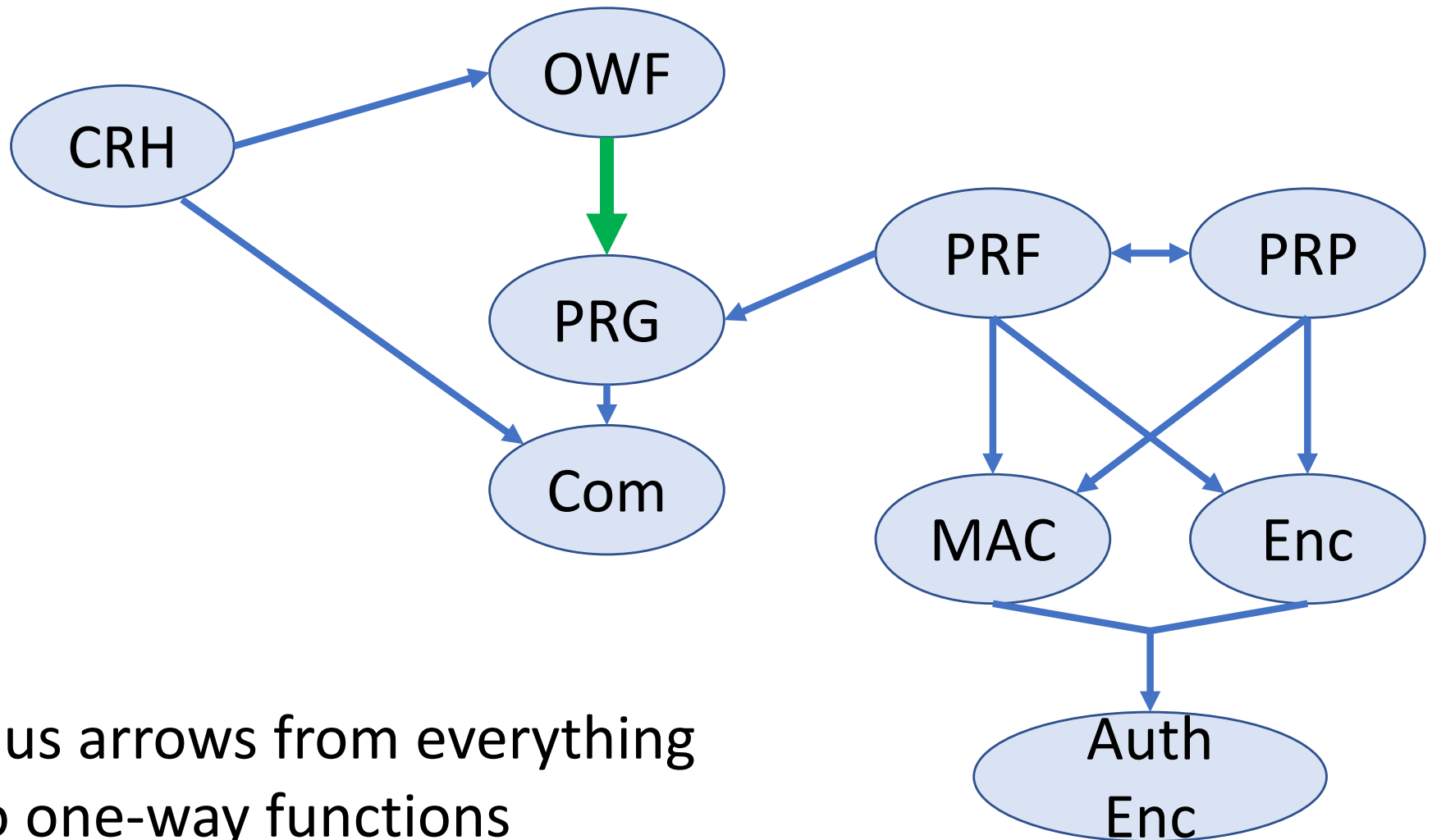
Implies PRG from any one-way *permutation*

- PRG from Dlog (Blum-Micali)
- PRG from RSA
- PRG from Factoring

Actually, can construct PRG from any OWF

- Proof beyond scope of course

So Far



Reminders

HW5 due next week

Keep working on project