

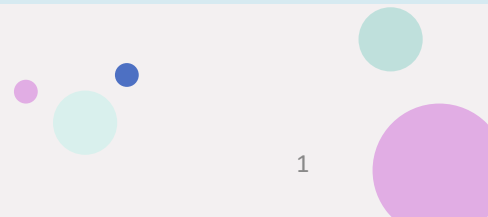


Lecture 14

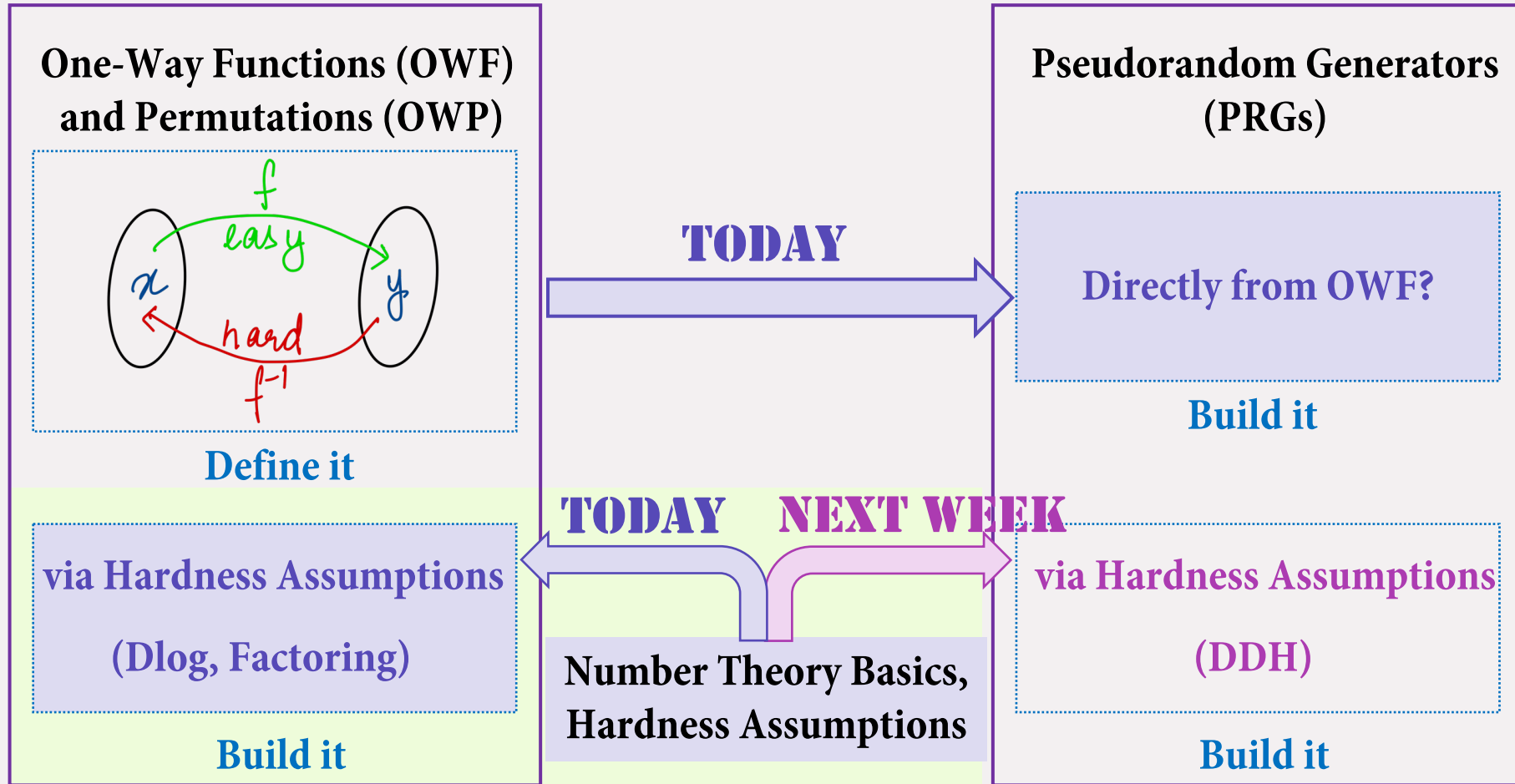
One Way Functions, Hard-core Predicates, Goldreich-Levin

CS409/6011: Introduction to Cryptography

Sruthi Sekar



Coming Up..



Coming Up..

Number Theory Basics, Hardness Assumptions

```
graph TD; A[Number Theory Basics, Hardness Assumptions] -.-> B[Groups]; A -.-> C[Cryptographic Hardness Assumptions]; A -.-> D[Computations on Groups];
```

Groups

- ✓ Definition and Properties
- ✓ Examples

Cryptographic Hardness Assumptions

- Discrete Log Assumption
- Factoring Assumption

Computations on Groups

- ✓ Efficient computation on groups
- ✓ Primality testing, generating random primes and random group elements
- Chinese Remainder Theorem

Chinese Remainder Theorem

Group Isomorphism denoted by $\mathbb{G} \equiv \mathbb{H}$

There exists a **bijective map** $\phi: \mathbb{G} \rightarrow \mathbb{H}$ such that $\forall a, b \in \mathbb{G}, \phi(a *_{\mathbb{G}} b) = \phi(a) *_{\mathbb{H}} \phi(b)$

$\mathbb{G} \times \mathbb{H}$ consists of all ordered pairs $(g, h), g \in \mathbb{G}, h \in \mathbb{H}$ with the operation: $(g, h) * (g', h') := (g *_{\mathbb{G}} g', h *_{\mathbb{H}} h')$

Chinese Remainder Theorem (CRT):

Let $N = pq$ where $p, q > 1$ are relatively prime (i.e., $\gcd(p, q) = 1$), then

(additive group) $\mathbb{Z}_N \equiv \mathbb{Z}_p \times \mathbb{Z}_q$ and $\mathbb{Z}_N^* \equiv \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ (multiplicative group)

With the same bijection map

$$\phi(x) := ([x \bmod p], [x \bmod q])$$

In the birthday demo we started with: we needed to efficiently find the inverse of this map.

$$\phi^{-1}((x_1, x_2)) := (c_1 x_1 + c_2 x_2) \bmod N, \text{ where } c_1 = q \cdot (q^{-1} \bmod p), c_2 = p \cdot (p^{-1} \bmod q)$$

**** ϕ^{-1} or ϕ usually need not be efficiently computable, CRT is an example where both are!**

Chinese Remainder Theorem

Where does isomorphism help?

Given $\mathbb{G} \equiv \mathbb{H}$ with $\cdot$, one can choose to do group computations in the group where it is more efficient!

EXAMPLES

$N = pq$ for distinct primes p and q

CRT tells you that addition, multiplication and group exponentiation **mod N** can be **transformed to analogous operations mod p and mod q** .

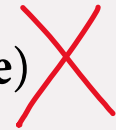
1. **Want to compute:** $14 \times 13 \bmod 15$ (in $\mathbb{Z}_{15}^*$) $\phi_{\text{CRT}}(14) = (14 \bmod 5, 14 \bmod 3) = (4, 2)$
 $\phi_{\text{CRT}}(13) = (13 \bmod 5, 13 \bmod 3) = (3, 1)$

$$(4, 2) * (3, 1) = ((4 \times 3 \bmod 5), (2 \times 1 \bmod 3)) = (2, 2) \quad \phi_{\text{CRT}}^{-1}((2, 2)) = 2 = 14 \times 13 \bmod 15$$

2. **Want to compute:** $29^{100} \bmod 35$ $\phi_{\text{CRT}}(29) = (29 \bmod 5, 29 \bmod 7) = (-1 \bmod 5, 1)$

$$(-1 \bmod 5, 1)^{100} = ((-1)^{100} \bmod 5, 1) = (1, 1) \quad \phi_{\text{CRT}}^{-1}((1, 1)) = 1 = 29^{100} \bmod 35$$

One Way Function Family

$f_{a,p}(x) := ax \bmod p$, where $a, x \in \mathbb{Z}_p^*$ and p is a prime. (Multiplication mod prime) 

$f_p(x) := x^2 \bmod p$, where $x \in \mathbb{Z}_p^*$ and p is a prime. (Squaring mod prime) 

$f_{p,g}(x) := g^x \bmod p$, where $x \in \mathbb{Z}_{p-1}$, $\mathbb{Z}_p^* = \langle g \rangle$, p is a prime. (Exponentiation mod prime) 

(If “*Dlog Assumption*” for $\mathbb{Z}_p^*$ then $\{f_{p,g}\}$ is a OWF family)

Discrete Log Assumption (intuitively)

The exponentiation function

$$f_g: \mathbb{Z}_q \rightarrow \mathbb{G}: f_g(x) = g^x$$

It's a group isomorphism.

 inverse

Discrete Log with respect to g

$$\text{Dlog}_g(h) := \text{unique } x \in \mathbb{Z}_q \text{ such that } h = g^x, \text{ for each } h \in \mathbb{G}$$

Cyclic Group $(\mathbb{G}, q, g)$

Group Order Generator

 Is f_g invertible efficiently?

In “some groups”, for random $h \in_R \mathbb{G}$ it is “hard” to find $\text{Dlog}_g(h)$

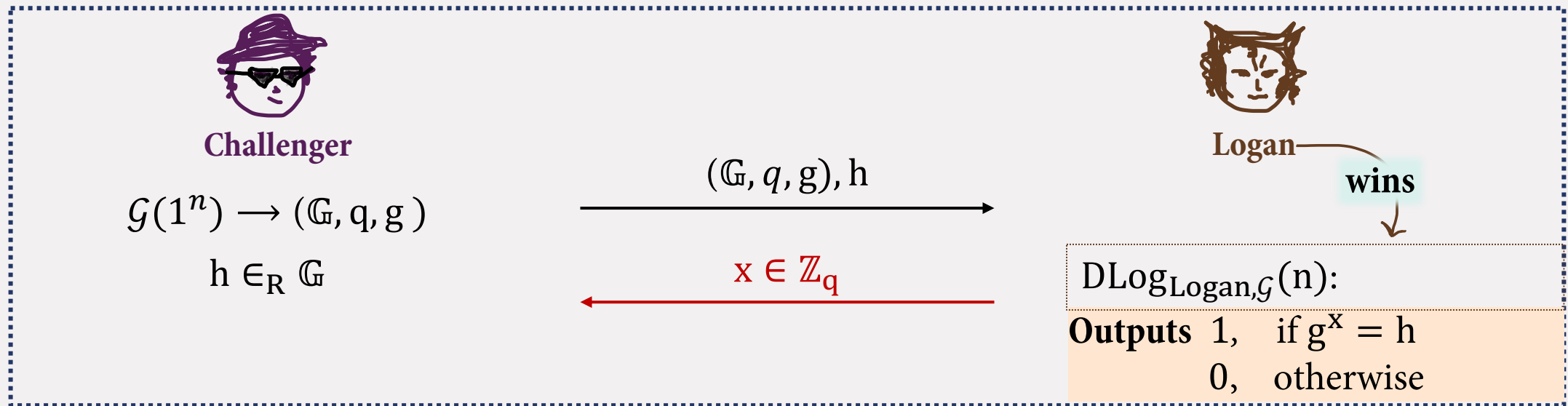
Example: Is it hard in this group?

$(\mathbb{Z}_N, + \bmod N)$

$(\mathbb{Z}_p^*, \times \bmod p)$

Discrete Log Assumption

Some generic, polytime **group generation algorithm** $\mathcal{G}(1^n) \rightarrow (\mathbb{G}, q, g)$, where $\|q\| = n$ and the description of group specifies how to represent each element as a unique bit-string, such that one can compute efficiently on the elements of $\mathbb{G}$.



Discrete Log Assumption

The **discrete log problem is hard relative to $\mathcal{G}$** if for all **PPT Logan**, there exists a negligible negl such that, $\forall n$

$$\Pr[\text{Dlog}_{\text{Logan}, \mathcal{G}}(n) = 1] \leq \text{negl}(n)$$

Discrete Log Assumption

Why do we believe this is a hard assumption?

Best known algorithm to find discrete log in an **arbitrary group** $\mathbb{G}$ runs in **time** $O(\sqrt{|\mathbb{G}|})$
(Baby-step giant-step algorithm)

Can we do better?

- For $\mathbb{Z}_p^*$, there are better algorithms:

Index Calculus Algorithm ($2^{O(\sqrt{\log y \log \log y})}$),

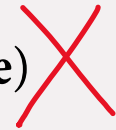
Number Field Sieve Algorithm ($2^{O((\log y)^{1/3} (\log \log y)^{2/3})}$), where $y \sim \sqrt{|\mathbb{G}|}$

- For finite abelian group with “smooth integer” order:

Pohlig-Hellman Algorithm (if $|\mathbb{G}| = N = \prod_i p_i^{\alpha_i}$, runs in time $O(\sum_i \alpha_i (\log N + \sqrt{p_i}))$)

Moral of the story: Yes, you can do better than the naïve algorithm, but still the time complexity is exponential for arbitrary groups, and sub-exponential for certain specific groups.

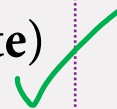
One Way Function Family

$f_{a,p}(x) := ax \bmod p$, where $a, x \in \mathbb{Z}_p^*$ and p is a prime. (Multiplication mod prime) 

$f_p(x) := x^2 \bmod p$, where $x \in \mathbb{Z}_p^*$ and p is a prime. (Squaring mod prime) 

$f_{p,g}(x) := g^x \bmod p$, where $x \in \mathbb{Z}_{p-1}$, $\mathbb{Z}_p^* = \langle g \rangle$, p is a prime. (Exponentiation mod prime) 

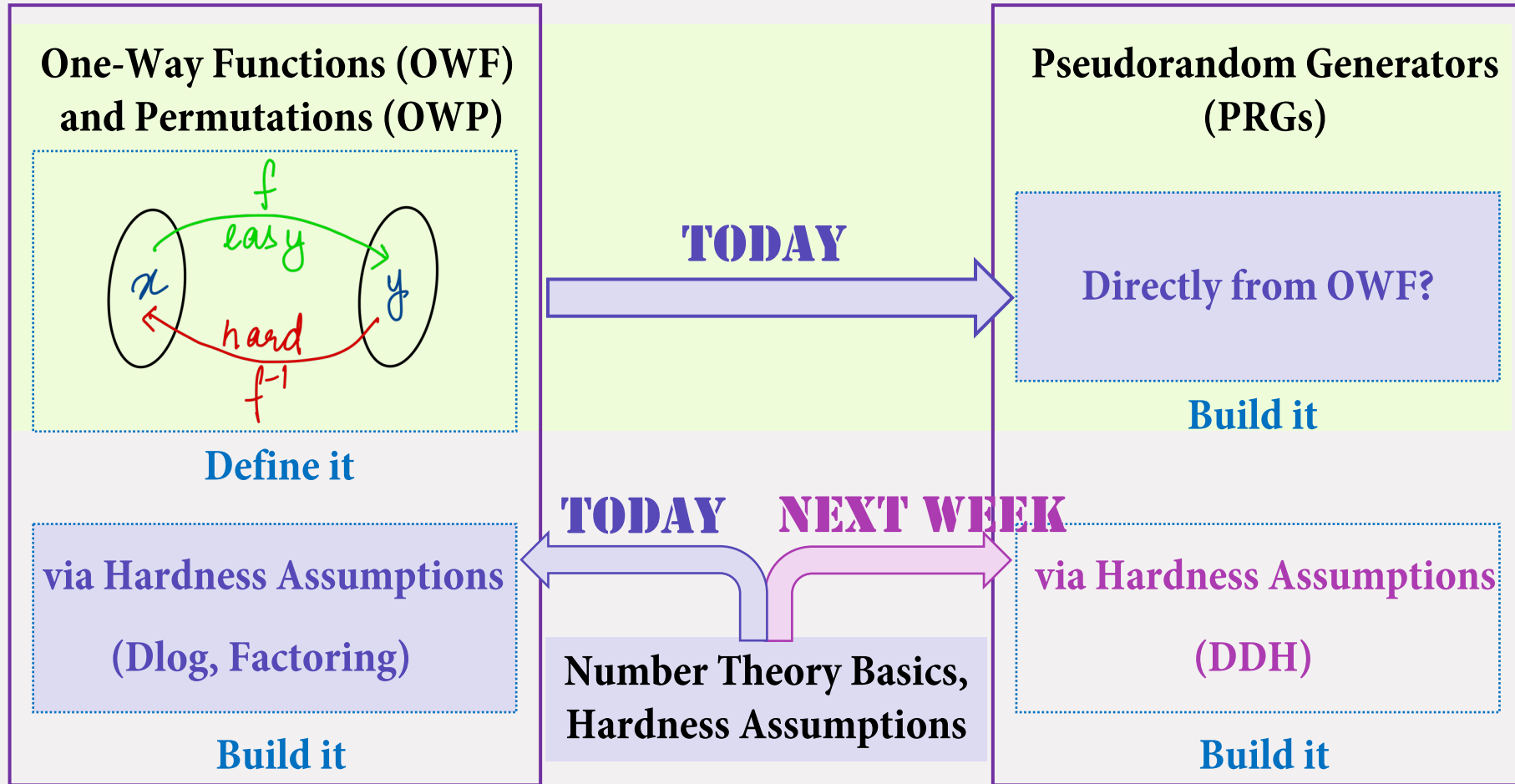
(If Dlog Assumption holds (in $\mathbb{Z}_p^*$ it is believed to hold) then $\{f_{p,g}\}$ is a OWF family)

$f_N(x) := x^2 \bmod N$, where $x \in \mathbb{Z}_N^*$ and $N = pq$ (large primes p and q). (Squaring mod composite) 

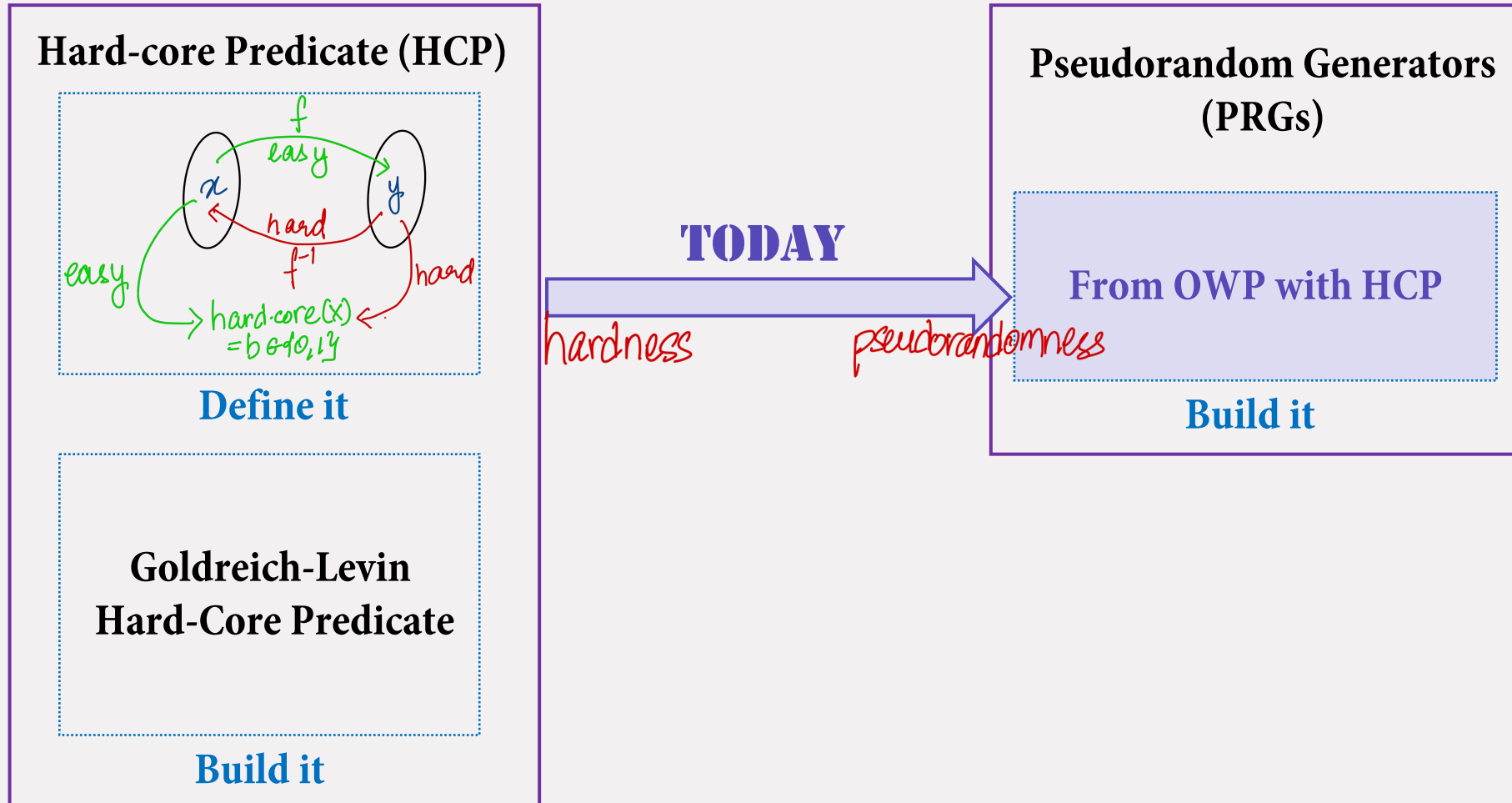
(If factoring N is hard (factoring assumption), then $\{f_N\}$ is a OWF family)

HW 1: Show that taking square root mod N
is equivalent to factoring N
(Hint: $x^2 - y^2 = (x + y)(x - y) \bmod N$), use CRT!)

Coming Up..



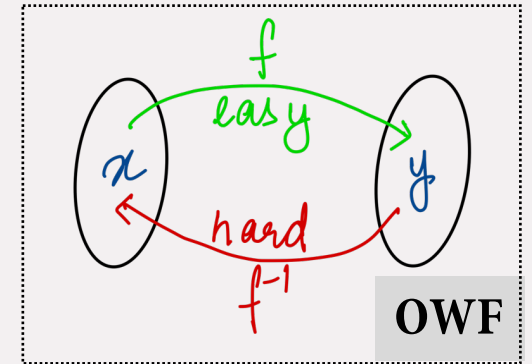
Coming Up..



Hard-Core Bit

One-way function/permutation

Alternate view: x cannot be computed **in its entirety** by any efficient algorithm (except with negligible probability)



Is it possible that f leaks a lot of information about x , even if it is one way?

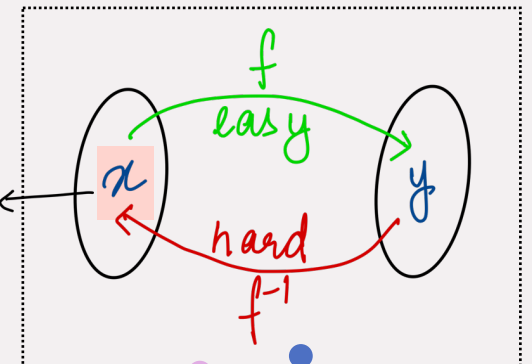
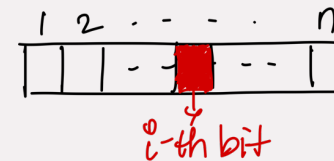
Yes! (f is a OWF (prove it) but reveals half its input!)

$f(x_1 || x_2) := x_1 || g(x_2)$, where g is a OWF and $|x_1| = |x_2|$

In general, can we identify what about x is hidden by a OWF f ?

ATTEMPT I

i -th bit is hardcore if
 x_i is hard to predict, given $f(x)$

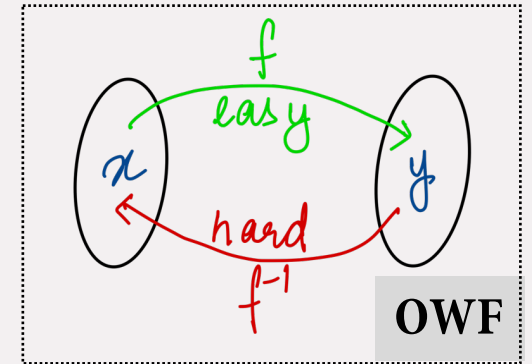


(HW 2) Problem: $\exists$ OWFs for which every input bit is “easy” to predict!

Hard-Core Predicates (HCP)

One-way function/permutation

Alternate view: x cannot be computed **in its entirety** by any efficient algorithm (except with negligible probability)



Is it possible that f leaks a lot of information about x , even if it is one way? **Yes!**

In general, can we identify what about x is hidden by a OWF f ?

Hard-core predicate

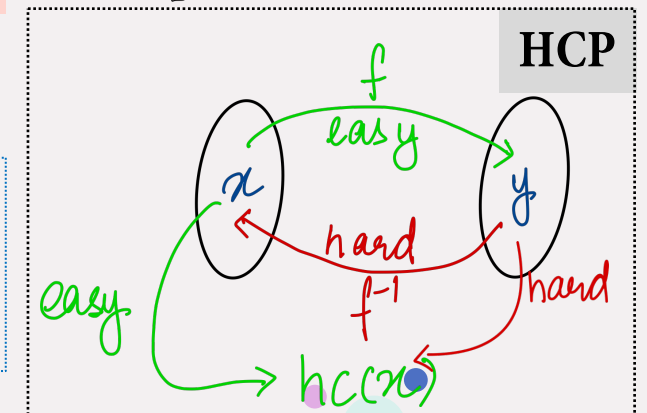
$hc: \{0,1\}^* \rightarrow \{0,1\}$ of a function f is such that:

$hc(x)$ easy to compute given x

$hc(x)$ is hard to compute given $f(x)$



INTUITIVELY



Hard-Core Predicates (HCP)

Definition 2 (Hard-core predicate of a function family)

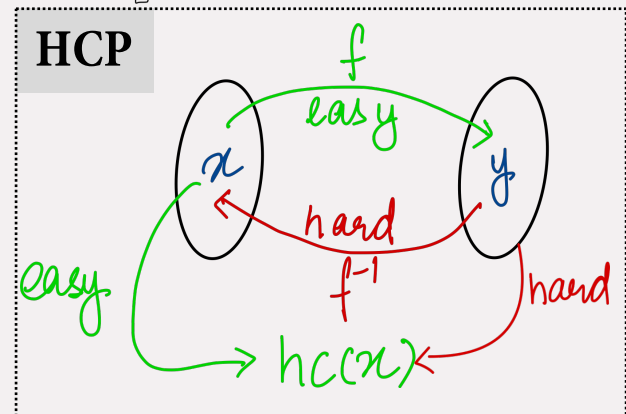
$$\{f_n: \{0,1\}^n \rightarrow \{0,1\}^{m(n)}\}_{n \in \mathbb{N}}$$

A predicate $hc: \{0,1\}^n \rightarrow \{0,1\}$ is a **hard-core predicate** for the above function family if for every **PPT** predictor **Pred**, $\exists \text{negl}$ such that $\forall n$

$$\Pr_{x \in_R \{0,1\}^n} [\mathbf{Pred}(1^n, f_n(x)) = hc(x)] \leq \frac{1}{2} + \text{negl}(n)$$



INTUITIVELY



SOME IDEAS FOR HCP:

$hc(x) \stackrel{\text{def}}{=} \bigoplus_{i=1}^n x_i$ Is it hard-core for every one-way function?

No! Consider $f(x) \stackrel{\text{def}}{=} (g(x), \bigoplus_{i=1}^n x_i)$, where g is a OWF



In fact, for any fixed hc , $\exists$ OWF f for which hc is not an HCP of f

Trivial HCP: Consider $f(x) = 0^n$ then x_i is hard-core $\forall i$
But, f is not a OWF!

This kind of HCP from “information loss” is useless for us

Goldreich-Levin HCP

Given any one-way function, can we find a deterministic hard-core predicate for it?

We don't know, it remains open!

Something
weaker

Goldreich-Levin Theorem

For every one-way function f , $\exists$ a related one-way function $g(x, r) \stackrel{\text{def}}{=} (f(x), r)$ for $|x| = |r| = n$ with a deterministic hard-core predicate $hc: \{0,1\}^{2n} \rightarrow \{0,1\}$ defined as

$$hc(x, r) = \bigoplus_{i=1}^n x_i \cdot r_i$$

****If interested in proof, refer to Katz and Lindell! We will skip it in this course!**

OWP $\Rightarrow$ PRG

Theorem: OWP with HCP gives a PRG

Let $f: \{0,1\}^n \rightarrow \{0,1\}^n$ be a one-way permutation, and let hc be a hard-core predicate of f .
Then, $G(s) \stackrel{\text{def}}{=} f(s) || hc(s)$ is a pseudorandom generator with expansion factor $n + 1$

Proof Sketch: $\exists \text{Pred}^G$ against $G \Rightarrow \exists \text{Pred}^{hc}$ against the hard-core predicate hc of f

$s \in_R \{0,1\}^n$

$f(s) = y_1 || y_2 || \dots || y_n$



Pred^{hc}

Pred^G

i

$y_1 y_2 \dots y_i$

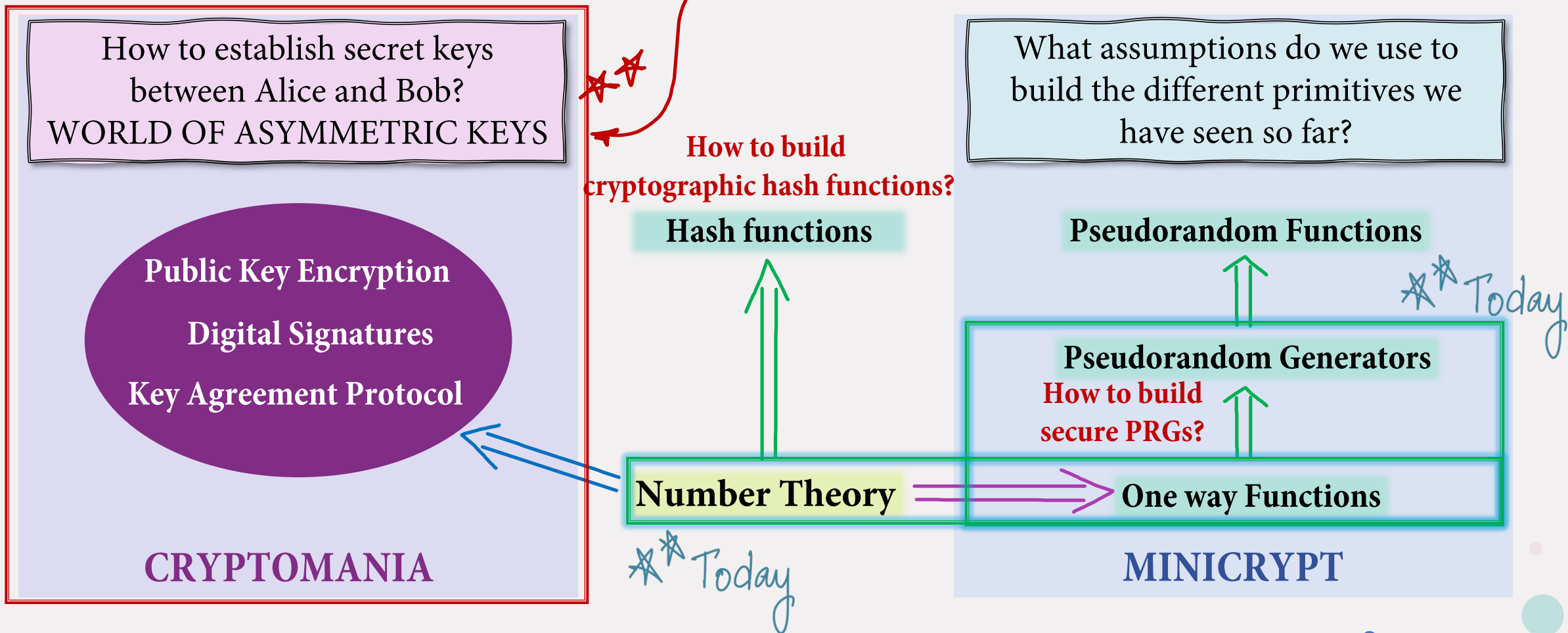
b

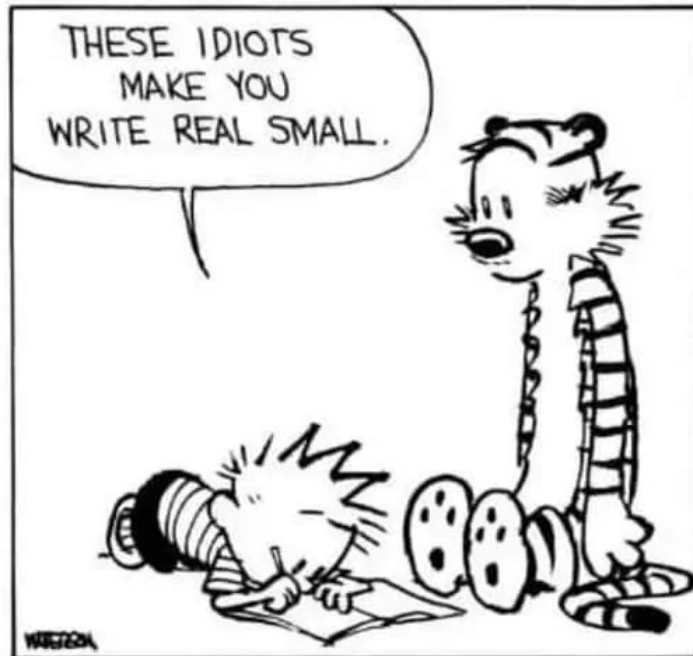
Predicts y_{i+1} w.p.
 $> 1/2 + \text{non-negl}$

1. Pred^G is useful only if $i = n$.
2. Pred^G has no advantage in predicting $y_1 || y_2 \dots || y_n$ (f permutation $\Rightarrow$ output random)

****OWF $\Rightarrow$ PRG, but the proof is harder!**

Next Week





THANK
YOU