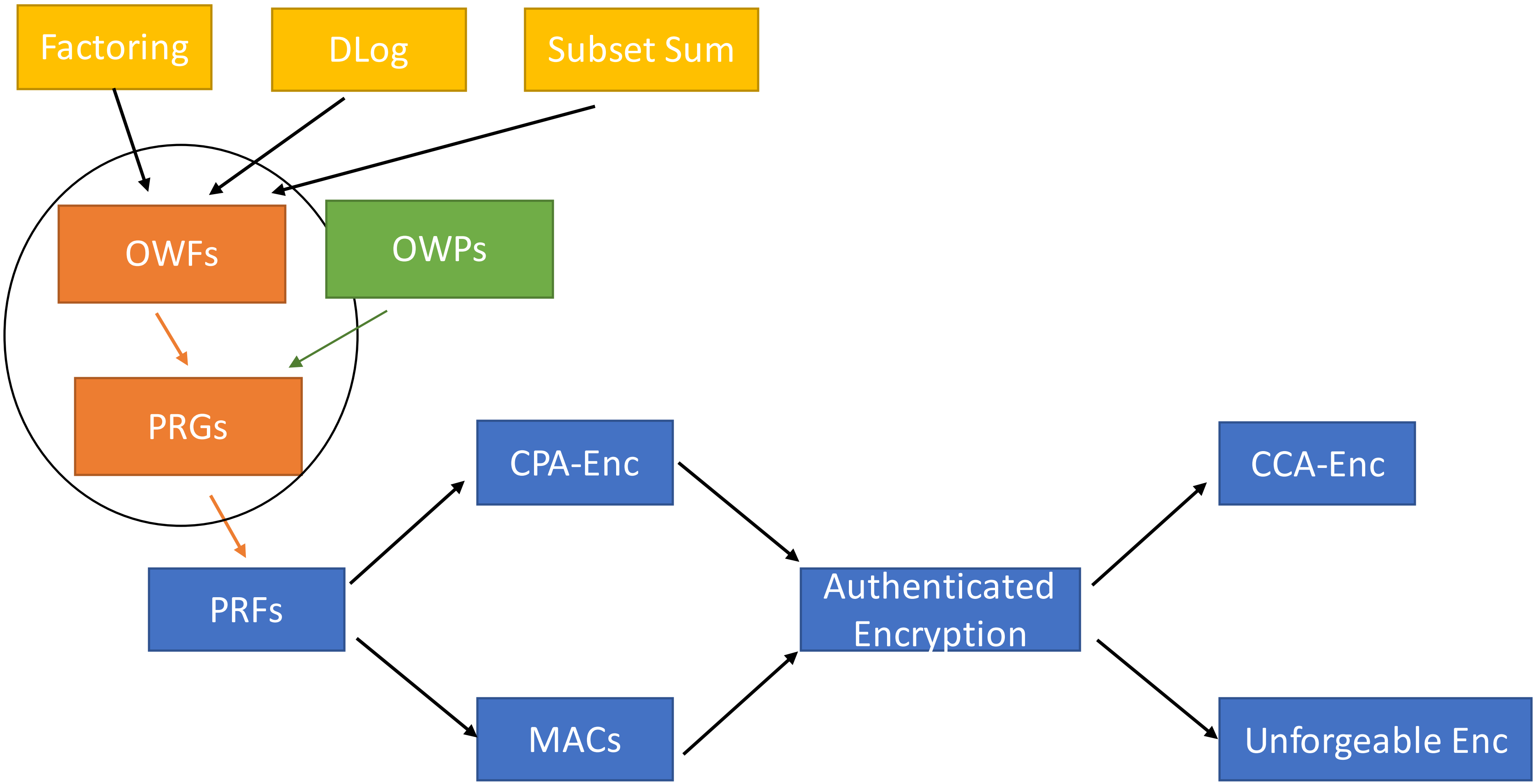
CS171: Cryptography

Lecture 12

Sanjam Garg

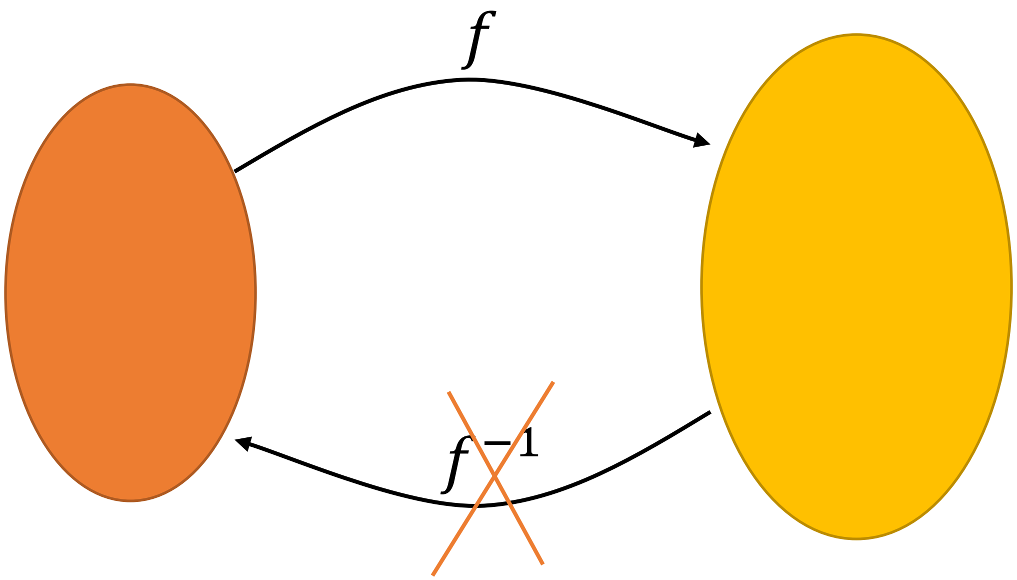
All Photos are by Unknown Author and are licensed under CC BY-SA

Theoretical Landscape



Define: One-Way Functions

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



One-Way Functions: Formally

- A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a one-way function if:
- (easy to compute) There exists a polynomial-time algorithm M_f computing f; i.e., for all x, $M_f(\mathrm{x}) = f(x)$.
- (hard to invert) For all PPT A, there is a negligible function $negl$ such that

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A\left(1^n, f(x)\right) \in f^{-1}\left(f(x)\right)\right] \le negl(n)$$

Is g a OWF?

- If f is a OWF and

$$g(x) = f(x)_1 \dots f(x)_{|f(x)|-1}$$

- Yes, for every OWF f we have that g is OWF.

- No, there exists OWF f such that g is not OWF.
  - Assume a OWF h and use that to construct f
  - Prove f is a OWF
  - Prove g is not a OWF

Is g a OWF?

- If f is a OWF and

$$g(x) = f(x)_1 \dots f(x)_{|f(x)|-1}$$

- No! Let h be a OWF $\{0,1\}^{n/2} \to \{0,1\}^{n/2}$ and let

$$f(x,y) = \begin{cases} h(x)|0^{\frac{n}{2}} & \text{if } y \neq 0^{n/2} \\ x\left|0^{\frac{n}{2}-1}\right|1 & \text{if } y = 0^{n/2} \end{cases}$$

The challenge: Is g a OWF?

- If f is a OWF and

$$g(x, y) = f(x)||y$$

- Yes!
- Lesson: The output of a OWF could include many bits from the input.
- Also, g is a OWP if f is a OWP!

We don’t know where the hardness is?

Hardness Concentration

Hard-Concentrate Bits

Concentrate the hardness of the OWP into one bit!

- A function $hc\colon \{0,1\}^n \to \{0,1\}$ is a hard-concentrate predicate of a permutation f if:

1. hc can be computed in polynomial time.
2. $\forall$ $PPT\ A$, $\exists$ a negligible function $\mathrm{neg}(\cdot)$ such that

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A\left(1^n, f(x)\right) = hc(x)\right] \leq \frac{1}{2} + neg(n)$$

Given $f(x)$, $hc(x)$ is fixed.

This would concentrate the hardness of the OWF f into one bit.

Can we take the xor or all input bits?

- Given OWF g, use $hc(x) = \sum_i x_i \; mod \; 2$

- Bad Idea!
- $g(x) = f(x_{-1}), hc(x)$

Practice Problem

- Construct a OWF g for which no input bit is hard-concentrate! (assume the existence of a OWF f)

$$g(x, i) = f(x_{-i}), i, x_i$$

- The i-th bit leaked with probability $1/n$.

Constructing the Hard-Concentrate Bit Function

- Give OWP f: we don't know if it has a hard-concentrate bit function. (Open problem)
- Instead, given OWP f: we will give a OWF g which has a hard-concentrate bit function hc

$$g(x,r) = f(x)||r, \qquad |x| = |r|$$

$$hc(x,r) = \sum_{i=1}^{n} x_i \cdot r_i \ mod\ 2$$

Proof

Will only prove for A that predicts with probability *1*.

- If $\exists$ $PPT\ A$ that can predict $hc(x,r)$ given $g(x,r)$, then $\exists$ $PPT\ B$ that can invert $g(x,r)$ or $f(x)$

$B(y = f(x))$

1. For $i = 1 \ldots n$
   1. $x_i \leftarrow A(f(x), e_i)$, where $e_i = 0^{i-1}10^{n-i-1}$
2. Output x

What's wrong if probability is < 1?

Proof (Hint)

Proof hint for A that predicts with probability < *1*.

- If $\exists\ PPT\ A$ that can predict $hc(x,r)$ given $g(x,r)$, then $\exists\ PPT\ B$ that can invert $g(x,r)$ or $f(x)$

$B(y = f(x))$

1. For $i = 1 \ldots n$
   1. Sample $r \leftarrow \{0,1\}^n$
   2. $\alpha_i \leftarrow A(f(x), r)$
   3. $\beta_i \leftarrow A(f(x), r \oplus e_i)$, where $e_i = 0^{i-1}10^{n-i-1}$
   4. $x_i = \alpha_i \oplus \beta_i$
2. Output x

PRG from Hardness Concentration

Construction of PRG (1 extra bit) from OWP

- Given a OWP f, let g and hc be defined as before.
- Note that $g(x,r) = f(x)||r$ is also a OWP
- Define $PRG(x,r) = g(x,r)||hc(x,r)$

$f(x)$	r	$hc(x,r)$	H_0
$f(x)$	r	U_1	H_1
$f(x)$	U_n	U_1	H_2
U_n	U_n	U_1	H_3

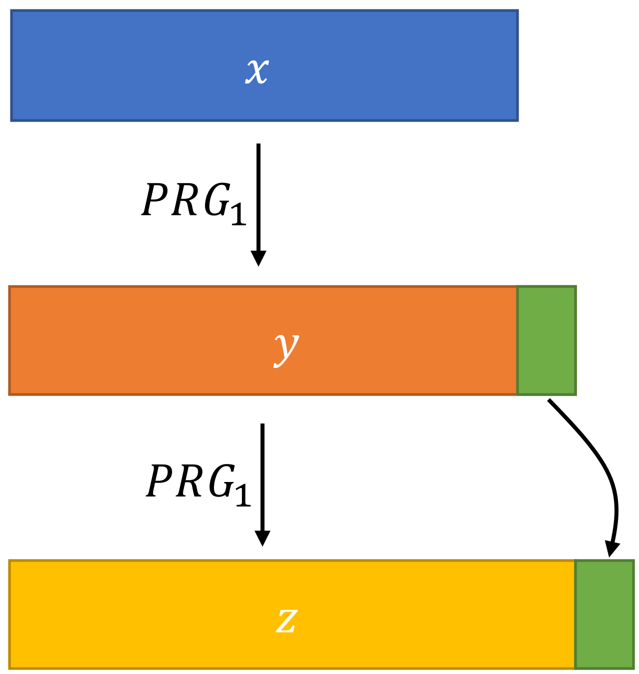
Getting beyond one bit

Going from one extra bit to more bits (PRG_2 from PRG_1)

$PRG_2(x_1 \dots x_n)$

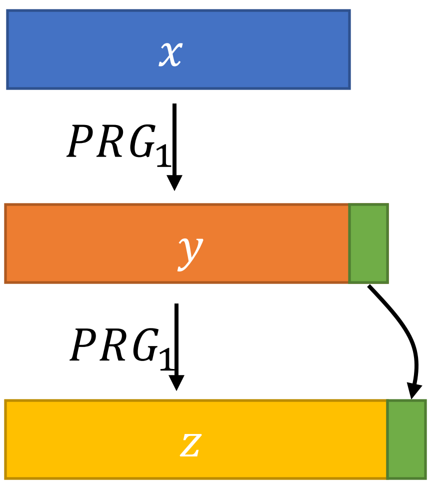
1. $y_1 \dots y_{n+1} = PRG_1(x_1 \dots x_n)$
2. $z_1 \dots z_{n+1} = PRG_1(y_1 \dots y_n)$
3. Output $z_1 \dots z_{n+1} y_{n+1}$

Going from one extra bit to more bits (PRG_2 from PRG_1)



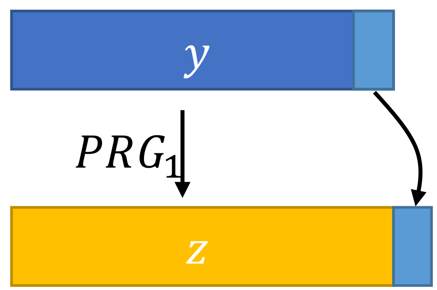
Security Proof

H_0



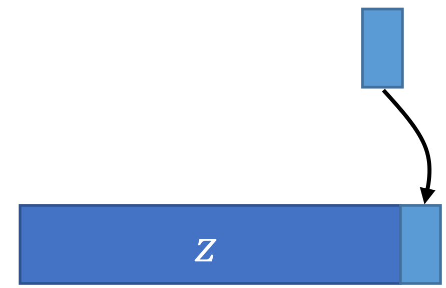
1. Sample $x_1 \ldots x_n$ from U_n
2. $y_1 \ldots y_{n+1} = PRG_1(x_1 \ldots x_n)$
3. $z_1 \ldots z_{n+1} = PRG_1(y_1 \ldots y_n)$
4. Output $z_1 \ldots z_{n+1} y_{n+1}$

H_1



1. Sample $y_1 \ldots y_n y_{n+1}$ from U_{n+1}
2. $z_1 \ldots z_{n+1} = PRG_1(y_1 \ldots y_n)$
3. Output $z_1 \ldots z_{n+1} y_{n+1}$

H_2



1. Sample y_{n+1} from U_1
2. Sample $z_1 \ldots z_{n+1}$ from U_{n+1}
3. Output $z_1 \ldots z_{n+1} y_{n+1}$

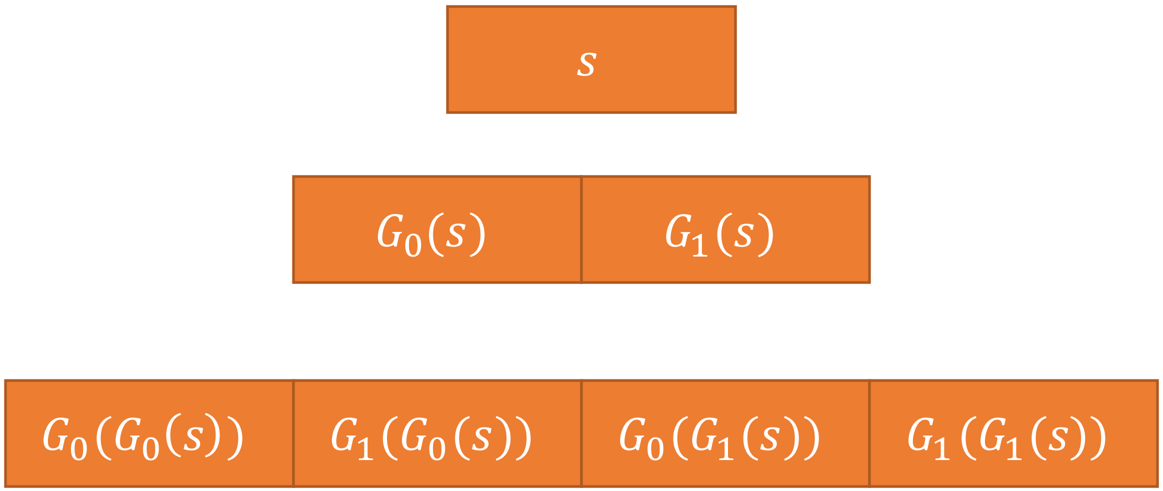
Going from one extra bit to more bits (PRG_2 from PRG_1)

Getting PRFs (High Level)

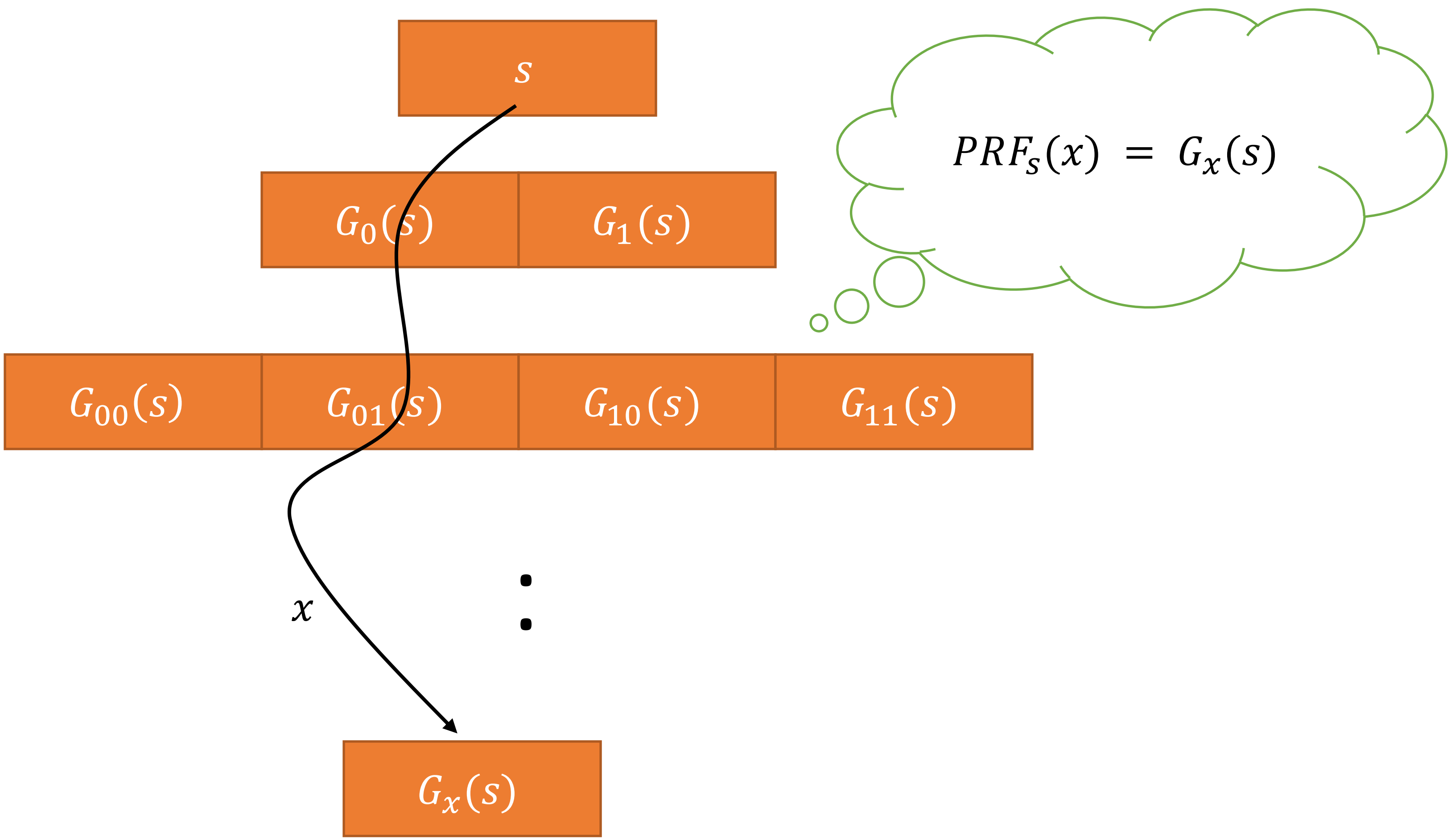
Pseudorandom Functions

- Construct PRF from PRG G: $\{0,1\}^n \rightarrow \{0,1\}^{2n}$
- Let $G_0(x)$ be left half of the output of $G(x)$
- Let $G_1(x)$ be right half of the output of $G(x)$

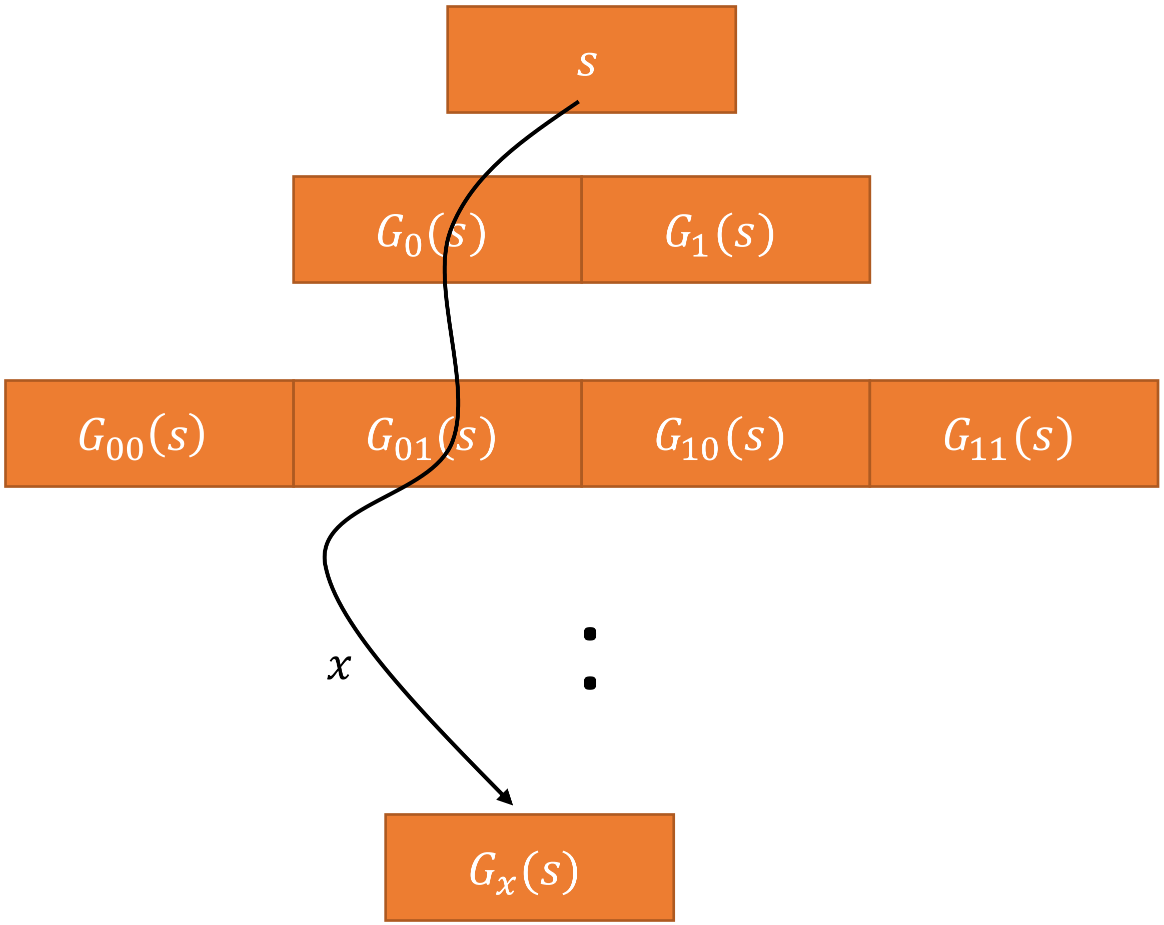
Pseudorandom Functions



Pseudorandom Functions



Proof Sketch



Thank You!