

PRFs from PRGs, and Toward Public-Key Cryptography: Groups and Discrete Log

CS 171: Cryptography

Sanjam Garg

October 13, 2026

Overview

- 1 Getting Beyond One Bit
- 2 PRFs from PRGs
- 3 Toward Public-Key Cryptography
- 4 Discrete Log
- 5 Summary

Recall: A PRG from a One-Way Permutation

- From a OWP f and its Hardness Concentration Bit:
 $\text{PRG}_1(x, r) = f(x) \parallel r \parallel \langle x, r \rangle$.
- *Proved:* a distinguisher for PRG_1 becomes a predictor for $\langle x, r \rangle$.
- But it stretches by only **one bit**.

Today

Stretch to any polynomial length, then build a **PRF** from a length-doubling PRG — and with it, everything in private-key cryptography from OWPs.

From One Extra Bit to Two

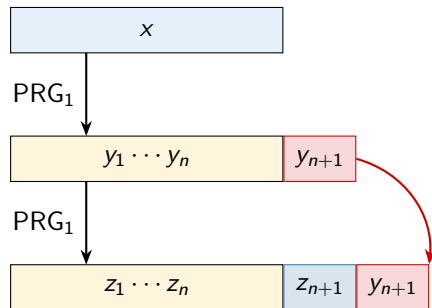
Let $\text{PRG}_1 : \{0, 1\}^n \rightarrow \{0, 1\}^{n+1}$ be a PRG with one bit of stretch. Build $\text{PRG}_2 : \{0, 1\}^n \rightarrow \{0, 1\}^{n+2}$:

$\text{PRG}_2(x_1 \cdots x_n)$

- 1 $y_1 \cdots y_n \textcolor{red}{y}_{n+1} := \text{PRG}_1(x_1 \cdots x_n)$.
- 2 $z_1 \cdots z_n \textcolor{blue}{z}_{n+1} := \text{PRG}_1(y_1 \cdots y_n)$.
- 3 Output $z_1 \cdots z_n \textcolor{blue}{z}_{n+1} \textcolor{red}{y}_{n+1}$.

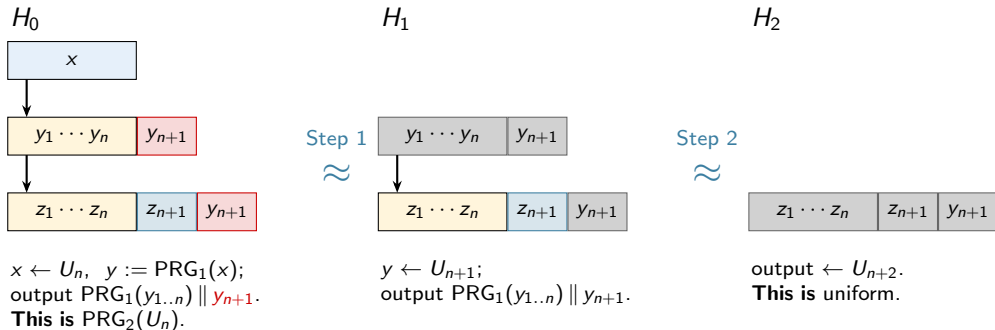
Idea

Set **one bit** aside; the other n bits look like a fresh seed, so stretch them **again**.



PRG₂ Is a PRG: Three Hybrids

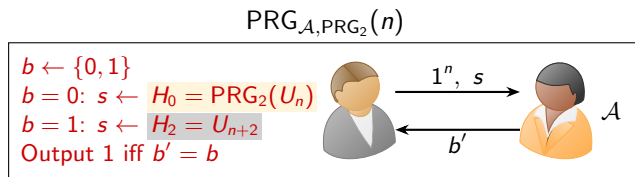
Replace the two PRG₁ calls by **truly uniform** strings (grey), **one call at a time**:



The PRG game for PRG₂ asks $\mathcal{A}$ to tell H_0 from H_2 . Each step changes **one** PRG₁ output into a uniform string.

PRG₂ Is a PRG: Suppose Not

Suppose some PPT $\mathcal{A}$ has $\Pr[\text{PRG}_{\mathcal{A}, \text{PRG}_2}(n) = 1] \geq \frac{1}{2} + \epsilon(n)$ for a non-negligible ϵ .



Write $\Pr[\mathcal{A}(H) = 1]$ for the probability that $\mathcal{A}$ outputs 1 on 1^n and $s \leftarrow H$. Since b is uniform:

$$\begin{aligned}\Pr[\text{PRG}_{\mathcal{A}, \text{PRG}_2}(n) = 1] &= \frac{1}{2} \Pr[\mathcal{A}(H_0) = 0] + \frac{1}{2} \Pr[\mathcal{A}(H_2) = 1] \\ &= \frac{1}{2} (1 - \Pr[\mathcal{A}(H_0) = 1]) + \frac{1}{2} \Pr[\mathcal{A}(H_2) = 1] \\ &= \frac{1}{2} + \frac{1}{2} (\Pr[\mathcal{A}(H_2) = 1] - \Pr[\mathcal{A}(H_0) = 1]).\end{aligned}$$

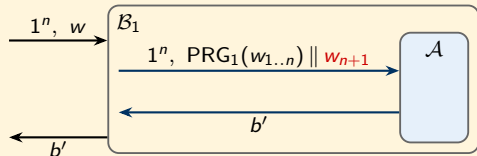
So $\mathcal{A}$ says 1 non-negligibly more often on H_2 than on H_0

$$\Pr[\mathcal{A}(H_2) = 1] - \Pr[\mathcal{A}(H_0) = 1] \geq 2\epsilon(n).$$

Each Step Is a Reduction to PRG_1

Each $\mathcal{B}_j$ plays the PRG game for PRG_1 , builds a hybrid from its challenge w , gives it to $\mathcal{A}$, and outputs $\mathcal{A}$'s guess b' .

$\mathcal{B}_1$ (Step 1): its challenge w plays y

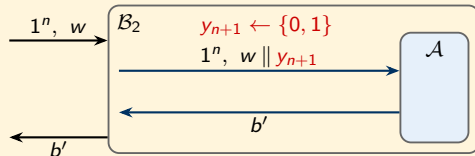


$b = 0, w \leftarrow \text{PRG}_1(U_n)$: $\mathcal{A}$ gets H_0 .

$b = 1, w \leftarrow U_{n+1}$: $\mathcal{A}$ gets H_1 . Same calculation as for $\mathcal{A}$:

$$\Pr[\text{PRG}_{\mathcal{B}_1, \text{PRG}_1}(n) = 1] = \frac{1}{2} + \frac{1}{2} (\Pr[\mathcal{A}(H_1) = 1] - \Pr[\mathcal{A}(H_0) = 1]).$$

$\mathcal{B}_2$ (Step 2): its challenge w plays z



$b = 0, w \leftarrow \text{PRG}_1(U_n)$: $\mathcal{A}$ gets H_1 .

$b = 1, w \leftarrow U_{n+1}$: $\mathcal{A}$ gets H_2 . Same calculation as for $\mathcal{A}$:

$$\Pr[\text{PRG}_{\mathcal{B}_2, \text{PRG}_1}(n) = 1] = \frac{1}{2} + \frac{1}{2} (\Pr[\mathcal{A}(H_2) = 1] - \Pr[\mathcal{A}(H_1) = 1]).$$

PRG₂: Putting It Together

From $\mathcal{A}$: $\Pr[\mathcal{A}(H_2) = 1] - \Pr[\mathcal{A}(H_0) = 1] \geq 2\epsilon(n)$. Add and subtract $\Pr[\mathcal{A}(H_1) = 1]$:

$$\begin{aligned} 2\epsilon(n) &\leq \Pr[\mathcal{A}(H_2) = 1] - \Pr[\mathcal{A}(H_0) = 1] \\ &= \underbrace{(\Pr[\mathcal{A}(H_1) = 1] - \Pr[\mathcal{A}(H_0) = 1])}_{\text{Step 1}} + \underbrace{(\Pr[\mathcal{A}(H_2) = 1] - \Pr[\mathcal{A}(H_1) = 1])}_{\text{Step 2}}. \end{aligned}$$

Two numbers that add up to at least $2\epsilon(n)$: **at least one** of them is $\geq \epsilon(n)$ (even if the other is negative). Say Step j . By the previous slide,

$$\Pr[\text{PRG}_{\mathcal{B}_j, \text{PRG}_1}(n) = 1] = \frac{1}{2} + \frac{1}{2} \cdot (\text{Step } j) \geq \frac{1}{2} + \frac{\epsilon(n)}{2},$$

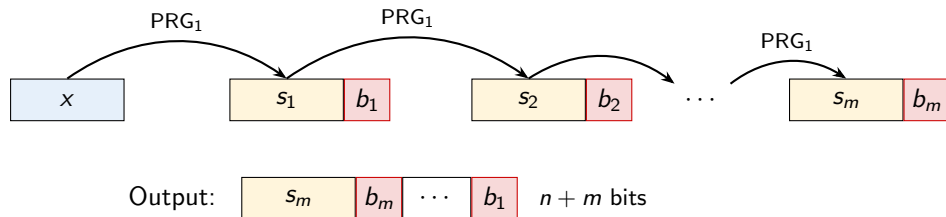
contradicting that PRG_1 is a PRG. □

Remark (running time)

Each $\mathcal{B}_j$ runs $\mathcal{A}$ once, plus one PRG_1 call or one coin flip: PPT.

Getting More Bits: PRG_m

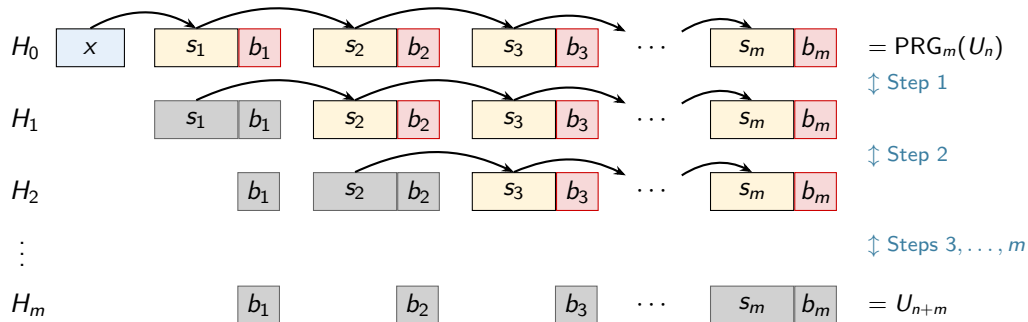
Keep going: each round sets aside one bit and re-stretches the remaining n .



With $m = n$: a **length-doubling** PRG $G : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ — the starting point for PRFs, next.

PRG_m Is a PRG: $m + 1$ Hybrids

Replace the m PRG₁ calls (arrows) by **truly uniform** strings (grey), **one at a time**:



H_i : $b_1, \dots, b_i$ and s_i uniform; rounds $i + 1, \dots, m$ use PRG₁. Step i (H_{i-1} to H_i) changes **one** PRG₁ output, $s_i || b_i$, into a uniform string.

As for PRG₂: the m step differences add up to $\geq 2\epsilon(n)$, so one of them is $\geq 2\epsilon(n)/m$ — still non-negligible, since m is polynomial in n .

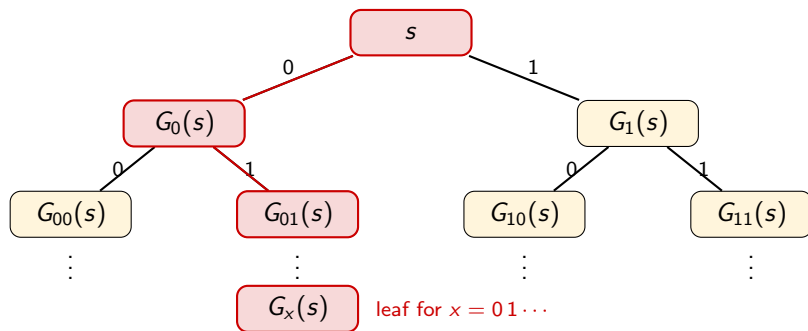
Getting PRFs: The GGM Idea

Goal

Build a PRF F — secure in the game $\text{PRF}_{\mathcal{A},F}(n)$ of lecture 5 — from a length-doubling PRG. This is the GGM (Goldreich–Goldwasser–Micali) construction.

- Start from a PRG $G : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ (length-doubling, e.g. PRG_n from before).
- Let $G_0(s)$ be the left half of $G(s)$, and $G_1(s)$ the right half.
- Idea: the input x chooses a root-to-leaf path in a binary tree of G applications.

The GGM Tree



Construction: $F_s(x) := G_x(s)$

For $x = x_1 \dots x_n$: $G_x(s) := G_{x_n}(\dots G_{x_2}(G_{x_1}(s)) \dots)$. Each step is one PRG call: n calls per evaluation.

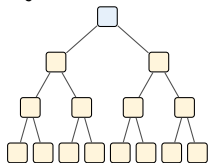
GGM Is a PRF: Hybrids, Level by Level

Theorem 1

If G is a length-doubling PRG, then $F_s(x) = G_x(s)$ is a PRF.

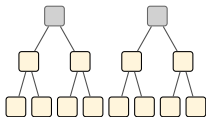
Replace the tree by truly uniform labels (grey), **one level at a time**; here $n = 3$:

H_0



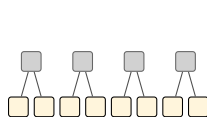
$s \leftarrow U_n$; G below.
This is F_s .

H_1



Depth-1 labels uniform;
 G below.

H_2



Depth-2 labels uniform;
 G below.

H_3



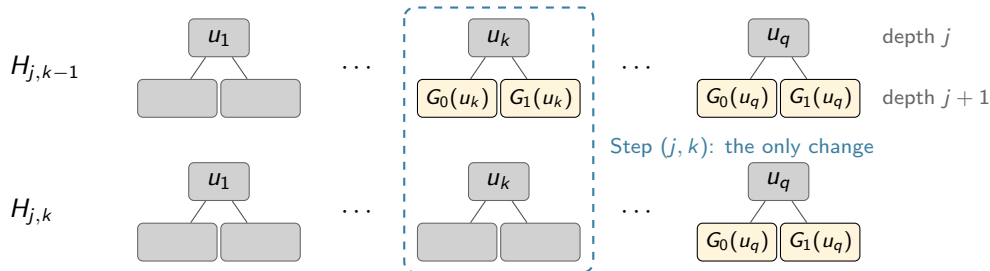
Leaves uniform: a truly
random function.

In general: hybrids $H_0, H_1, \dots, H_n$

H_j : each **depth- j** node gets a fresh uniform label when a query first reaches it; G below. The game $\text{PRF}_{\mathcal{A}, F}(n)$ asks $\mathcal{A}$ to tell oracle H_0 from oracle H_n .

GGM: From H_j to H_{j+1} , One Node at a Time

$\mathcal{A}$'s q queries reach at most q depth- j nodes; call their (uniform) labels $u_1, \dots, u_q$, in the order first reached. In $H_{j,k}$, the children of $u_1, \dots, u_k$ are **uniform**; those of every later u_i are $G(u_i)$.

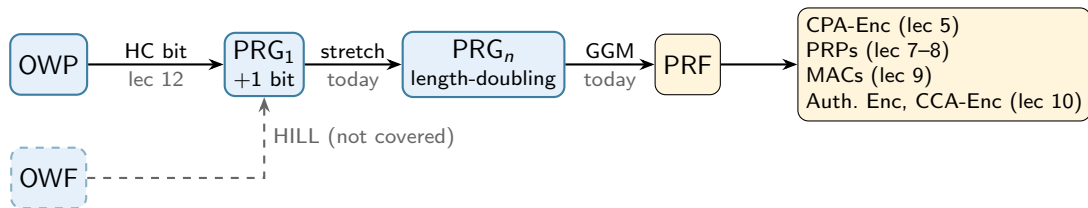


$H_{j,0} = H_j$ and $H_{j,q} = H_{j+1}$. Step (j, k) changes **one** G output, $G(u_k)$, into a uniform string.

Counting

n levels $\times q$ nodes $= nq$ steps, whose differences add up to $\geq 2\epsilon(n)$. So one step is $\geq 2\epsilon(n)/(nq)$: non-negligible, since q is polynomial in n .

Wrap-Up: Private-Key Cryptography from One-Way Functions



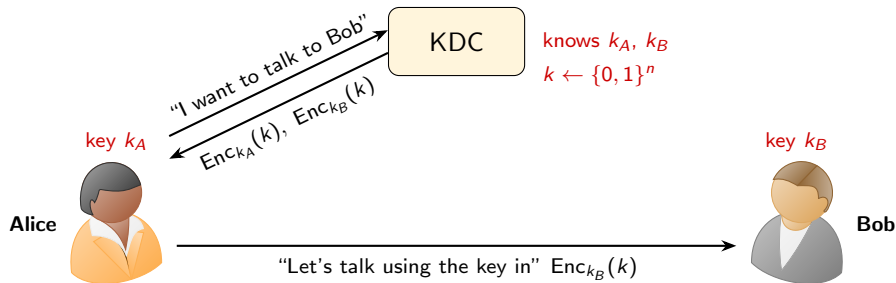
Drawbacks of Private-Key Cryptography

- **Key distribution is a problem.** Every pair of parties must *already* share a secret key, established over a secure channel before any communication.
- **Storing a large number of keys is problematic.** A network of n users who all wish to communicate needs $\binom{n}{2}$ distinct keys; each user stores $n - 1$ keys.
- **Inapplicable to open systems.** Two parties who have never met (e.g. a browser and a web server) cannot communicate securely.

Goal

Secure communication *without* a pre-shared secret key.

A Partial Solution: Key-Distribution Center



- Each user shares one long-term key with the KDC: $O(n)$ keys instead of $O(n^2)$.
- But the KDC must be **online** for every session and **fully trusted** (it knows every key k).

Public-Key Cryptography

The big idea (Diffie–Hellman, 1976)

Each party has a **public key** (published) and a **secret key** (kept private). Knowing the public key lets anyone interact securely with its owner.

Plan

First some **number theory** (the group $\mathbb{Z}_p^*$, discrete log, Diffie–Hellman), then **key exchange**.

Cryptographic Group

The group $\mathbb{Z}_p^*$

For a prime p : $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$, with $a \cdot b := ab \bmod p$. Every element has an inverse (as p is prime), so $\mathbb{Z}_p^*$ is a **group**, of **order** (size) $p-1$.

- For $g \in \mathbb{Z}_p^*$, its powers $\langle g \rangle := \{g, g^2, g^3, \dots\}$ form a group, **cyclic** with **generator** g .
- The **order** of g is the smallest $i > 0$ with $g^i = 1$: the size of $\langle g \rangle$.

Setup

Let p and q be primes with $p-1 = 2q$, and let $g \in \mathbb{Z}_p^*$ be an element of order q . Then $G = \langle g \rangle$ is a cyclic group of **prime** order q .

Example 2 ($p = 23$ and $q = 11$)

$\mathbb{Z}_{23}^* = \{1, 2, \dots, 22\}$, $a \cdot b := ab \bmod 23$.

Cyclic Subgroups of $\mathbb{Z}_{23}^*$

$\langle g \rangle$ for some $g \in \mathbb{Z}_{23}^*$

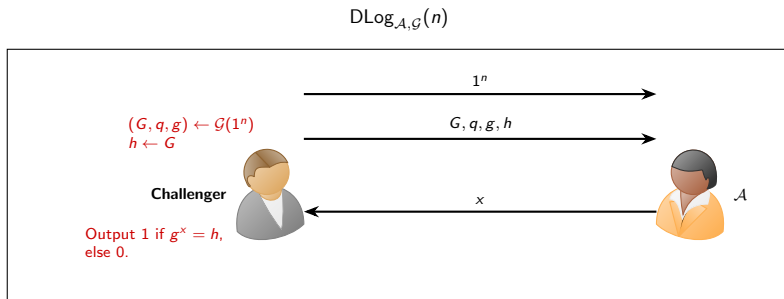
- $\langle 1 \rangle = \{1\}$ order 1
- $\langle 2 \rangle = \{2, 4, 8, 16, 9, 18, 13, 3, 6, 12, \underbrace{2^{11}}_{=1}\}$ order 11
- $\langle 5 \rangle = \{5, 2, 10, 4, 20, 8, 17, 16, 11, 9, \dots, 5^{22} = 1\}$ order 22: all of $\mathbb{Z}_{23}^*$
- $\langle 22 \rangle = \{22, 22^2 = 1\}$ order 2 ($22 = -1$)

- Every order divides $22 = |\mathbb{Z}_{23}^*|$. Pick **any** $g \neq 1$ with $g^{11} = 1$: e.g. $G = \langle 2 \rangle$ has prime order 11. **This is the group we want.**
- For hardness, use **large** primes.

The Discrete-Log Problem: Group Generator

- Let $\mathcal{G}$ be a PPT algorithm that on input 1^n outputs the description of a **cyclic** group G , its order q (with $|q| = n$), and a generator g .
- Each element has a **unique bit representation**, and the group operation is computable in time polynomial in n .
- **Sampling a uniform group element:** sample $x \leftarrow \mathbb{Z}_q$ and compute g^x .

$\text{DLog}_{\mathcal{A}, \mathcal{G}}$: Challenger vs. Adversary



Remark

Computing g^x from x is easy (repeated squaring); going back from h to x is assumed hard. So $x \mapsto g^x$ is a candidate **one-way function** (lecture 11).

The Discrete-Log Problem

Game $\text{DLog}_{\mathcal{A}, \mathcal{G}}(n)$

- 1 $(G, q, g) \leftarrow \mathcal{G}(1^n)$, and $h \leftarrow G$ uniform.
- 2 $\mathcal{A}$ is given 1^n and (G, q, g, h) , and outputs $x \in \mathbb{Z}_q$.
- 3 Output 1 if $g^x = h$, and 0 otherwise.

Definition 3 (Discrete-log hardness)

The discrete-log problem is **hard relative to** $\mathcal{G}$ if for all PPT $\mathcal{A}$:

$$\Pr [\text{DLog}_{\mathcal{A}, \mathcal{G}}(n) = 1] \leq \text{negl}(n).$$

Collision-Resistant Hash Functions from DLog

Let $\mathcal{G}$ output groups of **prime** order q .

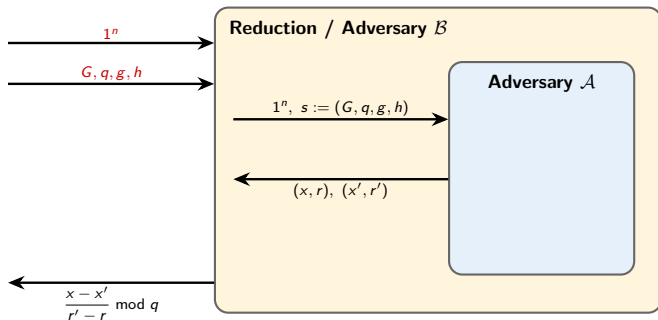
Construction (Gen, H)

- $\text{Gen}(1^n)$: $(G, q, g) \leftarrow \mathcal{G}(1^n)$; sample a uniform $h \in G$; output the key $s = (G, q, g, h)$.
- $H_s(x, r) := g^x h^r$, for $x, r \in \mathbb{Z}_q$.
- Maps two exponents $(x, r) \in \mathbb{Z}_q \times \mathbb{Z}_q$ to one group element: it compresses when group elements have about n bits (e.g. elliptic curves).
- The key s is public, as in Hash-coll (lecture 11). Note that nobody needs to know $\log_g h$.

Theorem 4

If the discrete-log problem is hard relative to $\mathcal{G}$, then (Gen, H) is collision resistant.

Proof by Reduction (If DLog Is Hard, Then CRHF)



Given / To prove

Given: $\Pr[\text{Hash-coll}_{\mathcal{A}, (\text{Gen}, H)}(n) = 1] \geq \epsilon(n)$ for non-negligible ϵ .

To prove: $\Pr[\text{DLog}_{\mathcal{B}, G}(n) = 1] \geq \epsilon(n)$.

CRHF from DLog: Putting It Together

The key s that $\mathcal{B}$ gives $\mathcal{A}$ is distributed exactly as in Gen. Suppose $\mathcal{A}$ outputs a collision: $(x, r) \neq (x', r')$ and

$$g^x h^r = g^{x'} h^{r'}.$$

- **Can** $r = r'$? Then $g^x = g^{x'}$, so $x = x' \bmod q$ — not a collision. So $r \neq r'$.
- Then $g^{x-x'} = h^{r'-r}$, and $r' - r$ is invertible modulo the **prime** q :

$$h = g^{(x-x')(r'-r)^{-1} \bmod q}.$$

Success probability of $\mathcal{B}$

$\mathcal{B}$ outputs $\log_g h$ whenever $\mathcal{A}$ wins: $\Pr [\text{DLog}_{\mathcal{B}, \mathcal{G}}(n) = 1] \geq \epsilon(n)$, contradicting the hardness of discrete log. □

Summary

- **Stretch:** $\text{PRG}_1 \Rightarrow \text{PRG}_m$ for any polynomial m , proved by hybrids.
- **GGM:** length-doubling PRG $\Rightarrow$ PRF. So OWPs give all of private-key crypto.
- **Public-key crypto:** no pre-shared keys; built on groups such as $\mathbb{Z}_p^*$.
- **Discrete log:** given g^x , find x ; assumed hard in prime-order groups.
- **CRHF from DLog:** $H_s(x, r) = g^x h^r$; a collision reveals $\log_g h$.