COS433/Math 473: Cryptography

Mark Zhandry

Princeton University

Fall 2020

Announcements/Reminders

Last day to turn in HW5
HW6 released soon

PR2 due Dec 5

Previously on COS 433…

Zero Knowledge

Interactive Proof

Statement **x**

Witness **w**

Zero Knowledge

For every malicious verifier **V***, ∃ "simulator" 🦁
s.t. for every true statement **x**, valid witness **w**,

$$🦁(x) \quad \approx_c \quad P(x,w) \underset{\rightarrow}{\overset{\rightarrow}{\leftarrow}} V^*(x)$$

QR Protocol

Statements: $\mathbf{x}$ is a Q.R. mod $\mathbf{N}$

Witness: $\mathbf{w}$ s.t. $\mathbf{w^2 \bmod N = x}$

Protocol:

$\mathbf{w}$

$u \leftarrow Z_N^*$

$y \leftarrow u^2 \bmod N$

y →

← b

$z = w^b u \bmod N$ →

$b \leftarrow \{0,1\}$

↓

$z^2 == x^b y \bmod N?$

Today

Zero knowledge proofs of knowledge
Crypto from minimal assumptions

Proofs of Knowledge

Sometimes, not enough to prove that statement is true, also want to prove "knowledge" of witness

Ex:

- Identification protocols: prove knowledge of key
- Discrete log: always exists, but want to prove knowledge of exponent.

Proofs of Knowledge

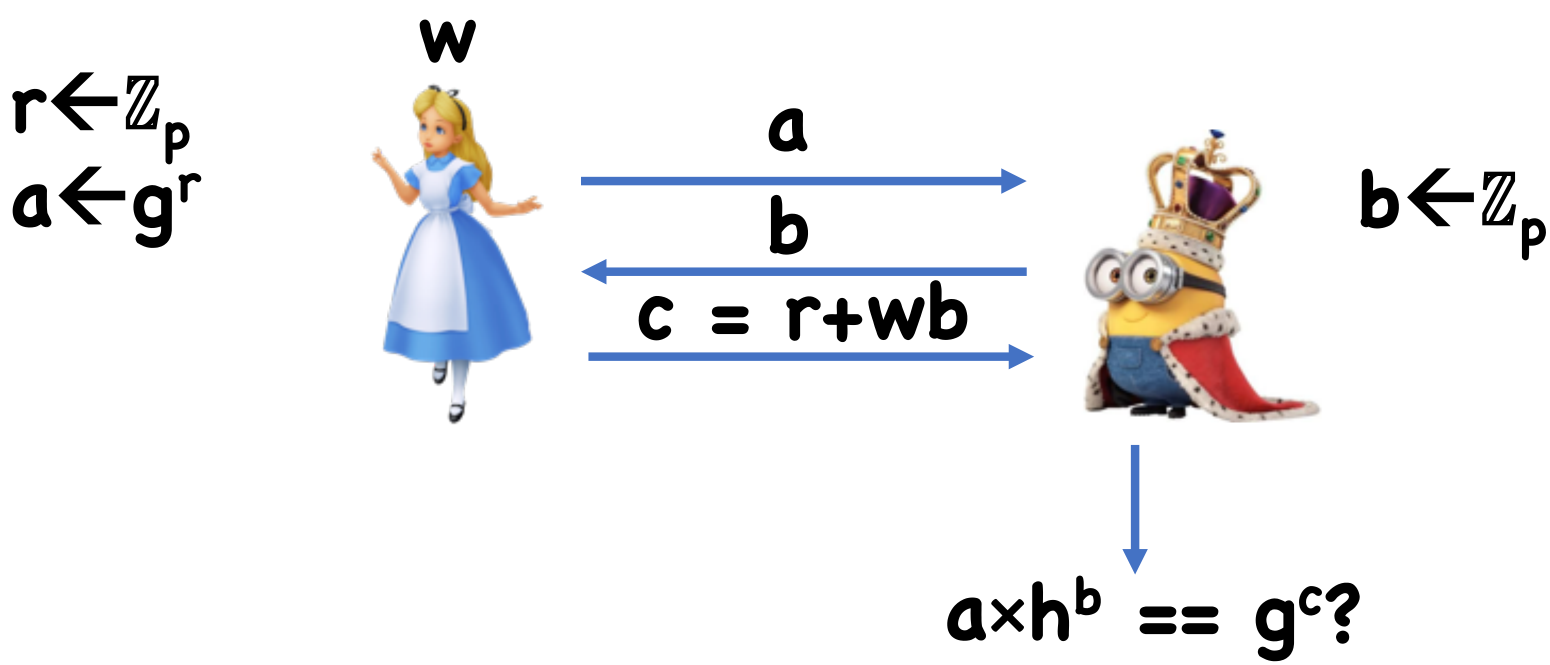
We won't formally define, but here's the intuition:

Given any (potentially malicious) PPT prover **P*** that causes **V** to accept, it is possible to "extract" from **P*** a witness **w**

Schnorr PoK for DLog

Statement: **(g,h)**
Witness: **w** s.t. **$h=g^w$**

Protocol:



Schnorr PoK for DLog

Completeness:

- $g^c = g^{r+wb} = a \times h^b$

Honest Verifier ZK:

- Transcript = (a,b,c) where $a=g^c/h^b$ and (b,c) random in $\mathbb{Z}_p$
- Can easily simulate. How?

Schnorr PoK for DLog

Proof of Knowledge?

Idea: once Alice commits to $a=g^r$, show must be able to compute $c = r+bw$ for any b of Bob's choosing

- Intuition: only way to do this is to know w
- Run Alice on two challenges, obtain:

$$c_0 = r_0 + b_0 w, c_1 = r_1 + b_1 w$$

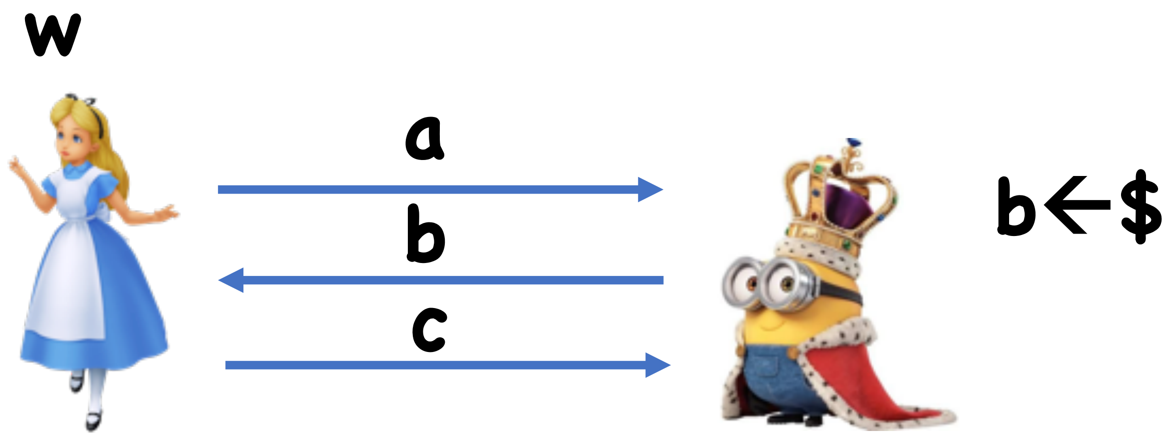
(Can solve linear equations to find w)

Deniability

Zero Knowledge proofs provide deniability:

- Alice proves statement $\mathbf{x}$ is true to Bob
- Bob goes to Charlie, and tries to prove $\mathbf{x}$ by providing transcript
- Charlie not convinced, as Bob could have generated transcript himself
- Alice can later deny that she knows proof of $\mathbf{x}$

∑ Protocols



(fancy name for 3-round "public coin" protocols)

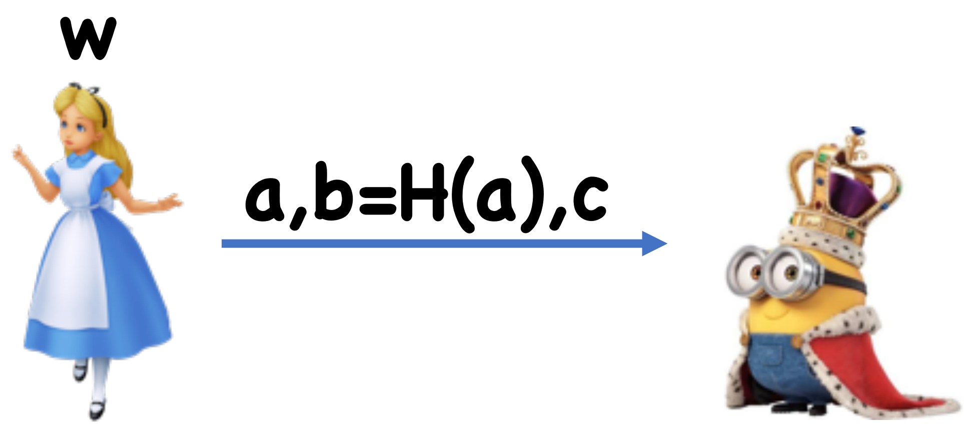
Fiat-Shamir Transform

Idea: set **b = H(a)**

- Since **H** is a random oracle, **a** is a random output

Notice: now prover can compute **b** for themselves!

- No need to actually perform interaction



Theorem: If (P,V) was a secure ZKPoK for honest verifiers, and if H is a random oracle, then compiled protocol is a ZKPoK

Proof idea: second message is exactly what you'd expect in original protocol

Complication: adversary can query H to learn second message, and throw it out if she doesn't like it

Signatures from ∑ Protocols

Idea: what if set **b = H(m,a)**

- Challenge **b** is message specific
- Intuition: proves that someone who knows **sk** engaged in protocol depending on **m**
- Can use resulting transcript as signature on **m**

Schnorr PoK → Schnorr Signatures

Applications of ZK (PoK)

Identification protocols: prove that you know the secret without revealing the secret

Signatures: prove that you know the secret in a “message dependent” way

Protocol Design:

- E.g. CCA secure PKE
  - To avoid mauling attacks, provide ZK proof that ciphertext is well formed
  - Problem: ZK proof might be malleable
  - With a bit more work, can be made CCA secure
- Example: multiparty computation
  - Prove that everyone behaved correctly

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 $\mathbf{F}: \mathcal{D} \rightarrow \mathcal{R}$

No correctness properties other than deterministic

Security?

Definition: F is One-Way if, for all polynomial time , ∃ negligible **ε** such that:

Pr[x←**(F(x)),x←D] < ε**

Trivial example:

F(x) = parity of **x**

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

Security

Definition: F is One-Way if, for all polynomial time , $\exists$ negligible **ε** such that:

$\Pr[F(x)=F(y) : y \leftarrow$  $(F(x)), x \leftarrow D] < \varepsilon$

Examples

Any PRG

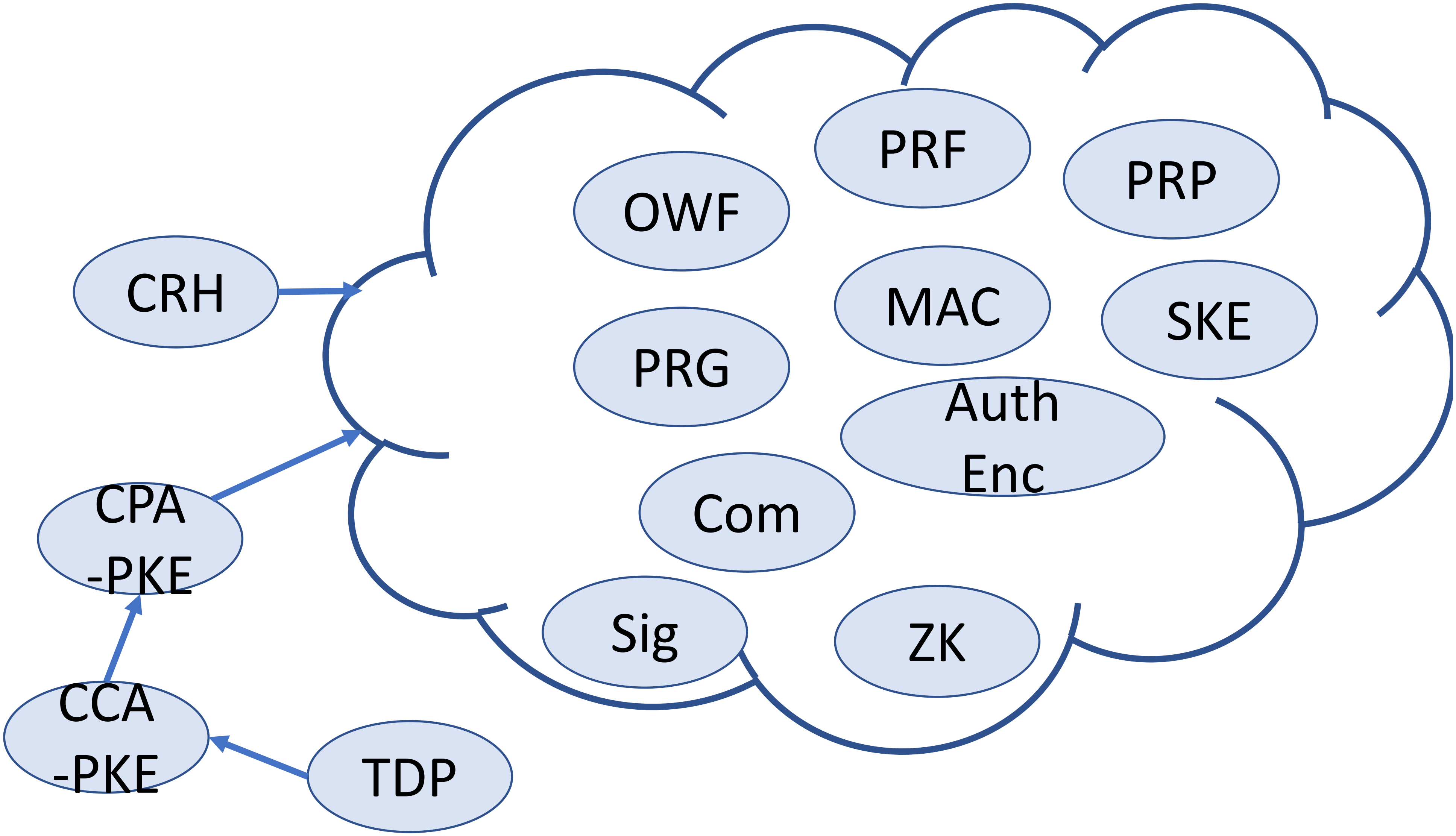
Any Collision Resistant Hash Function (with sufficient compression)

$F(p,q) = pq$

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

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

What's Known



OWF
PRF
PRP
MAC
SKE
PRG
Auth Enc
Com
Sig
ZK
CRH
CPA -PKE
CCA -PKE
TDP

Theory vs Practice

Most arrows are “feasibility” results

- Can build A from B in principle
- But sometimes horribly inefficient

In practice, typically start from powerful building blocks, e.g.

- PRPs
- TDPs
- Discrete log/DDH

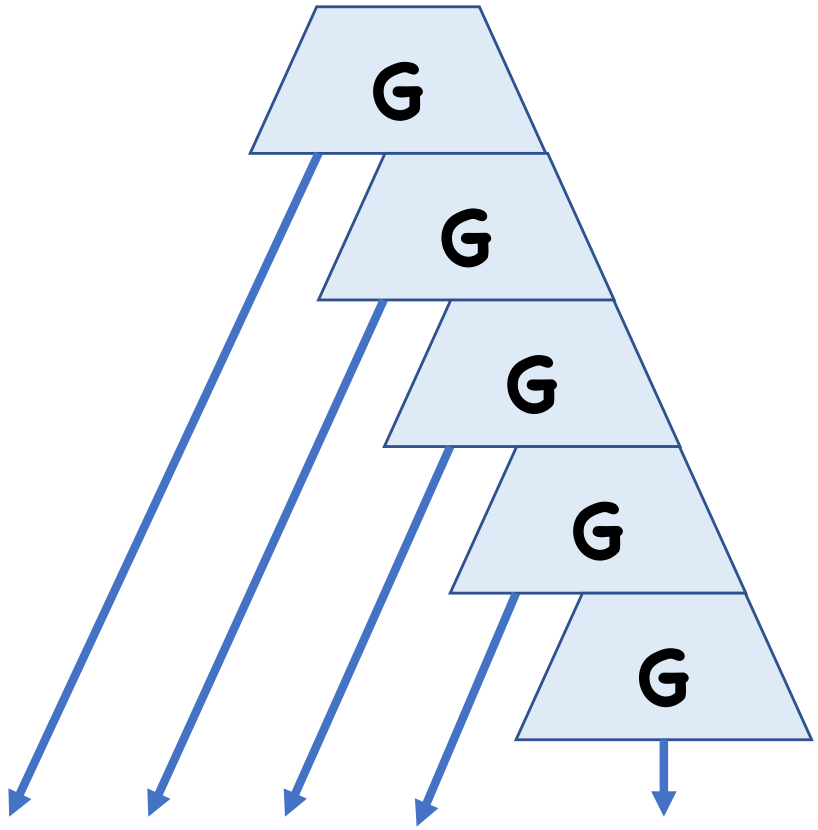
Roadmap

We will just prove a subset of implications

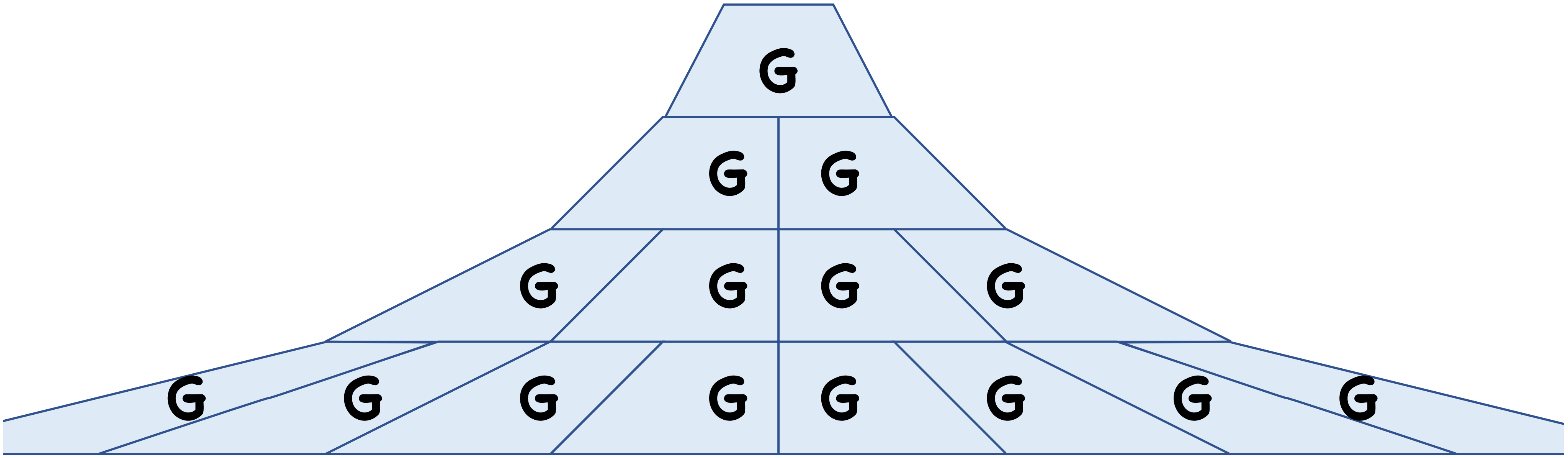
- PRGs → PRFs
- One-way *permutation* → PRGs
- OWF → One-time Signatures (if time)

PRGs → PRFs

First: Expanding Length of PRGs



A Different Approach



Advantage of Tree-based Approach

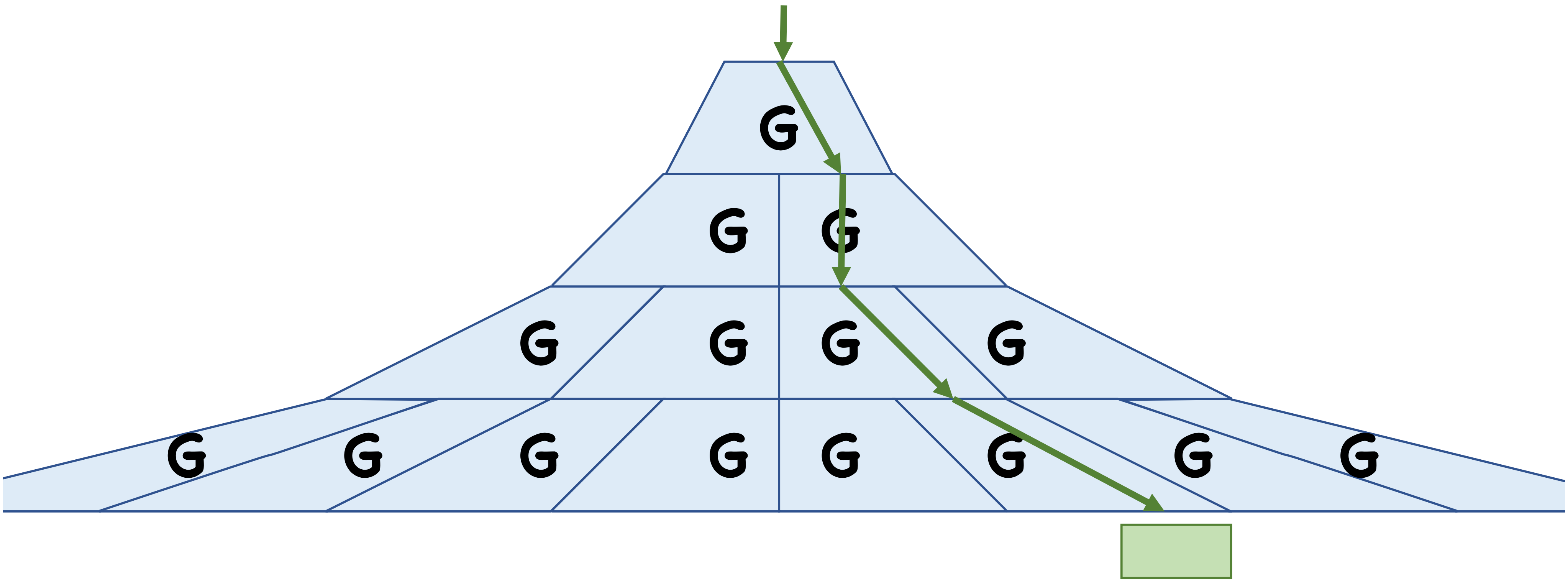
To expand λ bits into $2^h\lambda$ bits, need h levels

Can compute output locally:

- To compute ith chunk of λ bits, only need h PRG evaluations

In other words, can locally compute in logarithmic time

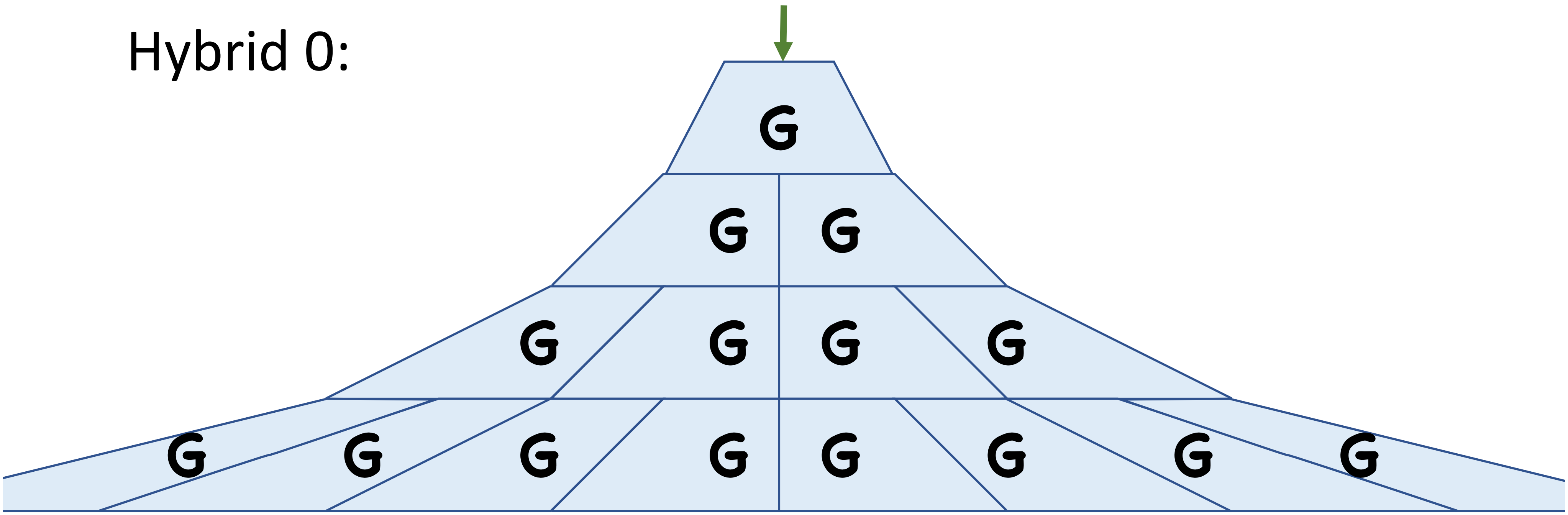
Advantage of Tree-based Approach



Theorem: For any logarithmic **h**, if **G** is a secure PRG, then so is the tree-based PRG

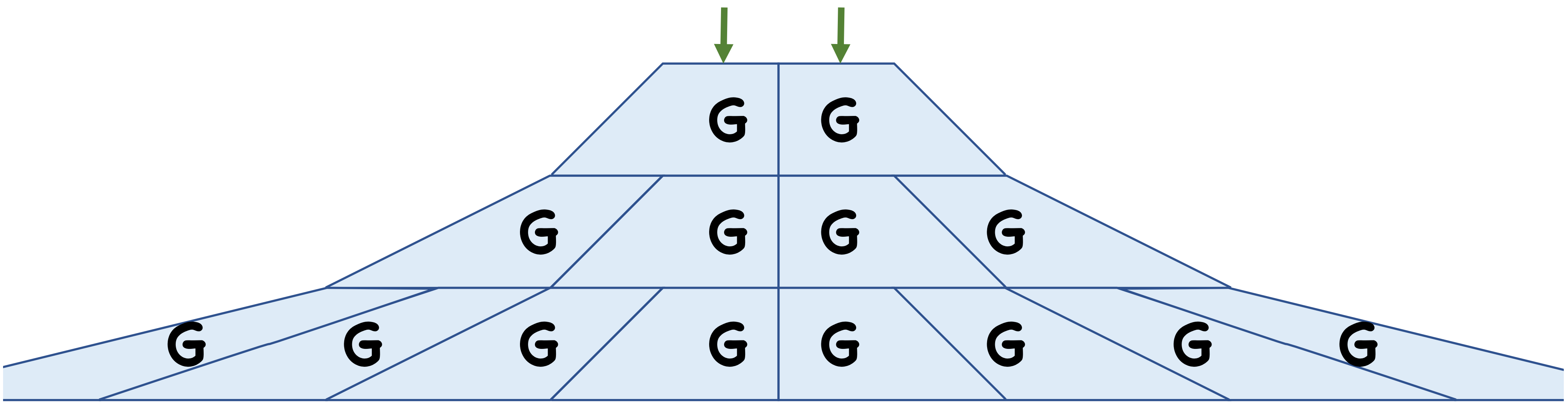
Proof

Hybrid 0:



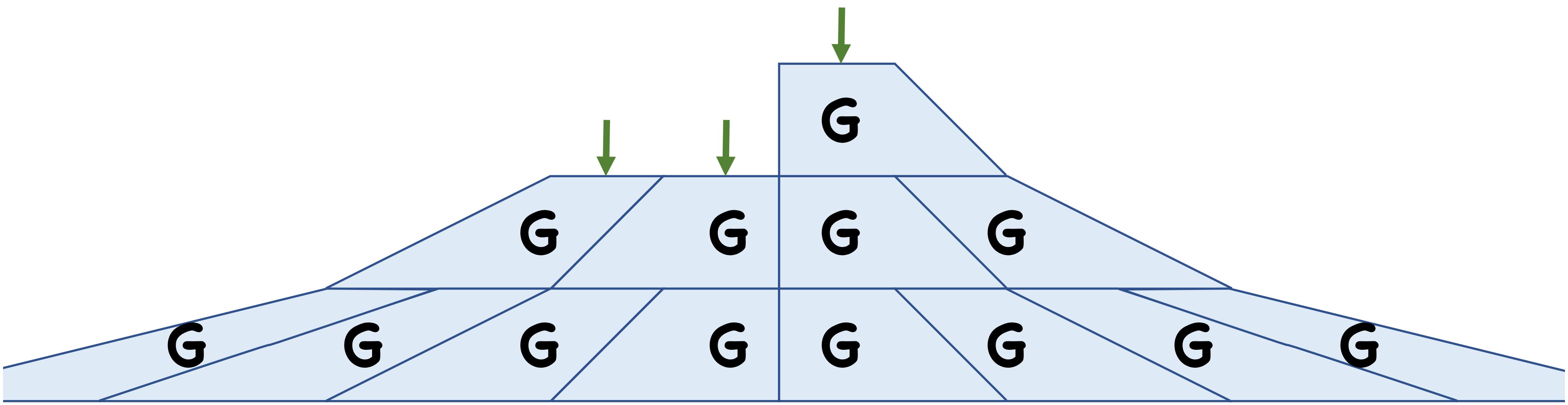
Proof

Hybrid 1:



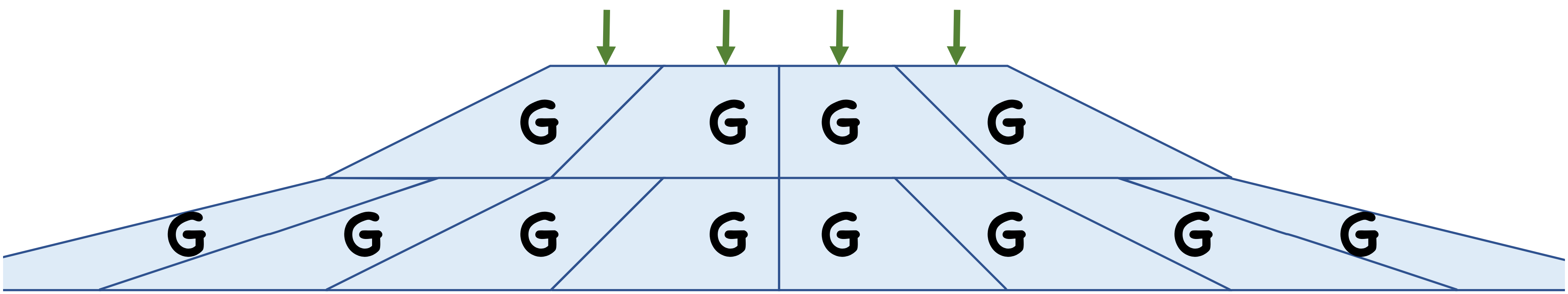
Proof

Hybrid 2:



Proof

Hybrid 3:



Proof

Hybrid $\mathbf{t}$:

Proof

What is **t** in terms of **h**?

PRG adversary distinguishes Hybrid 0 from Hybrid **t** with advantage **ε**

- $\exists$ **i** such that adversary distinguishes Hybrid **i-1** from Hybrid **i** with advantage **ε/t**
- Can use to construct adversary for **G** with advantage **ε/t**

A PRF

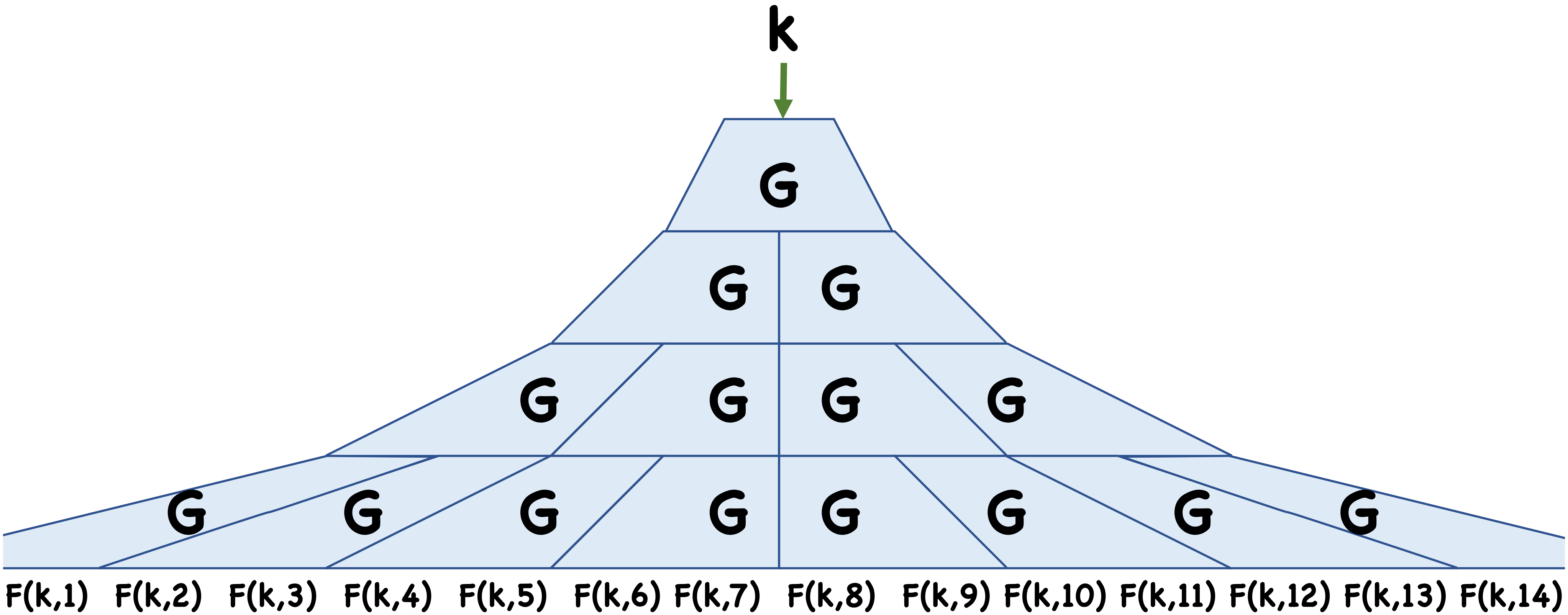
Domain $\{0,1\}^n$

Set $h = n$

$F(k, x)$ is the xth block of λ bits

- Computation involves h evals of G, so efficient

A PRF

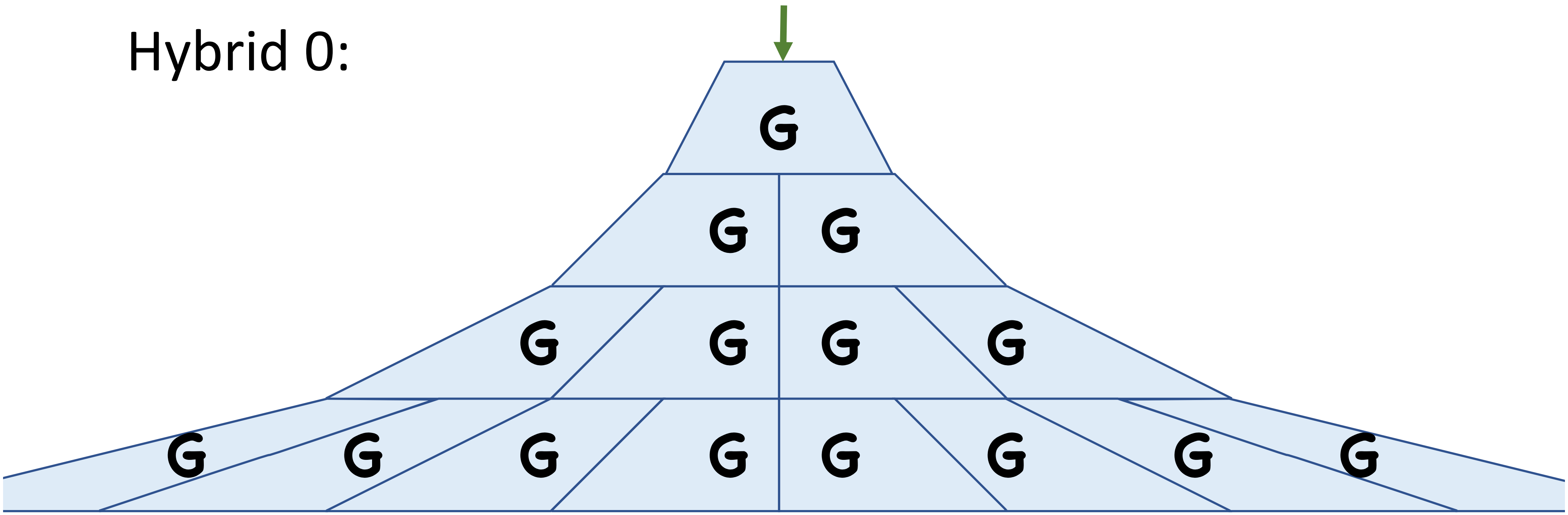

k
G
G
G
G
G
G
G
G
G
G
G
G
G
G
G
F(k,1)
F(k,2)
F(k,3)
F(k,4)
F(k,5)
F(k,6)
F(k,7)
F(k,8)
F(k,9)
F(k,10)
F(k,11)
F(k,12)
F(k,13)
F(k,14)

Problem with Security Proof

Suppose we have a PRF adversary with advantage **ε**. In the proof, what is the advantage of the derived PRG adversary?

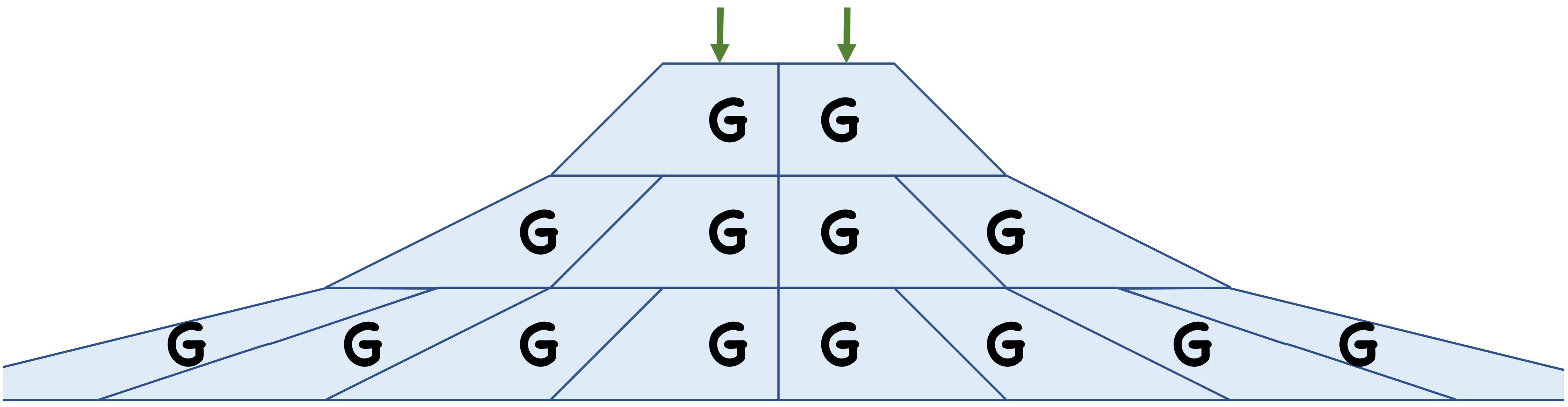
A Better Proof

Hybrid 0:



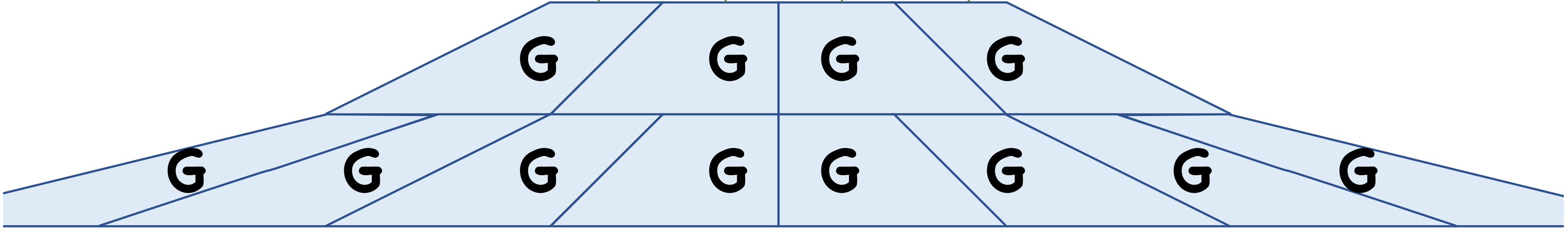
A Better Proof

Hybrid 1:



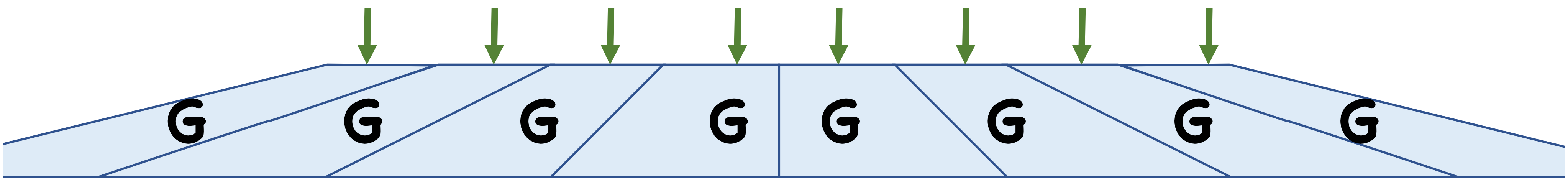
A Better Proof

Hybrid 2:



A Better Proof

Hybrid 3:



A Better Proof

Hybrid **h=n**:

A Better Proof

Now if PRF adversary distinguishes Hybrid 0 from Hybrid $h=n$ with advantage ε, $\exists i$ such that adversary distinguishes Hybrid $i-1$ from Hybrid i with advantage ε/n

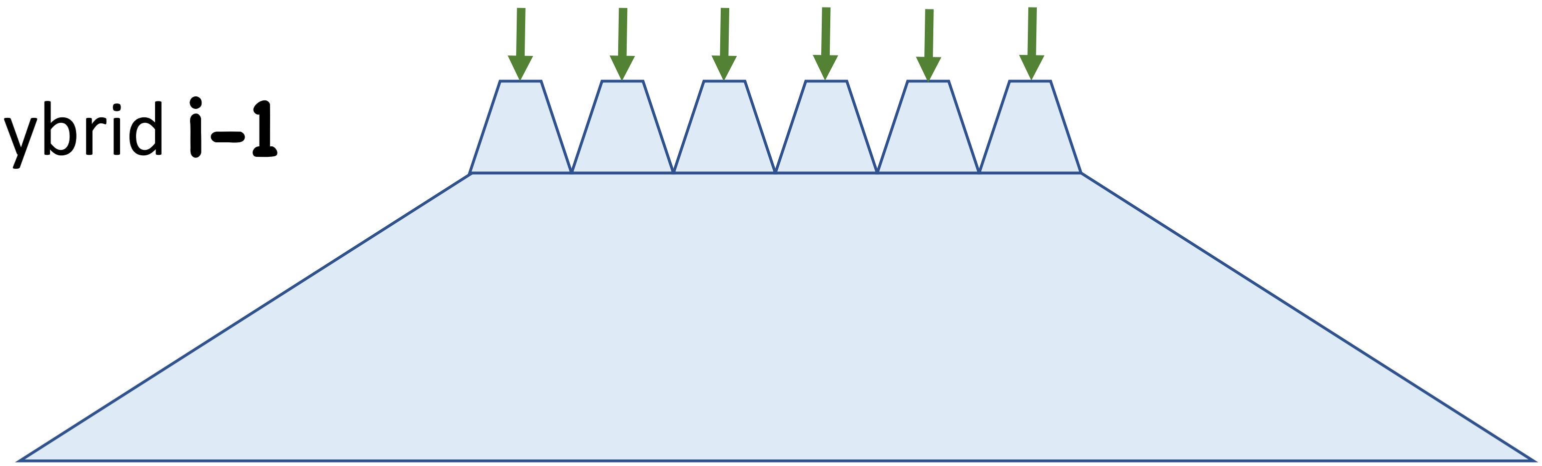
- Non-negligible advantage

Not quite done: Distinguishing Hybrid $i-1$ from Hybrid i does not immediately give a PRG distinguisher

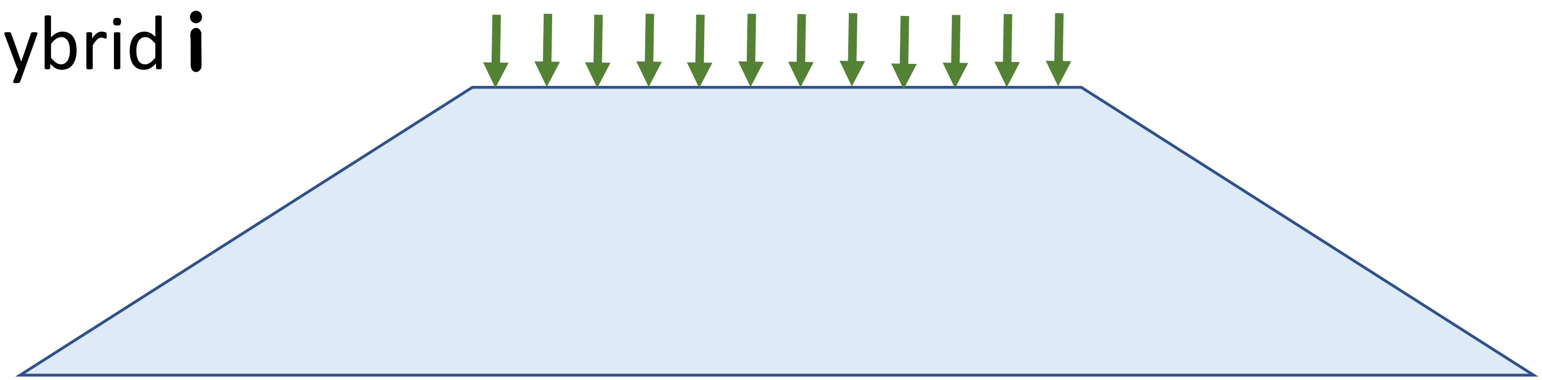
- Exponentially many PRG values changed!

A Better Proof

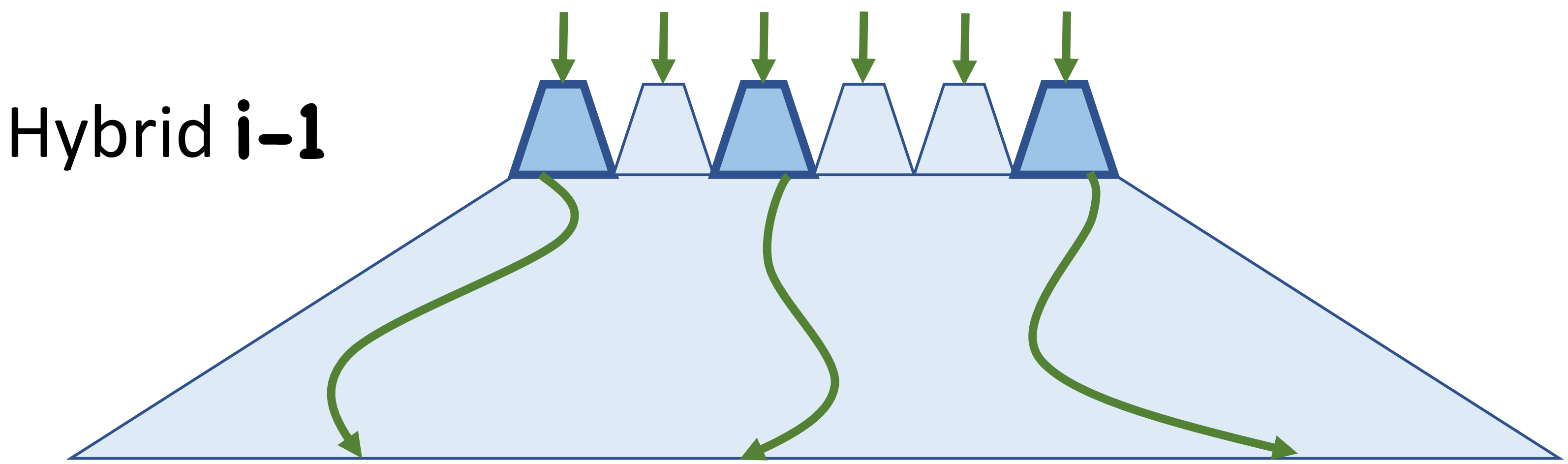
Hybrid **i−1**



Hybrid **i**



Key Observation:



Adversary only queries polynomially many outputs
⇒ Only need to worry about polynomially many PRG instances in level i

A Better Proof

More Formally:

Given distinguisher **A** for Hybrid $\mathbf{i-1}$ and Hybrid $\mathbf{i}$, can construct distinguisher **B** for the following two oracles from $\{0,1\}^{i-1} \rightarrow \{0,1\}^{2\lambda}$

- $\mathbf{H_0}$: each output is a fresh random PRG sample
- $\mathbf{H_1}$: each output is uniformly random

If **A** makes $\mathbf{q}$ queries, **B** makes at most $\mathbf{q}$ queries

A Better Proof

Now we have a distinguisher B with advantage **ε/n** that sees at most **q** values, where either

- Each value is a random output of the PRG, or
- Each value is uniformly random

By introducing **q** hybrids, can construct a PRG distinguisher with advantage **ε/qn**

$\Rightarrow$ non-negligible