Adaptive Security of Constrained PRFs

Georg Fuchsbauer (IST Austria)
Momchil Konstantinov (LSGNT,UK)
Krzysztof Pietrzak (IST Austria)
Vanishree Rao (UCLA)

UPC, 18 July 2014

Overview

We consider adaptive/full security of recent primitive:

constrained PRFs

[+] Improve reduction of GGM from exponential loss to quasipolynomial loss

[-] Show that exponential loss for Boneh-Waters CPRF is inherent

PRG

$\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a (length-doubling) (ϵ, s)-secure pseudorandom generator (PRG) if for all A, $|\mathsf{A}| \leq s$

$$|\Pr_{x \leftarrow U_\lambda}[\mathsf{A}(\mathsf{G}(x)) \to 1] - \Pr_{y \leftarrow U_{2\lambda}}[\mathsf{A}(y) \to 1]| \leq \epsilon$$

Crypto 101

PRG

$\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ is a (length-doubling) (ϵ, s)-secure pseudorandom generator (PRG) if for all A, $|\mathsf{A}| \leq s$

$$|\Pr_{x \leftarrow U_\lambda}[\mathsf{A}(\mathsf{G}(x)) \to 1] - \Pr_{y \leftarrow U_{2\lambda}}[\mathsf{A}(y) \to 1]| \leq \epsilon$$

PRF

$\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a (ϵ, s, q)-secure pseudorandom function (PRF) if for all A, $|\mathsf{A}| \leq s$, making $\leq q$ queries:

$$\left|\Pr_{K \leftarrow \mathcal{K}}[\mathsf{A}^{\mathsf{F}(K,.)} \to 1] - \Pr_{f \leftarrow \mathcal{F}[\mathcal{X},\mathcal{Y}]}[\mathsf{A}^{f(.)} \to 1]\right| \leq \epsilon$$

Constrained PRFs [BW13]

constrained PRF

$\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a constrained PRF for set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if there is

- constrained key space $\mathcal{K}_c$
- $\mathsf{F.cstr}\colon \mathcal{K} \times \mathcal{S} \to \mathcal{K}_c$
- $\mathsf{F.eval}\colon \mathcal{K}_c \times \mathcal{X} \to \mathcal{Y}$, s.t.

Constrained PRFs [BW13]

constrained PRF

$\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a constrained PRF for set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if there is

- constrained key space $\mathcal{K}_c$
- $\mathsf{F.cstr}\colon \mathcal{K} \times \mathcal{S} \to \mathcal{K}_c$
- $\mathsf{F.eval}\colon \mathcal{K}_c \times \mathcal{X} \to \mathcal{Y}$, s.t.

$$k_S \leftarrow \mathsf{F.cstr}(k, S)$$

$$\Rightarrow \quad \mathsf{F.eval}(k_S, x) = \begin{cases} \mathsf{F}(k, x) & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

Constrained PRFs [BW13]

constrained PRF

$\mathsf{F}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a constrained PRF for set system $\mathcal{S} \subseteq 2^{\mathcal{X}}$ if there is

- constrained key space $\mathcal{K}_c$
- $\mathsf{F.cstr}\colon \mathcal{K} \times \mathcal{S} \to \mathcal{K}_c$
- $\mathsf{F.eval}\colon \mathcal{K}_c \times \mathcal{X} \to \mathcal{Y}$, s.t.

$$k_S \leftarrow \mathsf{F.cstr}(k, S)$$

$$\Rightarrow \quad \mathsf{F.eval}(k_S, x) = \begin{cases} \mathsf{F}(k, x) & \text{if } x \in S \\ \perp & \text{otherwise} \end{cases}$$

$\Rightarrow$ F should look random where one cannot evaluate

Adaptive vs. selective security

Adaptive security experiment

- Challenger chooses $k \leftarrow \mathcal{K}$, $b \leftarrow \{0,1\}$
- A can query $\mathsf{F.cstr}(k,\cdot)$ (and $\mathsf{F}(k,\cdot)$)
- A submits challenge $x^* \in \{0,1\}^n$
- A gets $\begin{cases} \mathsf{F}(k,x^*) & \text{if } b=1 \\ y \leftarrow \mathcal{Y} & \text{if } b=0 \end{cases}$
- A outputs b'

Adaptive vs. selective security

Adaptive security experiment

- Challenger chooses $k \leftarrow \mathcal{K}$, $b \leftarrow \{0,1\}$
- A can query F.cstr$(k, \cdot)$ (and F$(k, \cdot)$)
- A submits challenge $x^* \in \{0,1\}^n$
- A gets $\begin{cases} \mathsf{F}(k, x^*) & \text{if } b = 1 \\ y \leftarrow \mathcal{Y} & \text{if } b = 0 \end{cases}$
- A outputs b'

A must choose $x^* \notin \bigcup S_i$, $\{S_i\}$ queried to F.cstr

Adaptive vs. selective security

Adaptive security experiment

- Challenger chooses $k \leftarrow \mathcal{K}$, $b \leftarrow \{0,1\}$
- A can query $\mathsf{F.cstr}(k,\cdot)$ (and $\mathsf{F}(k,\cdot)$)
- A submits challenge $x^* \in \{0,1\}^n$
- A gets $\begin{cases} \mathsf{F}(k,x^*) & \text{if } b=1 \\ y \leftarrow \mathcal{Y} & \text{if } b=0 \end{cases}$
- A outputs b'

F is (ϵ, s, q)-secure if

for all q-query A, $|\mathsf{A}| \leq s$: $|\Pr[b'=b] - \frac{1}{2}| \leq \epsilon$

Adaptive vs. selective security

Selective security experiment

- Challenger chooses $k \leftarrow \mathcal{K}$, $b \leftarrow \{0,1\}$
- A submits challenge $x^* \in \{0,1\}^n$
- A can query $\mathsf{F.cstr}(k,\cdot)$ (and $\mathsf{F}(k,\cdot)$)
- A gets $\begin{cases} \mathsf{F}(k,x^*) & \text{if } b=1 \\ y \leftarrow \mathcal{Y} & \text{if } b=0 \end{cases}$
- A outputs b'

F is (ϵ, s, q)-selectively secure if

for all q-query A, $|\mathsf{A}| \leq s$: $|\Pr[b'=b] - \frac{1}{2}| \leq \epsilon$

Complexity leveraging

Lemma

If A *breaks adaptive security with advantage* $\epsilon \Rightarrow$
can use A *to break selective security with advantage* $\epsilon/2^n$.

Complexity leveraging

Lemma

If A *breaks adaptive security with advantage* $\epsilon \Rightarrow$
can use A *to break selective security with advantage* $\epsilon/2^n$.

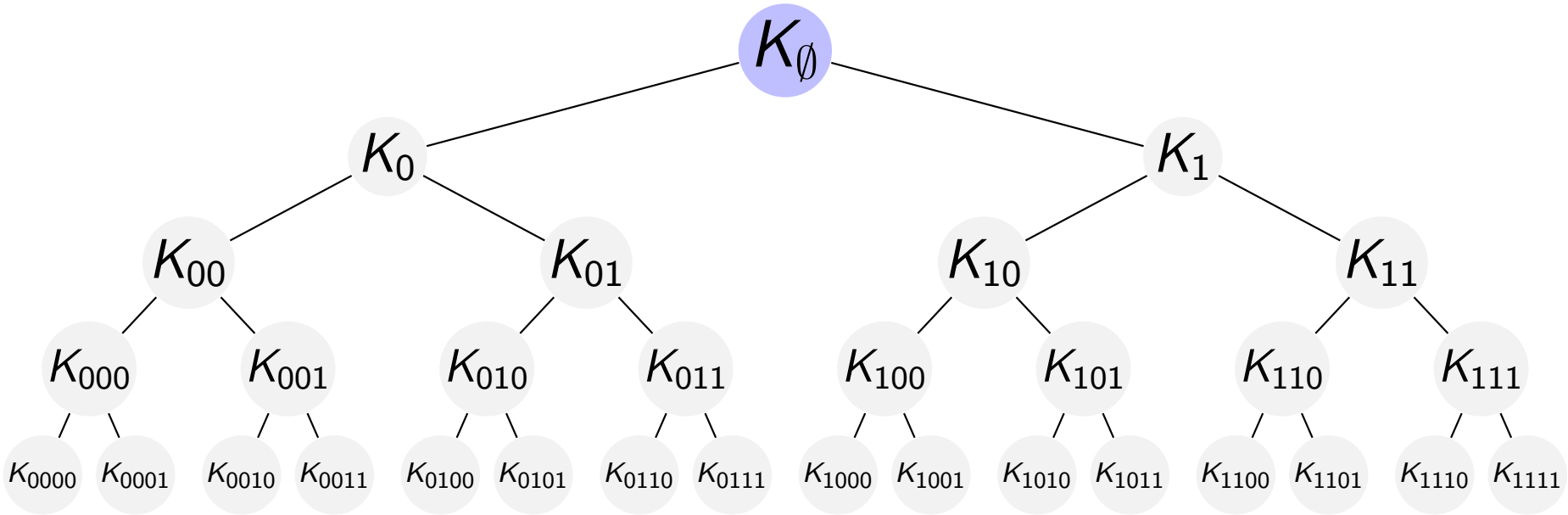
Selective attack using adaptive A

- Guess random challenge $x' \leftarrow \{0,1\}^n$
- Forward A's queries to F.cstr$(\cdot)$
- A submits challenge x^*
- If $x' \neq x^*$, give up
- Else forward y to A, output A's output

Goldreich-Goldwasser-Micali 84

GGM PRF

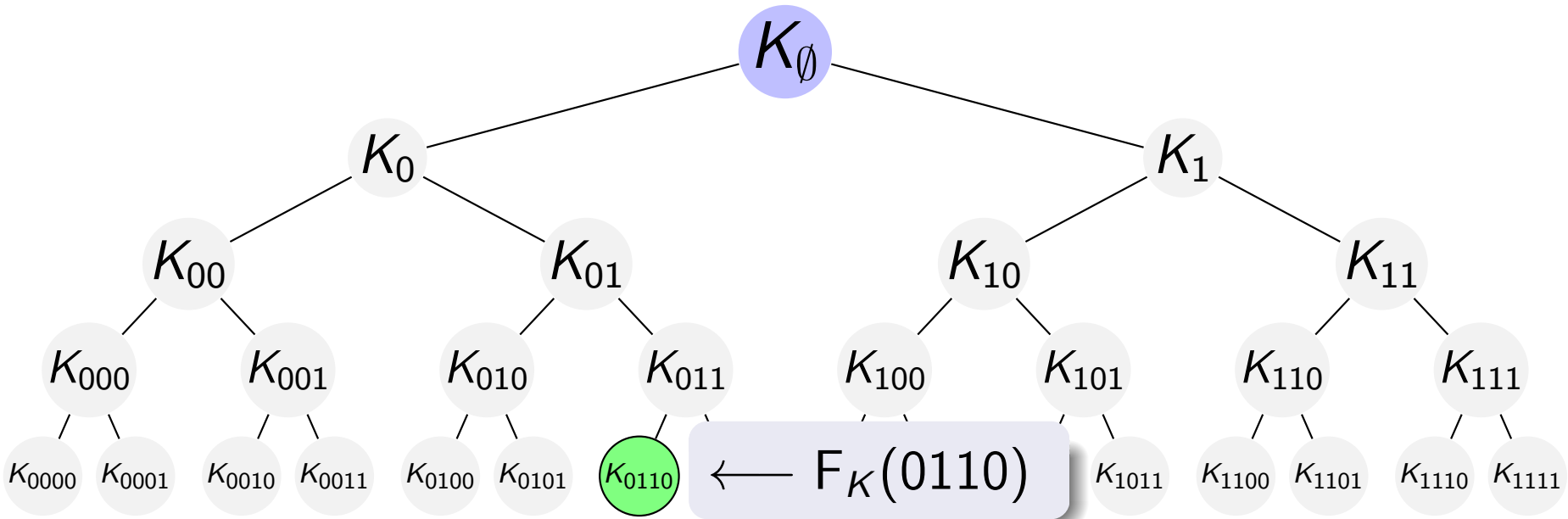
- PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$
- $K = K_\emptyset \leftarrow \{0,1\}^n$
- $K_{x\|0} \| K_{x\|1} = G(K_x)$
- $F_K(x) = K_x$



Goldreich-Goldwasser-Micali 84

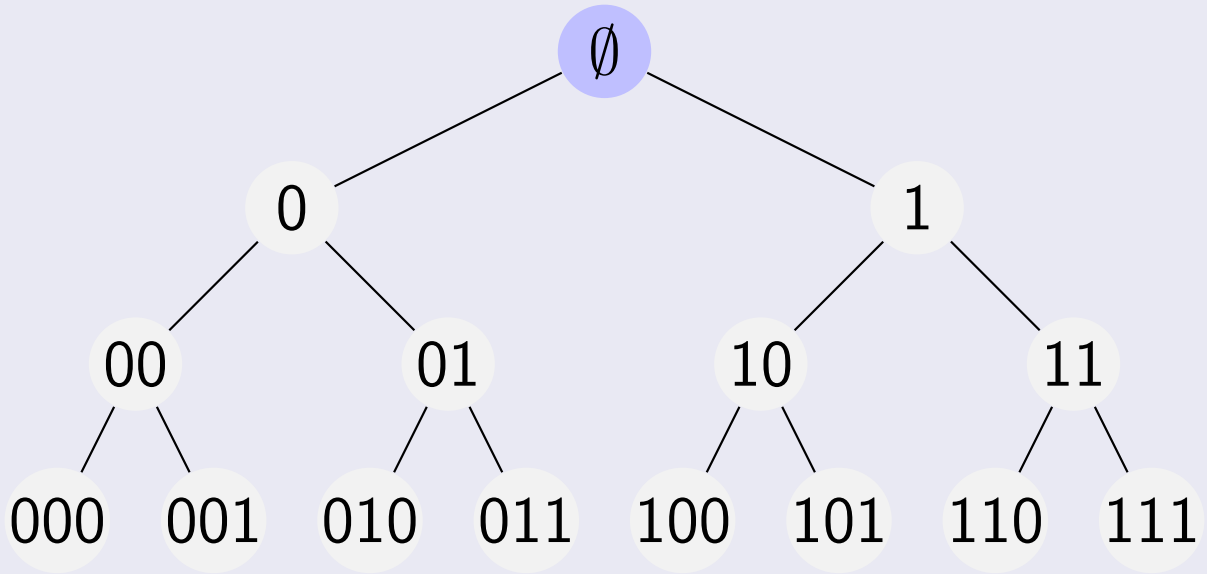
GGM PRF

- PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$
- $K = K_\emptyset \leftarrow \{0,1\}^n$
- $K_{x\|0} \| K_{x\|1} = G(K_x)$
- $F_K(x) = K_x$



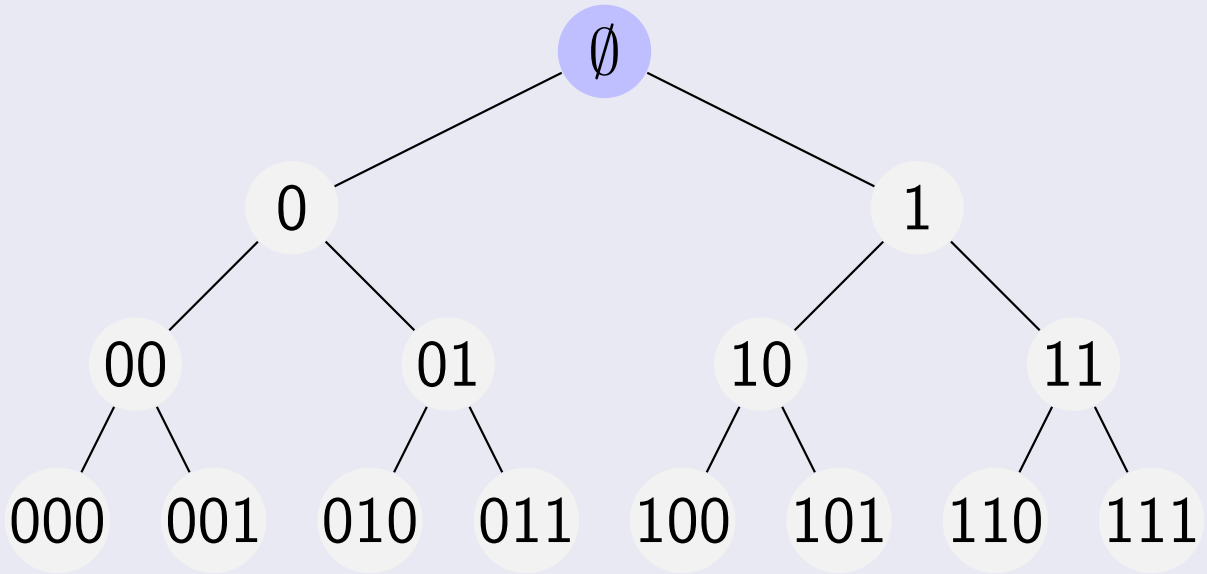
GGM hybrid argument

GGM



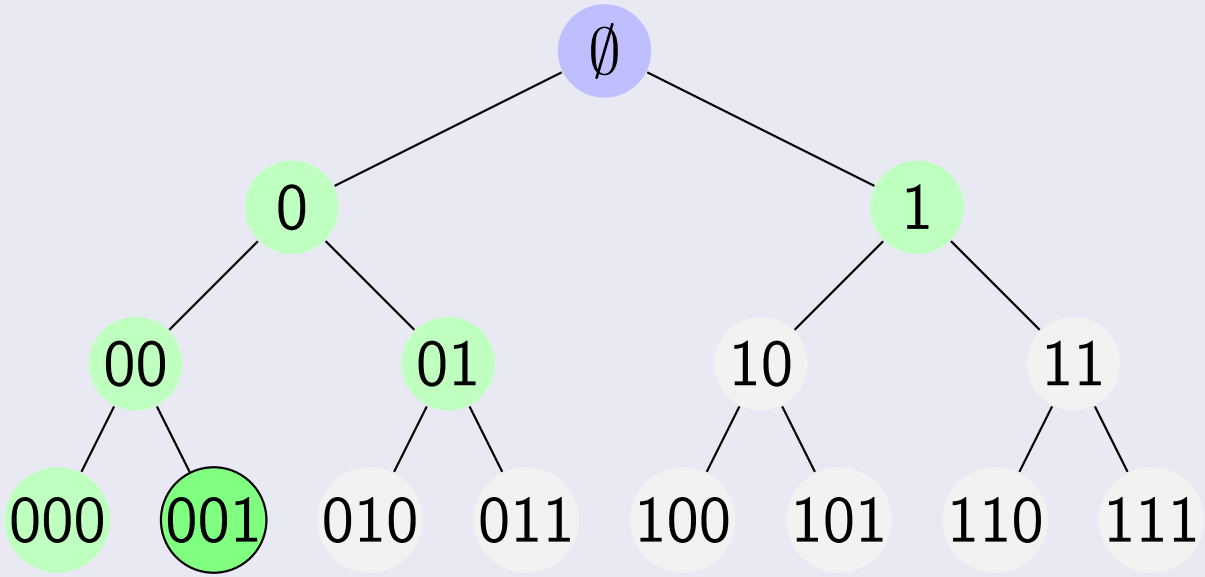
GGM hybrid argument

GGM



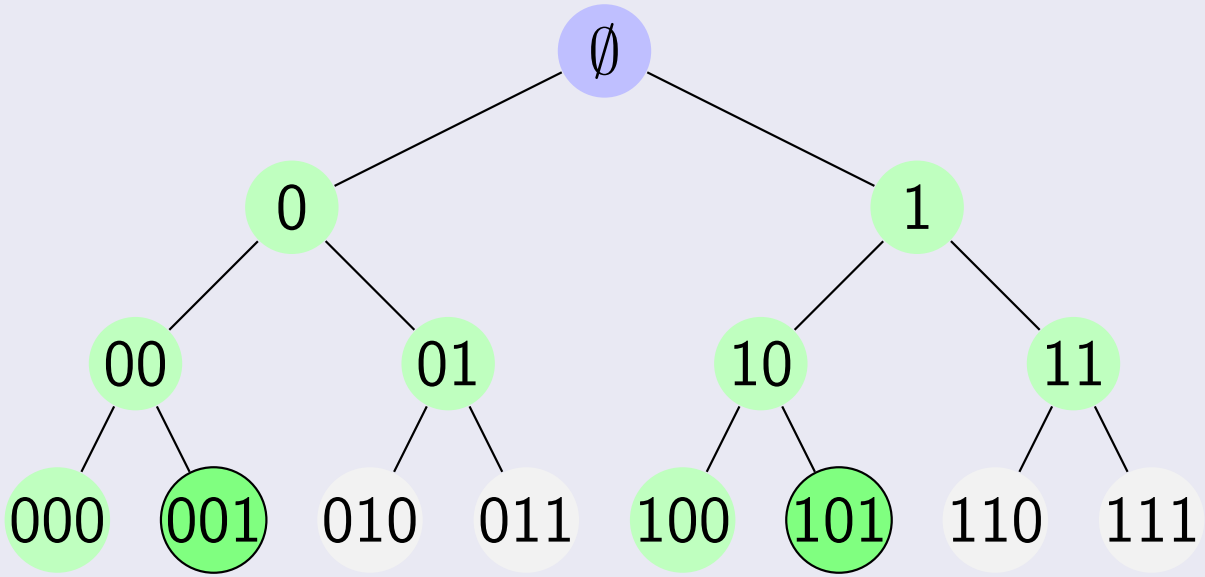
GGM hybrid argument

GGM: evaluation at 001



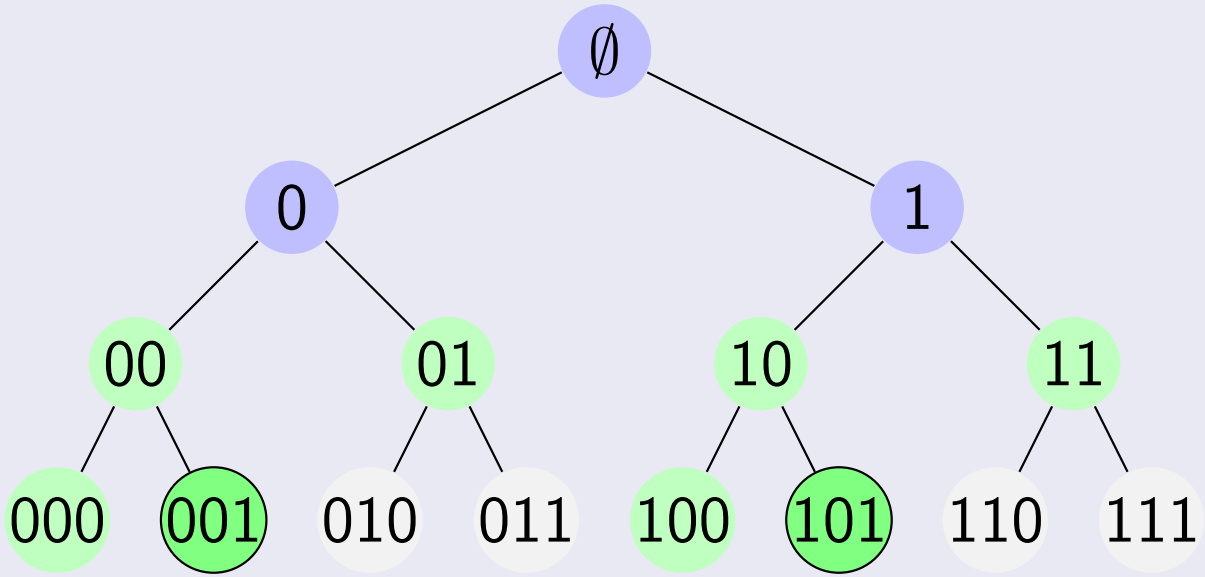
GGM hybrid argument

Hybrid H_0 (the real game)



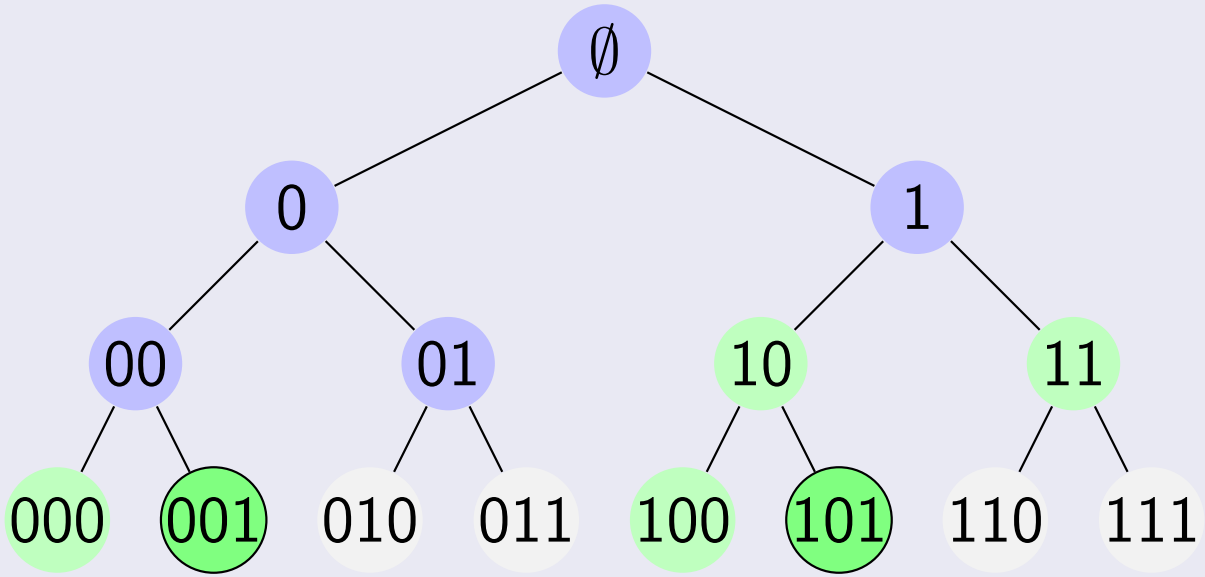
GGM hybrid argument

Hybrid H_1



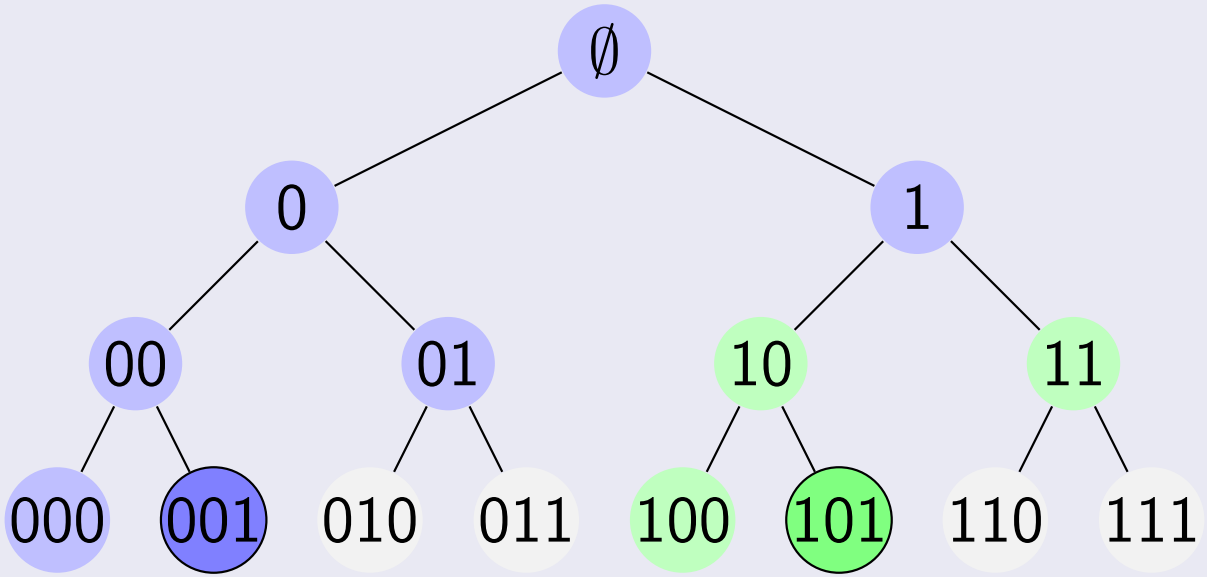
GGM hybrid argument

Hybrid H_2



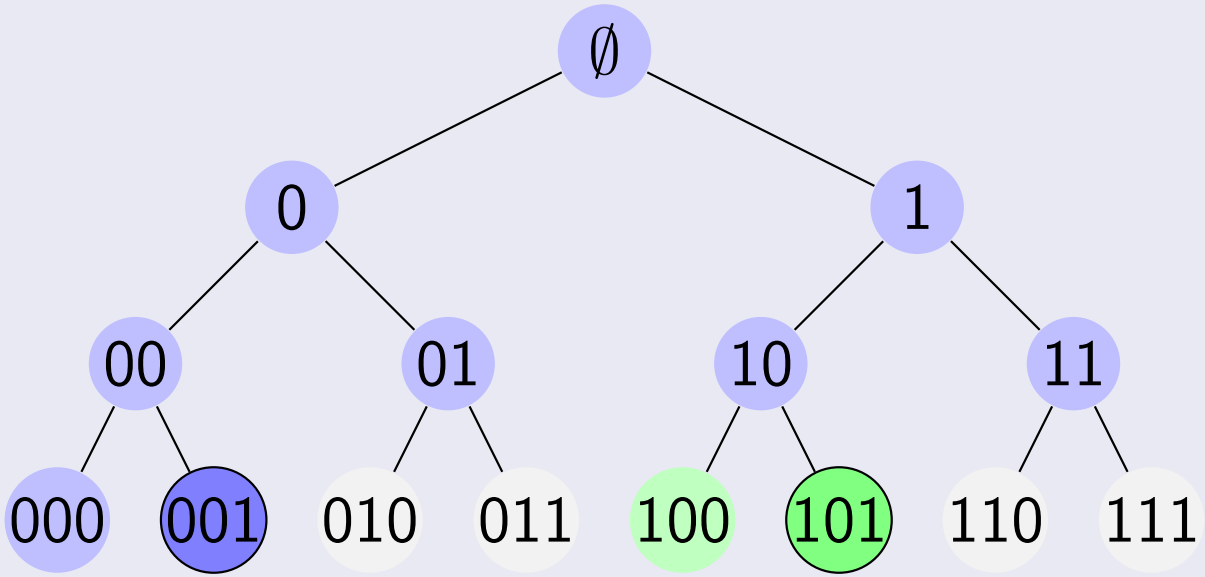
GGM hybrid argument

Hybrid H_3



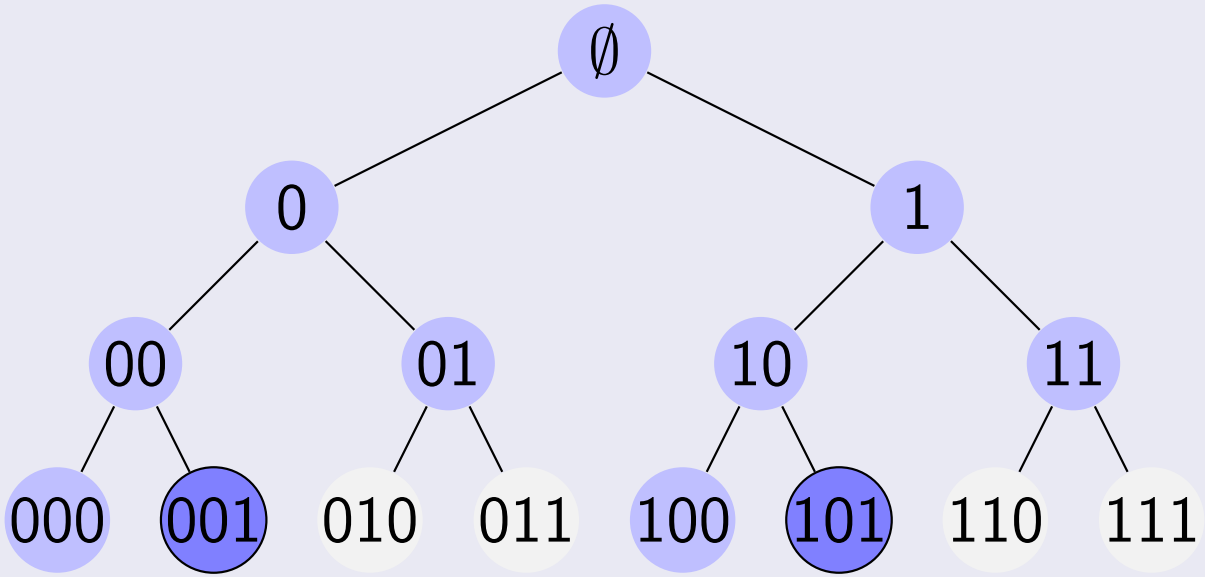
GGM hybrid argument

Hybrid H_4



GGM hybrid argument

Hybrid H_5 (the random game)



- $\mathrm{Adv}(H_0, H_{qn}) = \epsilon$ $\quad q = \#\text{queries}$, n=input length.

$\Rightarrow$ $\mathrm{Adv}(H_i, H_{i+1}) \geq \epsilon/qn$ for some i

$\Rightarrow$ break G with adv. $\geq \epsilon/qn$

GGM as constrained PRF

[1] Boneh, Waters: *Constrained Pseudorandom Functions and Their Applications.* Asiacrypt 2013

[2] Kiayias, Papadopoulos, Triandopoulos, Zacharias: *Delegatable Pseudorandom Functions and Applications.* CCS 2013

[3] Boyle, Goldwasser, Ivan: *Functional Signatures and Pseudorandom Functions.* PKC'14

- GGM is prefix-constrained PRF
- ie constrained PRF for set system

$$\mathcal{S} = \left\{ S_p \mid p \in \{0,1\}^{\leq n} \right\}$$

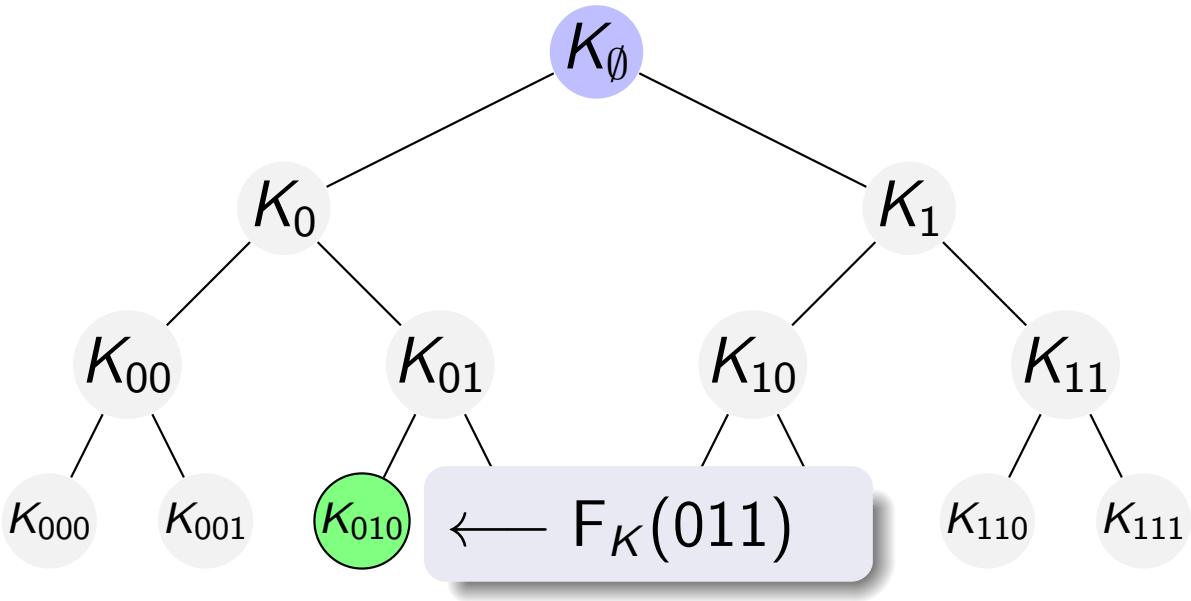
- with $S_p = \left\{ p \| z \;\middle|\; z \in \{0,1\}^{n-|p|} \right\}$

GGM as constrained PRF

[1] Boneh, Waters: *Constrained Pseudorandom Functions and Their Applications.* Asiacrypt 2013

[2] Kiayias, Papadopoulos, Triandopoulos, Zacharias: *Delegatable Pseudorandom Functions and Applications.* CCS 2013

[3] Boyle, Goldwasser, Ivan: *Functional Signatures and Pseudorandom Functions.* PKC'14

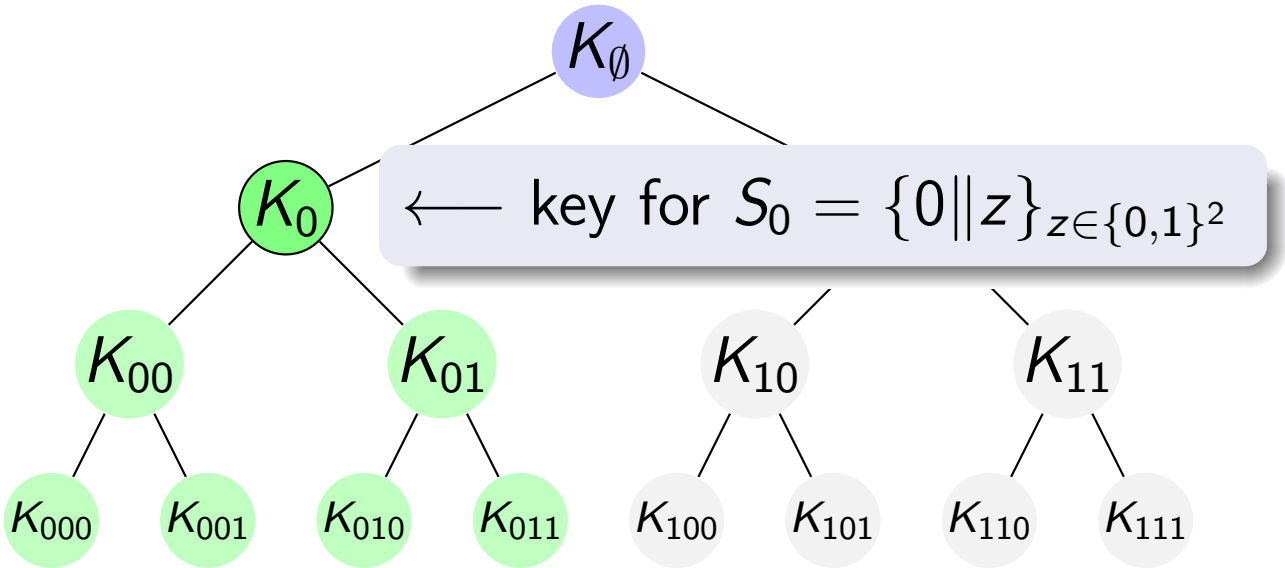


GGM as constrained PRF

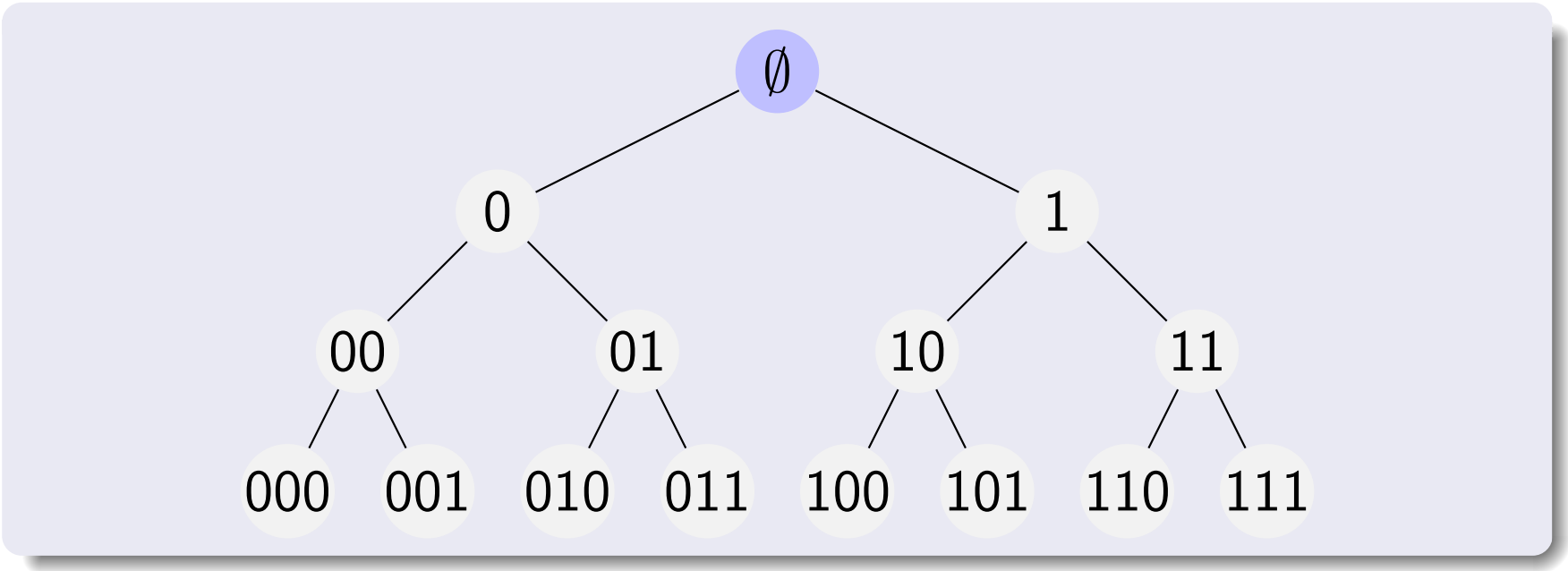
[1] Boneh, Waters: *Constrained Pseudorandom Functions and Their Applications.* Asiacrypt 2013

[2] Kiayias, Papadopoulos, Triandopoulos, Zacharias: *Delegatable Pseudorandom Functions and Applications.* CCS 2013

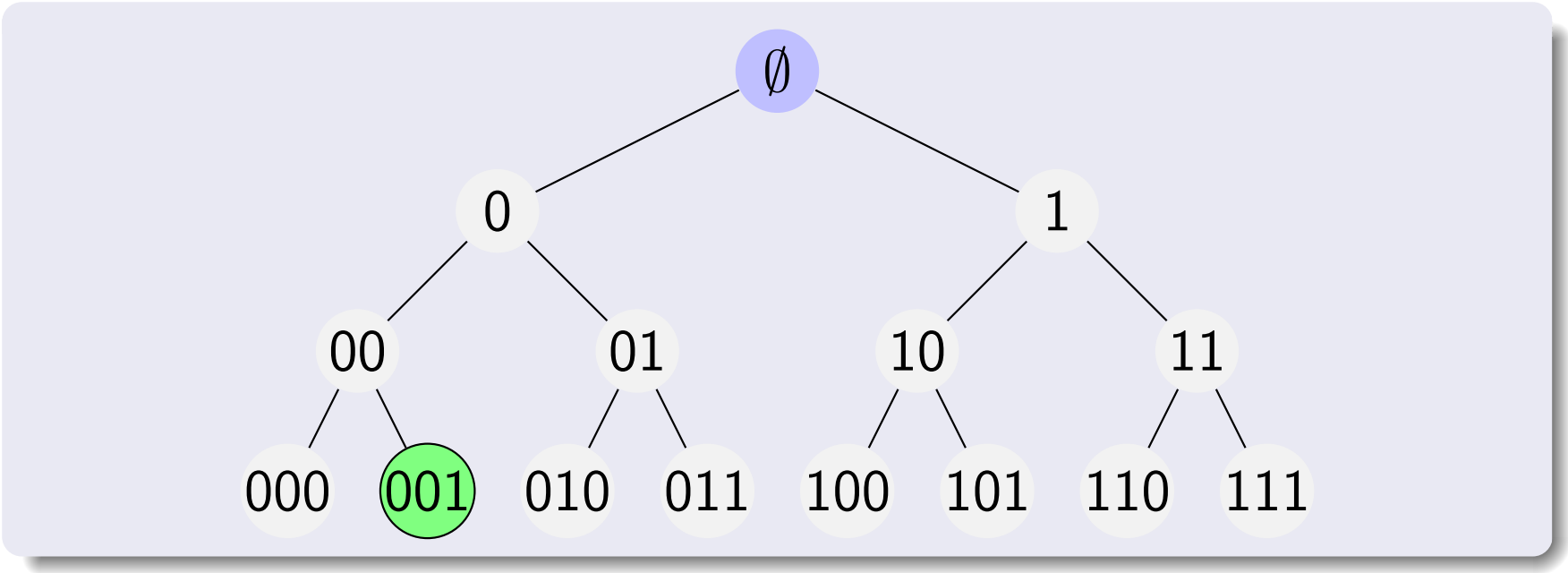
[3] Boyle, Goldwasser, Ivan: *Functional Signatures and Pseudorandom Functions.* PKC'14



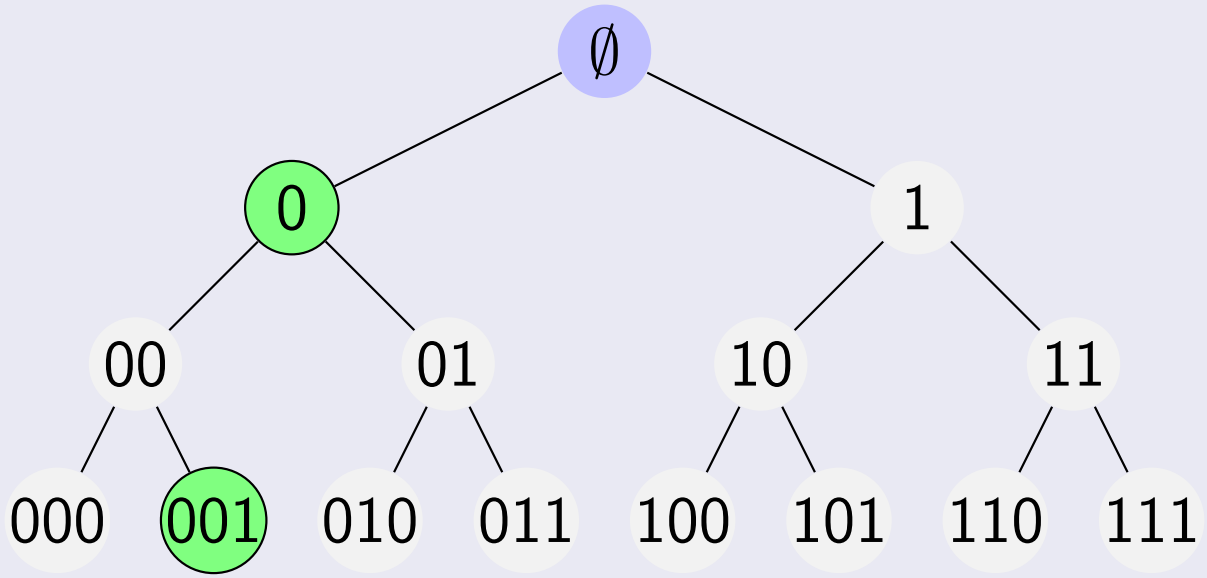
GGM as constrained PRF



GGM as constrained PRF

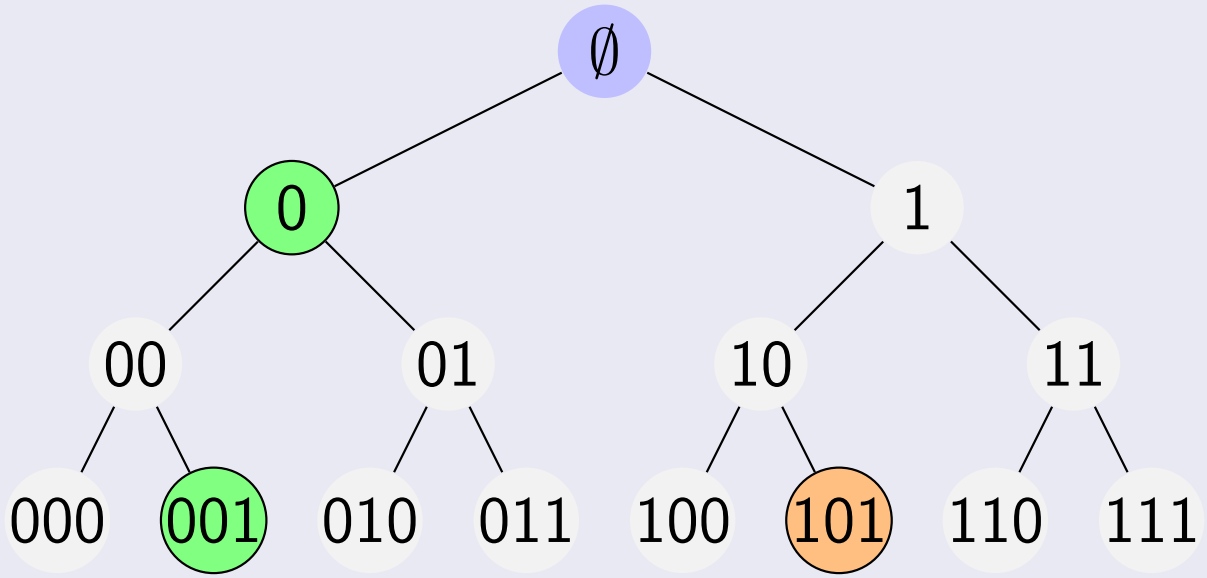


GGM as constrained PRF



$K_{x\|y}$ trivially distinguishable from random given K_x.

GGM as constrained PRF

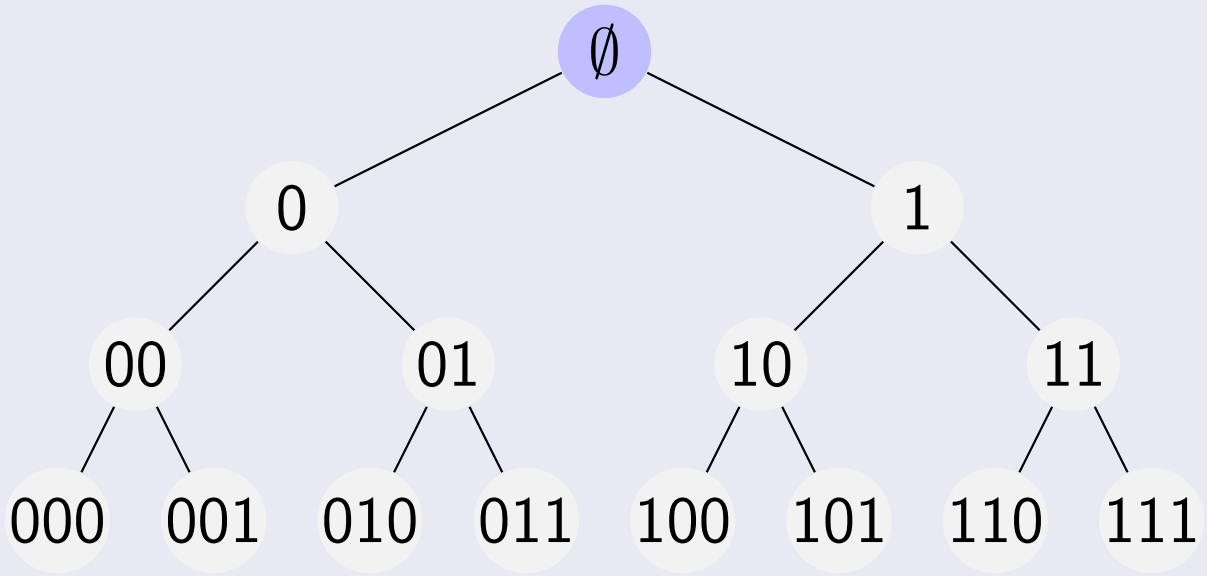


$K_{x\|y}$ trivially distinguishable from random given K_x.

Security game for constrained PRFs

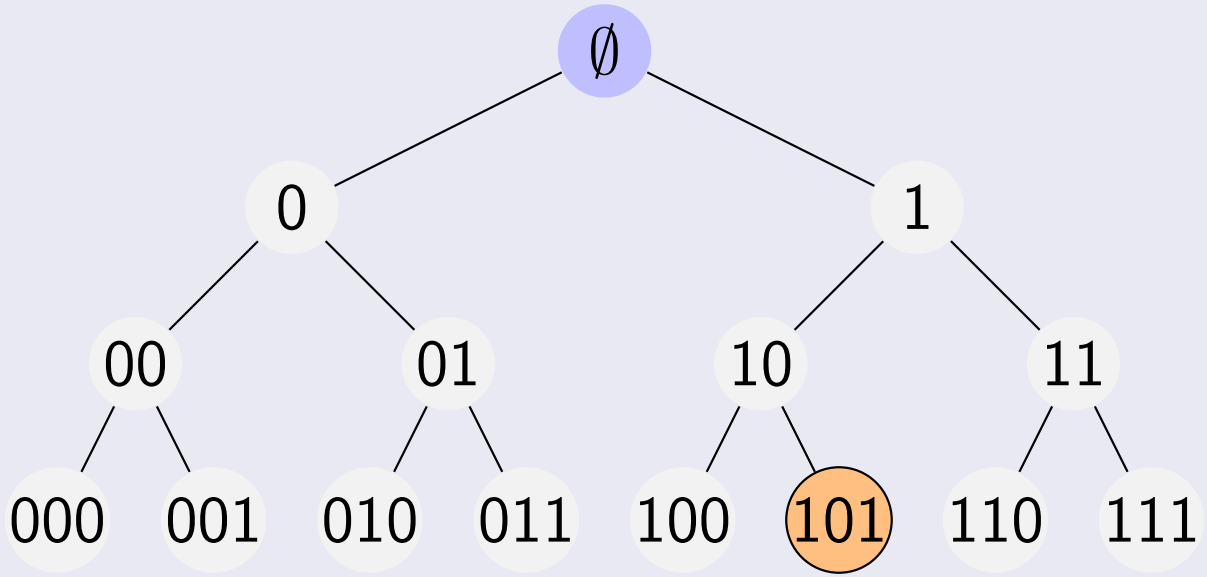
- choose x^* where no prefix of x^* was queried.
- distinguish K_{x^*} from random.

Proving selective security



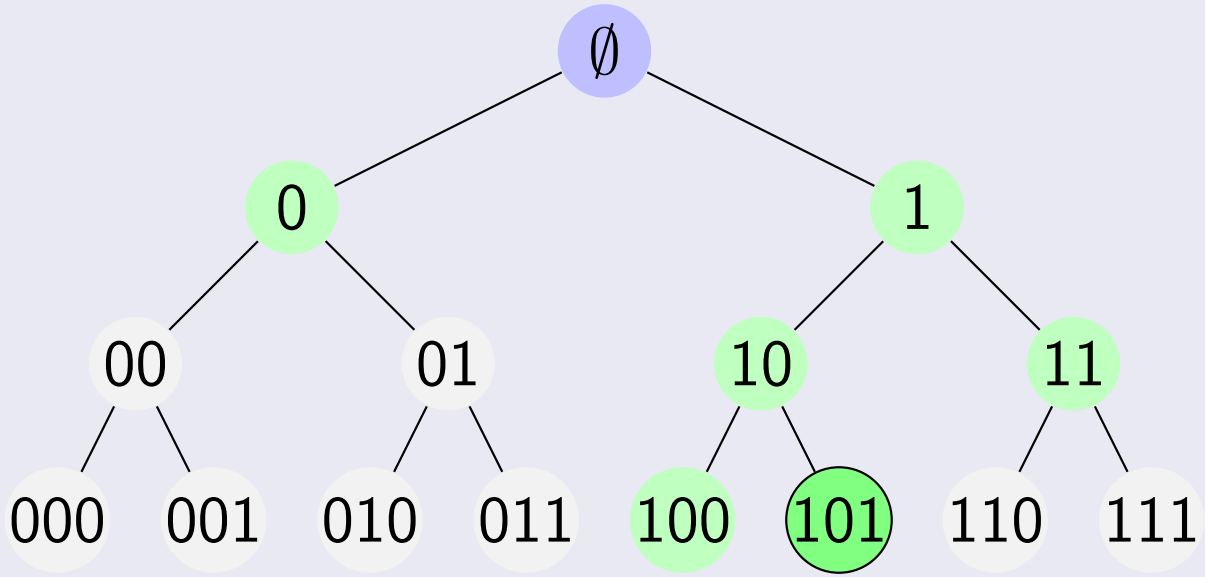
Proving selective security

Submitted challenge



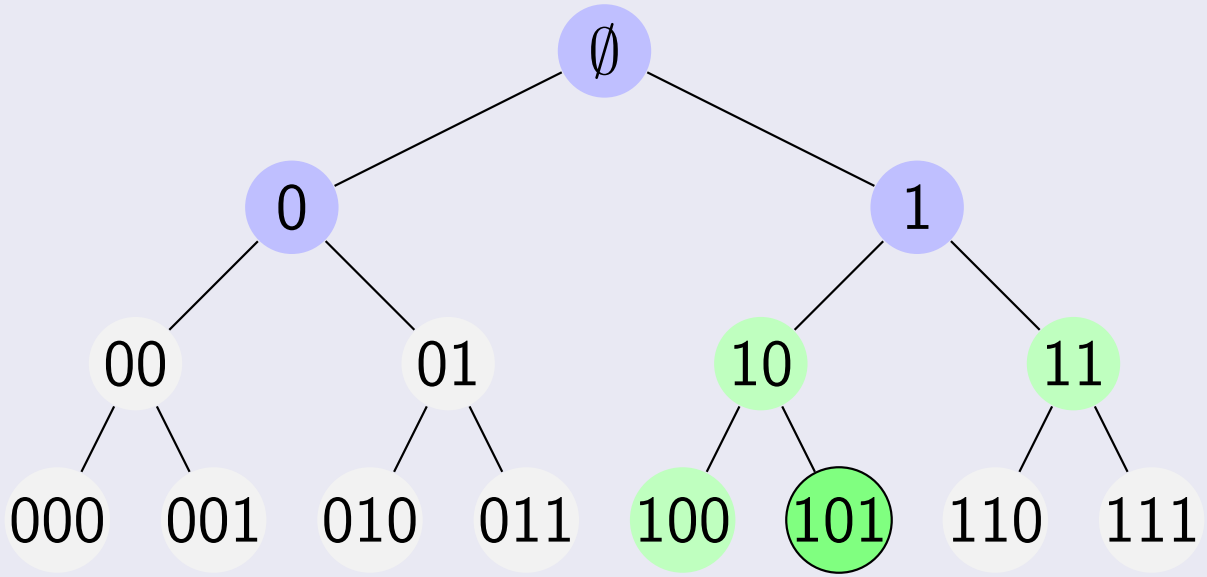
Proving selective security

Hybrid H_0 (real game)



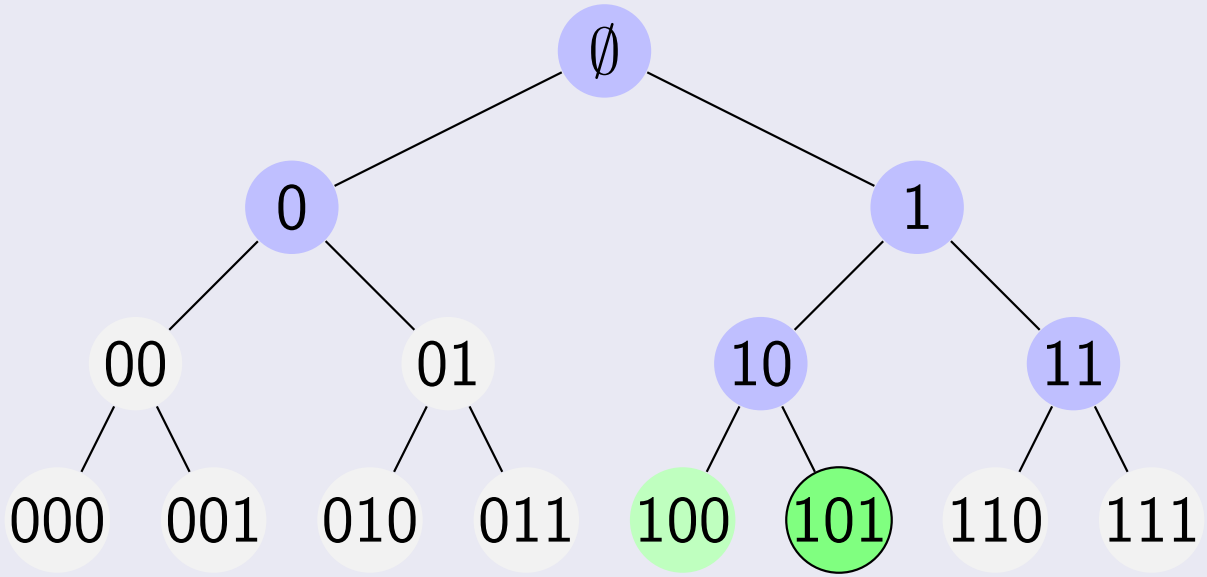
Proving selective security

Hybrid H_1



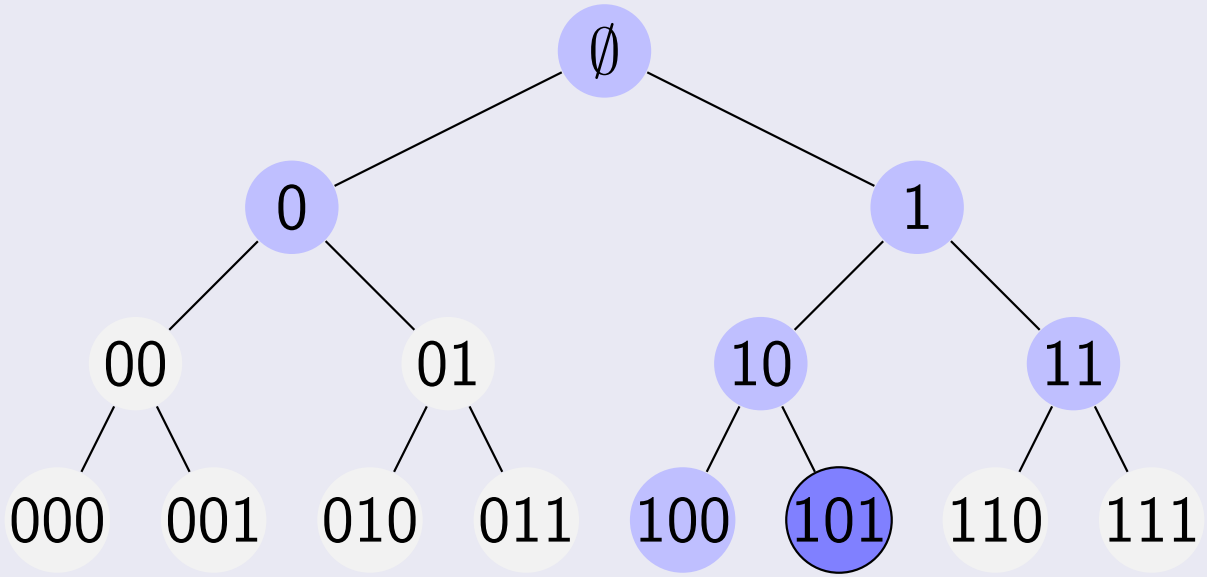
Proving selective security

Hybrid H_2



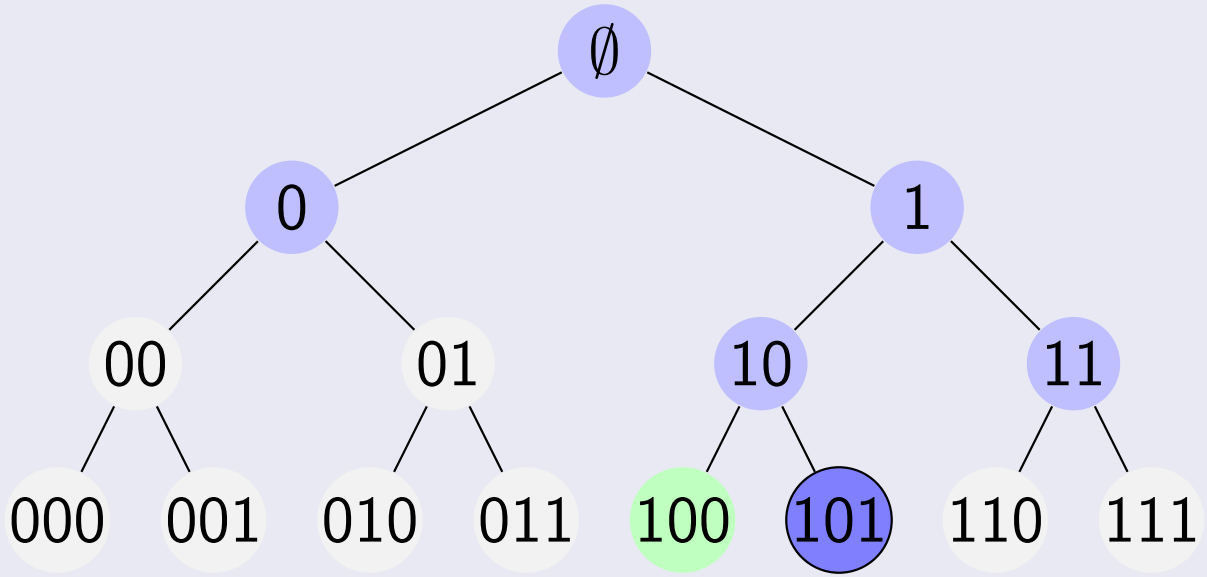
Proving selective security

Hybrid H_3



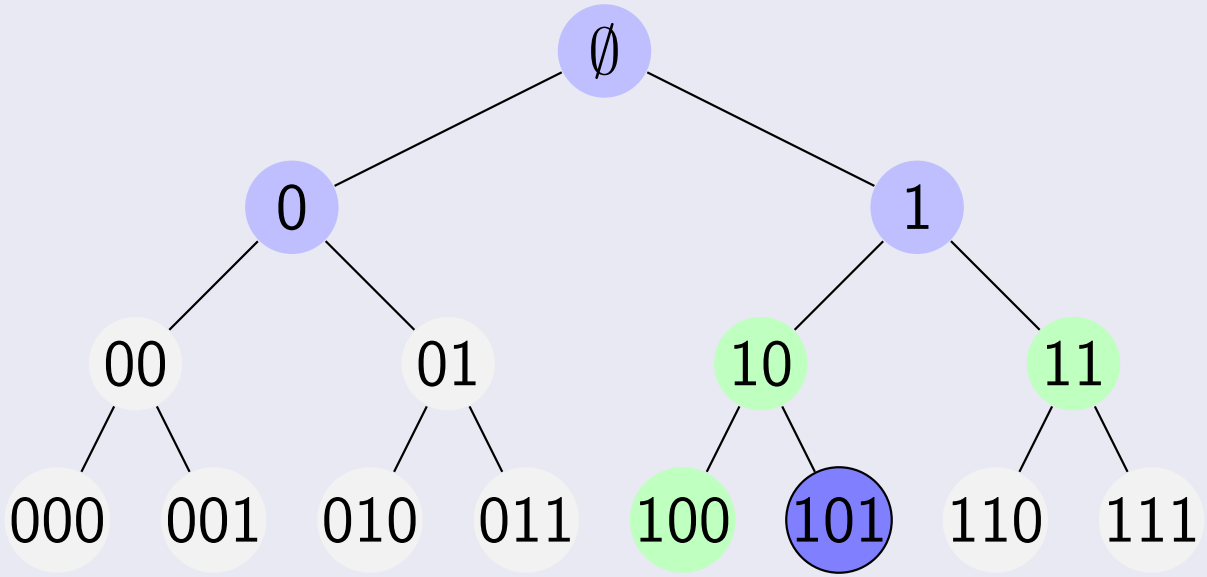
Proving selective security

Hybrid H_4



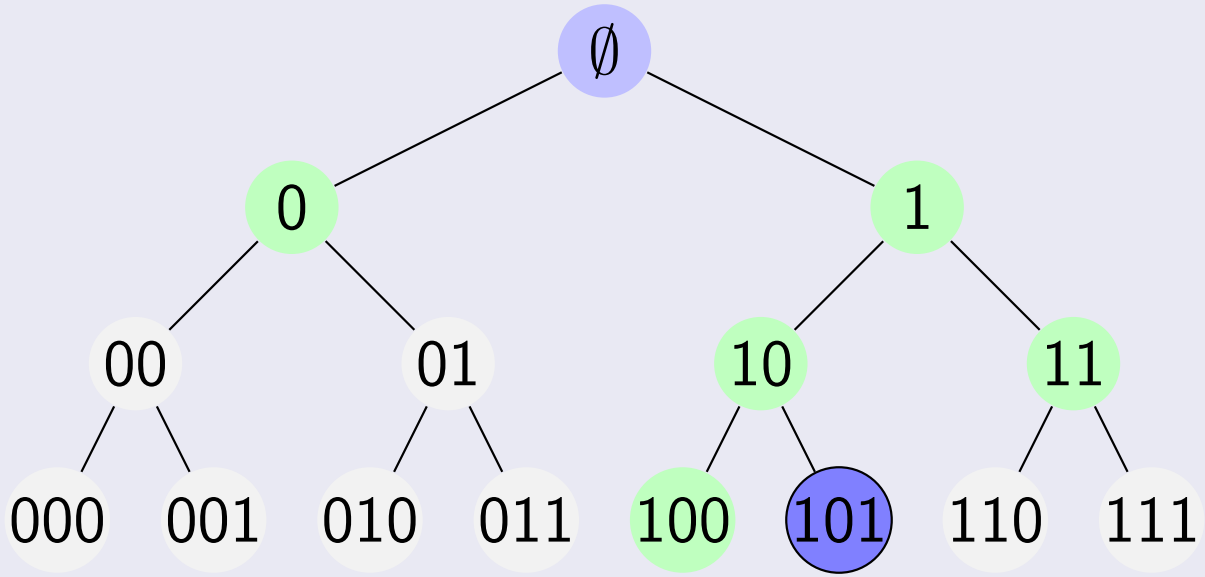
Proving selective security

Hybrid H_5



Proving selective security

Hybrid H_6 (random game)



- $\mathsf{Adv}(H_0, H_{2n}) = \epsilon$

$\Rightarrow$ $\mathsf{Adv}(H_i, H_{i+1}) \geq \epsilon/2n$

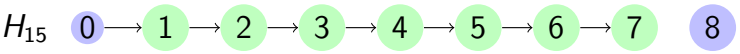
$\Rightarrow$ $\mathsf{Adv}(\mathsf{G}(U_\lambda), U_{2\lambda}) \geq \epsilon/2n$

Proving adaptive security using leveraging

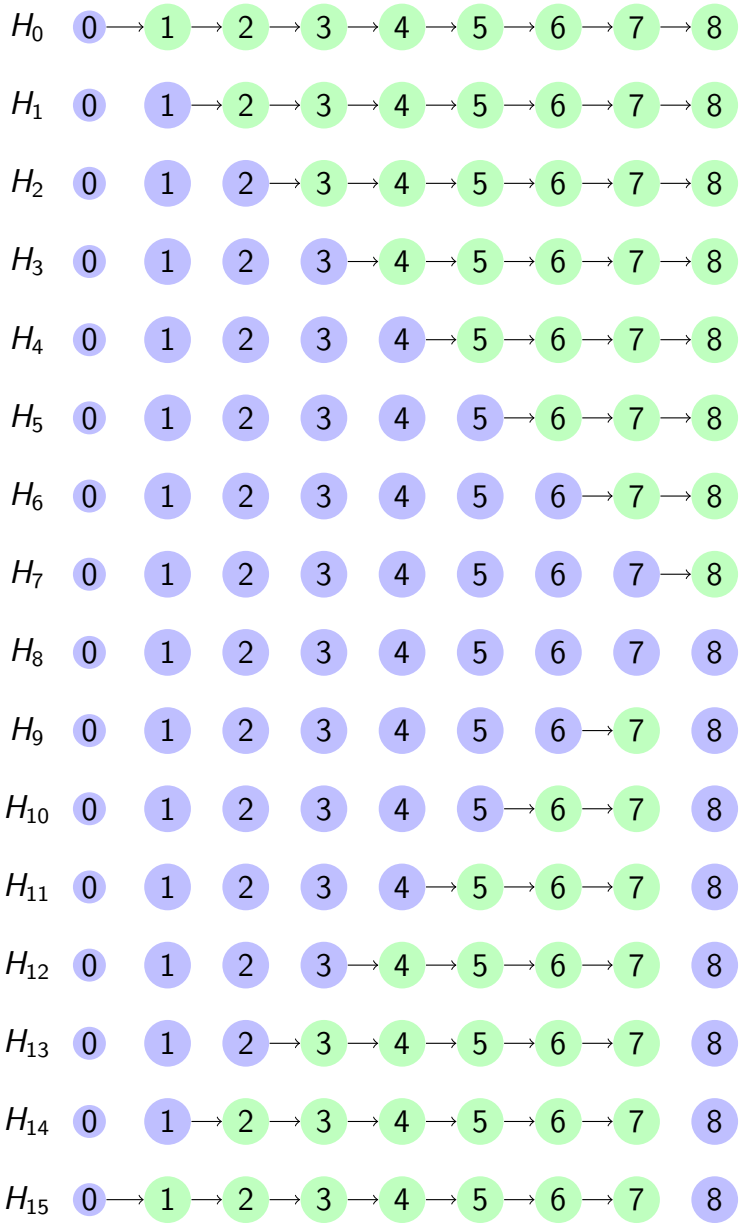
Proof

- Leveraging: guess challenge

$$\epsilon \rightarrow \frac{\epsilon}{2^n}$$



Proving adaptive security using leveraging



Proof

- Leveraging: guess challenge
- Hybrid Argument

$$\epsilon \rightarrow \frac{\epsilon}{2^n} \rightarrow \frac{\epsilon}{2^n \cdot 2n}$$

Nested hybrids in a nutshell

Proving adaptive security via leveraging

1. adaptive $\Pi_n \rightarrow$ selective Π_n (losing factor 2^n).
2. selective $\Pi_n \rightarrow$ G (hybrid argument loses poly(n)).

Nested hybrids in a nutshell

Proving adaptive security via leveraging

1. adaptive $\Pi_n \rightarrow$ selective Π_n (losing factor 2^n).
2. selective $\Pi_n \rightarrow$ G (hybrid argument loses poly(n)).

Nesting hybrids

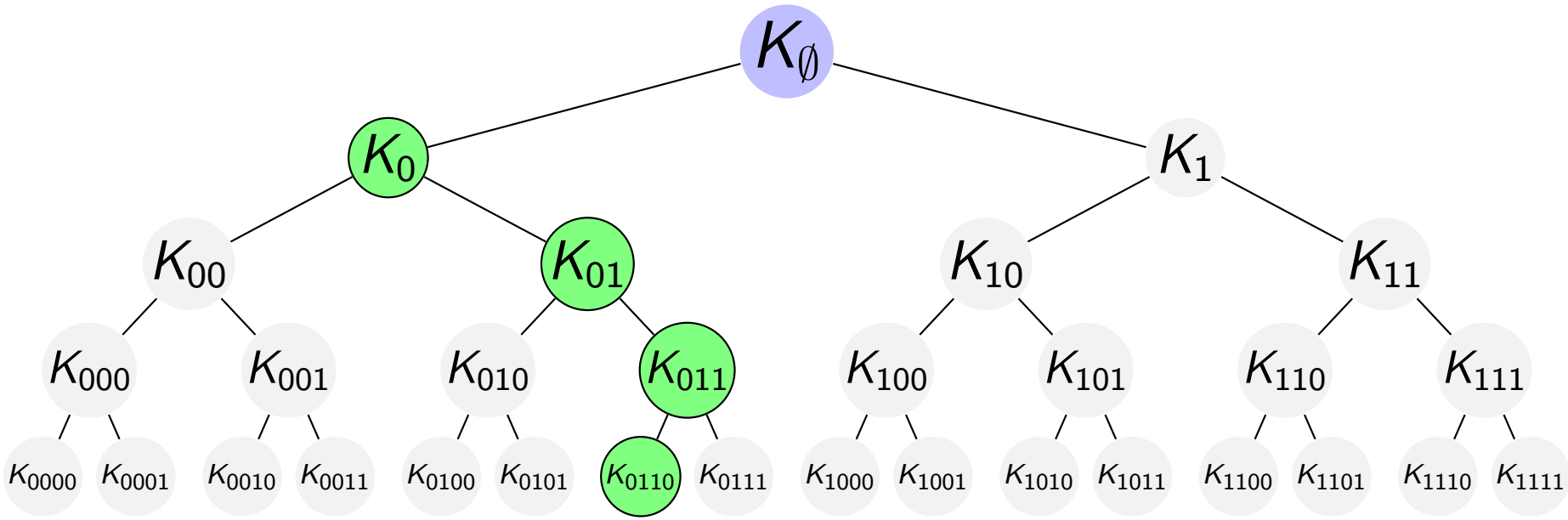
1. adaptive $\Pi_n \rightarrow$ adaptive* Π_n (losing small factor α)
2. adaptive* $\Pi_n \rightarrow$ adaptive $\Pi_{n/2}$ (hybrid losing factor β)
3. iterate steps 1 and 2 $\log(n)$ times:
   adaptive $\Pi_n \rightarrow$ adaptive Π_1 losing $(\alpha\beta)^{\log(n)}$
4. adaptive $\Pi_1 \rightarrow$ G lossless.

Goldreich-Goldwasser-Micali 84

Selective proof strategy

- need to embed random points on path to x^*
- but don't know x^* $\Rightarrow$ could guess x^*

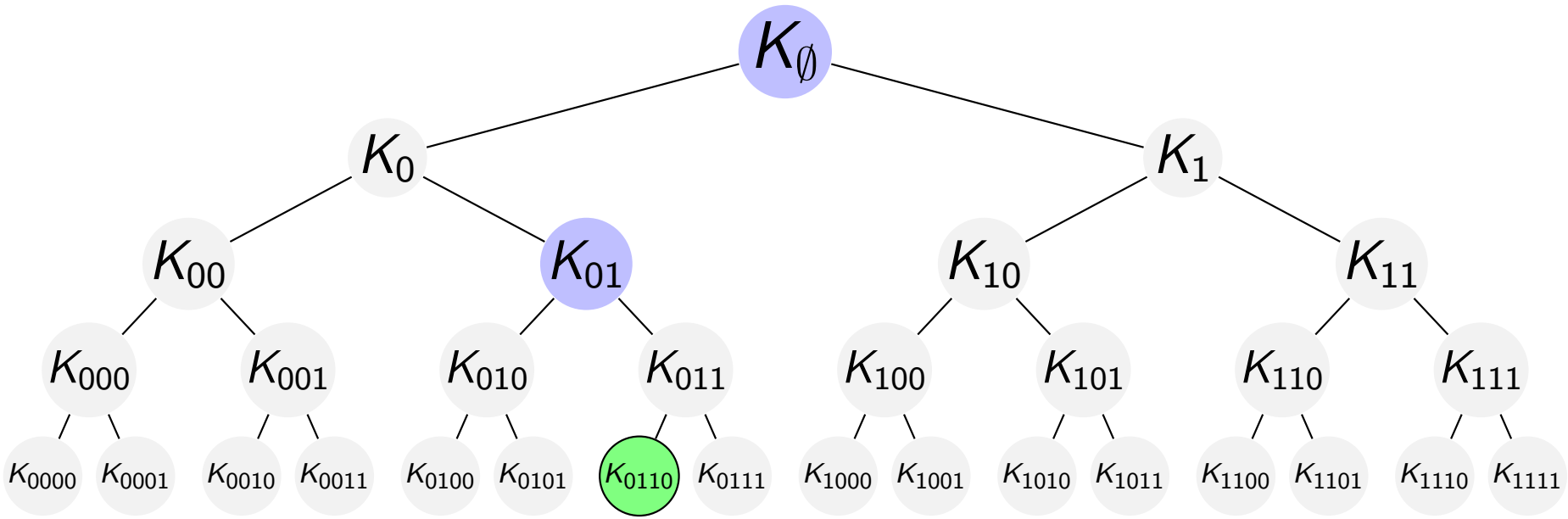
  $\Rightarrow$ lose 2^n



Goldreich-Goldwasser-Micali 84

Halving the game

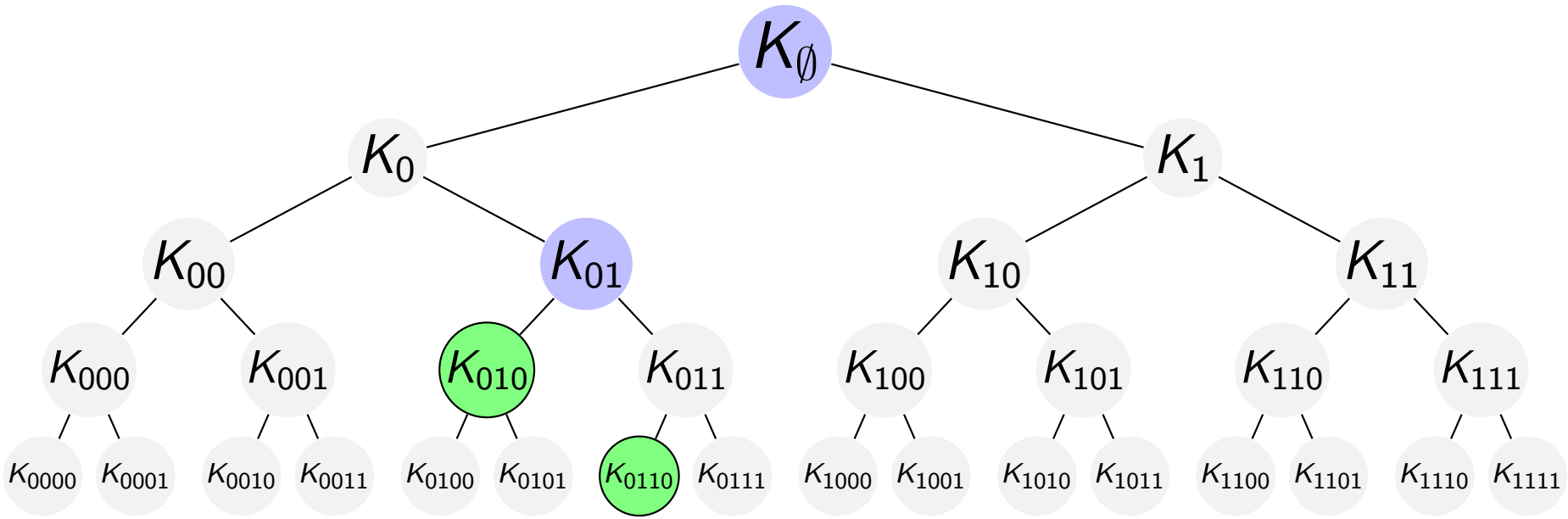
- embed random point on half the way
- wait until x^* is queried



Goldreich-Goldwasser-Micali 84

Halving the game

- embed random point on half the way
- wait until x^* is queried
- problem: what if there was a query before?

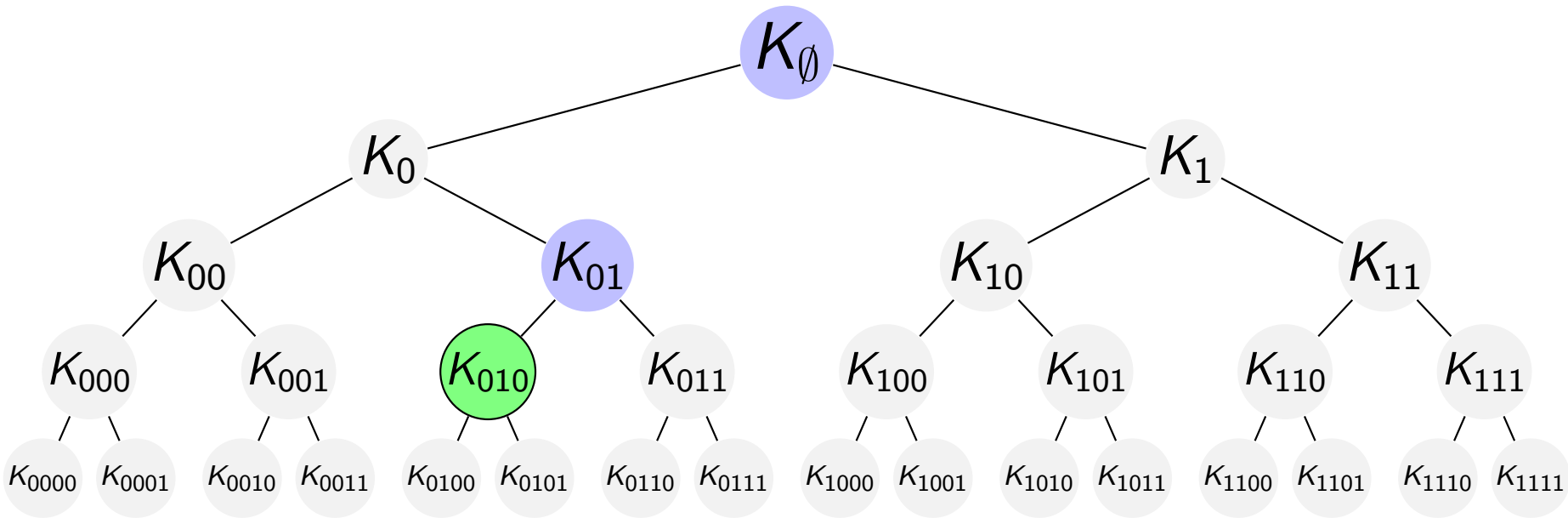


Goldreich-Goldwasser-Micali 84

New proof strategy

- guess which query is first with prefix $x^*[1 \dots n/2]$

  $\Rightarrow$ lose $1/q$

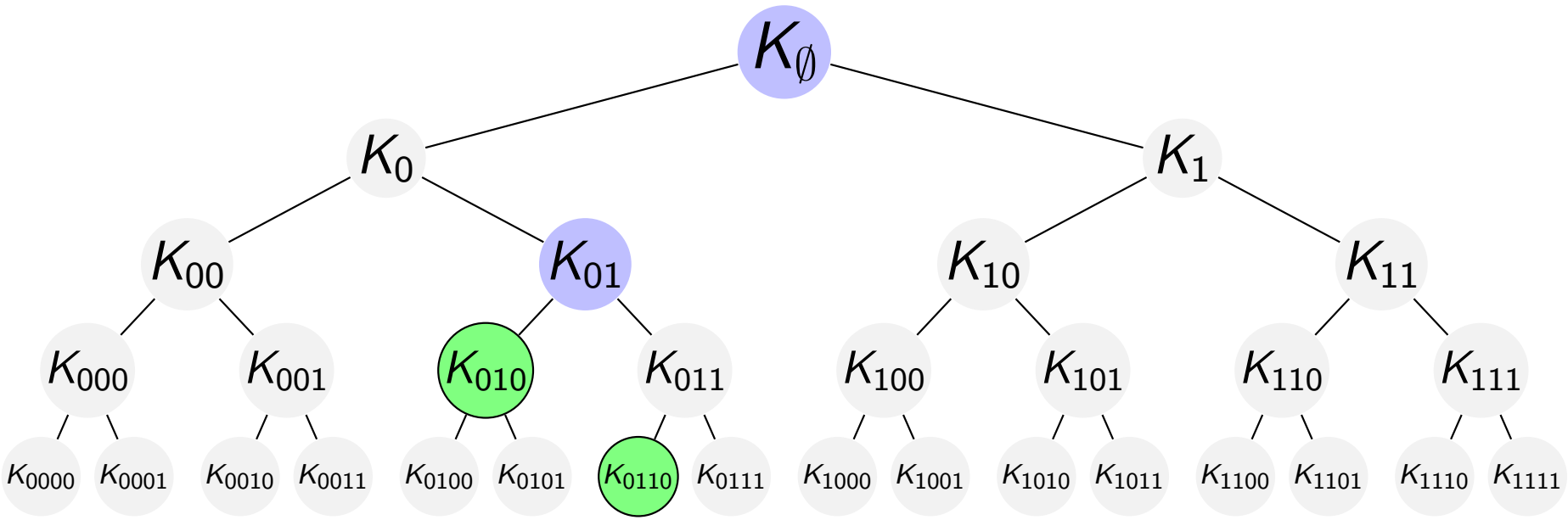


Goldreich-Goldwasser-Micali 84

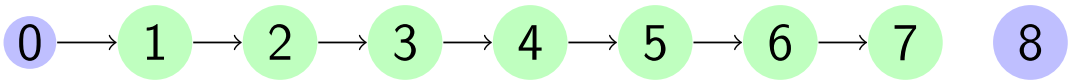
New proof strategy

- guess which query is first with prefix $x^*[1 \dots n/2]$

  $\Rightarrow$ lose $1/q$
- if guess correct, can simulate correctly



Proving adaptive security by nesting

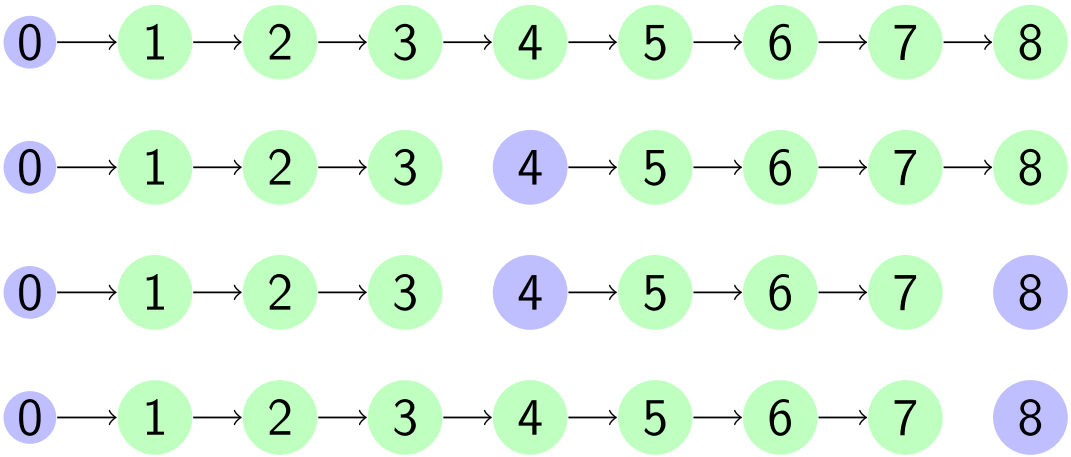


Proof

1. Guess first query that agrees with x^* on 4-prefix.

$$\epsilon \longrightarrow \frac{\epsilon}{q}$$

Proving adaptive security by nesting

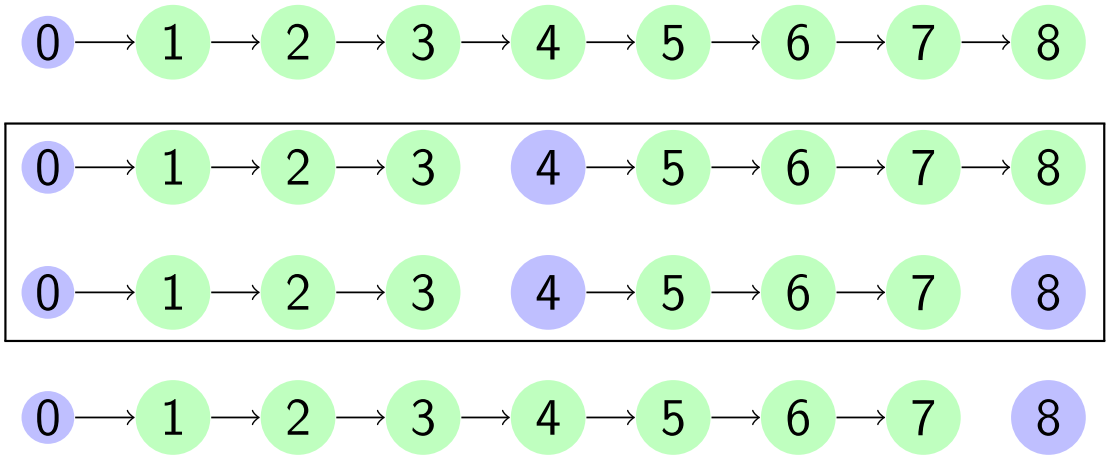


Proof

1. Guess first query that agrees with x^* on 4-prefix.

$$\epsilon \longrightarrow \frac{\epsilon}{q}$$

Proving adaptive security by nesting



Proof

1. Guess first query that agrees with x^* on 4-prefix.
2. Hybrid argument.

$$\epsilon \to \frac{\epsilon}{q} \to \frac{\epsilon}{3q}$$

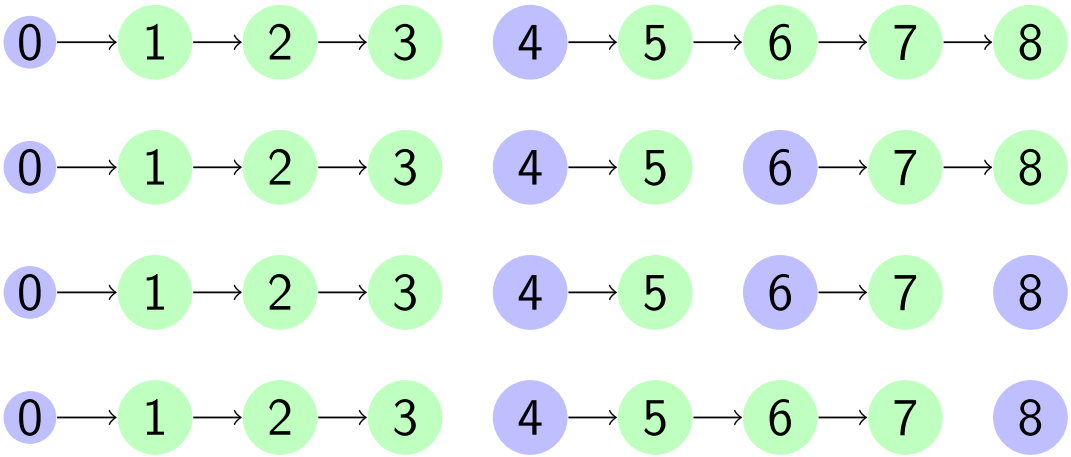
Proving adaptive security by nesting

Proof

1. Guess first query that agrees with x^* on 6-prefix.
2. Hybrid argument.

$$\epsilon \to \frac{\epsilon}{q} \to \frac{\epsilon}{3q} \to \frac{\epsilon}{3q^2}$$

Proving adaptive security by nesting

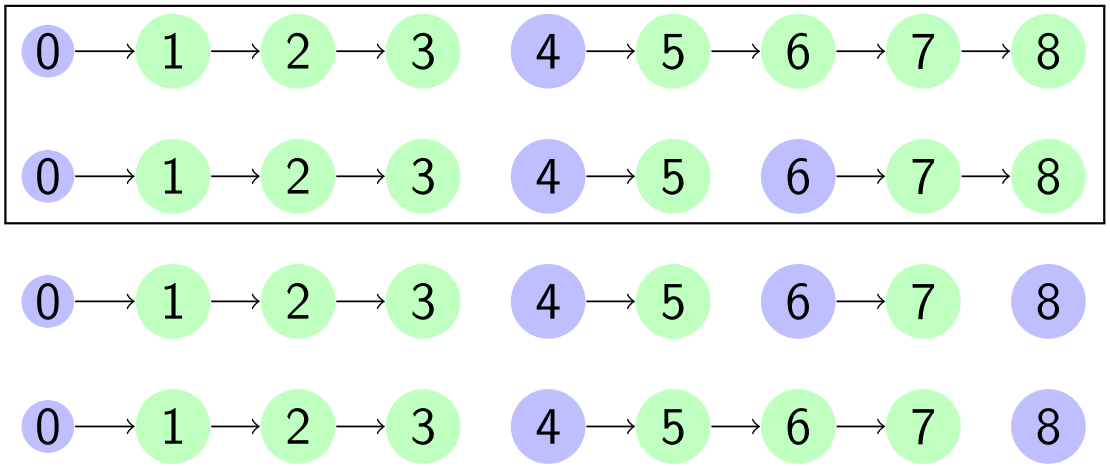


Proof

1. Guess first query that agrees with x^* on 6-prefix.
2. Hybrid argument.

$$\epsilon \to \frac{\epsilon}{q} \to \frac{\epsilon}{3q} \to \frac{\epsilon}{3q^2}$$

Proving adaptive security by nesting



Proof

1. Guess first query that agrees with x^* on 6-prefix.
2. Hybrid argument.

$$\epsilon \to \frac{\epsilon}{q} \to \frac{\epsilon}{3q} \to \frac{\epsilon}{3q^2} \to \frac{\epsilon}{3^2 q^2}$$

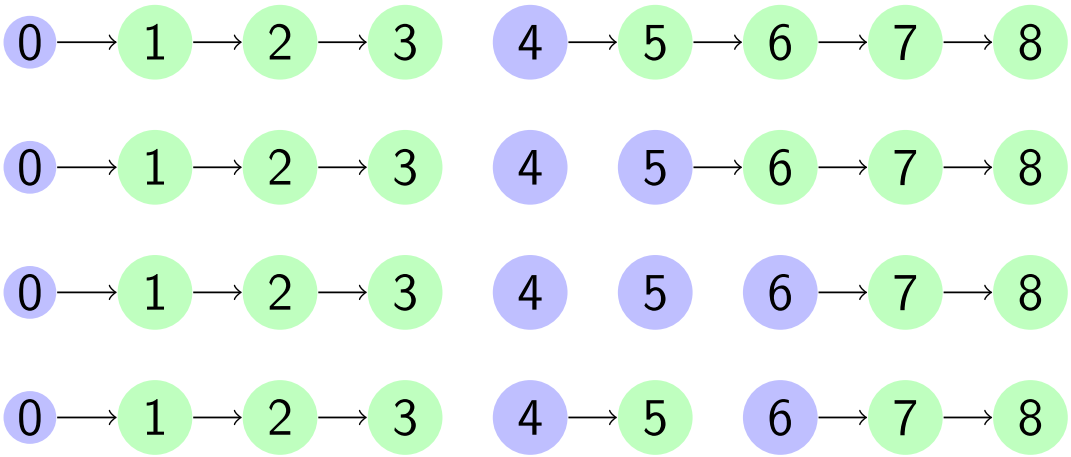
Proving adaptive security by nesting

Proof

1. Guess first query that agrees with x^* on 5-prefix.
2. Hybrid argument.

$$\epsilon \to \frac{\epsilon}{q} \to \frac{\epsilon}{3q} \to \frac{\epsilon}{3q^2} \to \frac{\epsilon}{3^2q^2} \to \frac{\epsilon}{3^2q^3}$$

Proving adaptive security by nesting

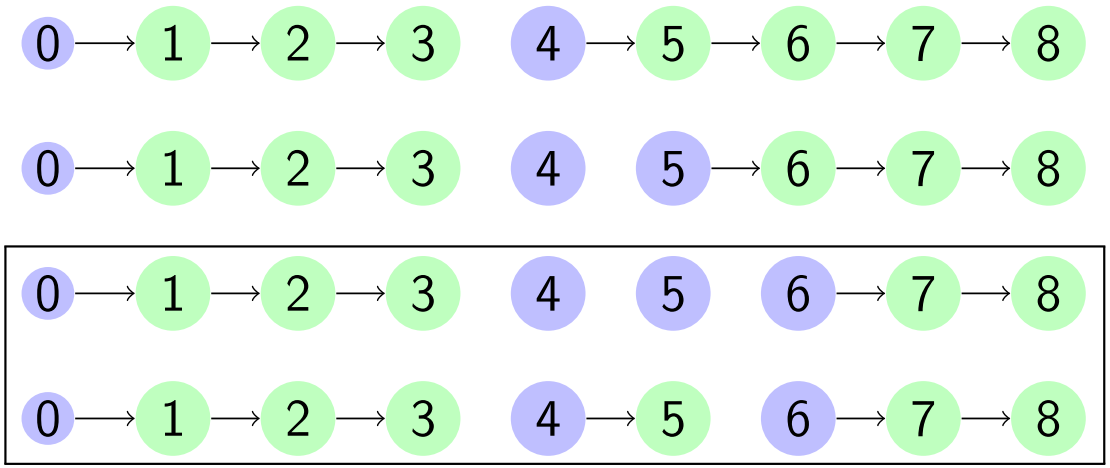


Proof

1. Guess first query that agrees with x^* on 5-prefix.
2. Hybrid argument.

$$\epsilon \to \frac{\epsilon}{q} \to \frac{\epsilon}{3q} \to \frac{\epsilon}{3q^2} \to \frac{\epsilon}{3^2q^2} \to \frac{\epsilon}{3^2q^3}$$

Proving adaptive security by nesting



Proof

1. Guess first query that agrees with x^* on 5-prefix.
2. Hybrid argument.

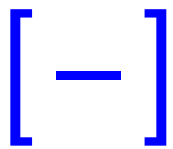
$$\epsilon \to \frac{\epsilon}{q} \to \frac{\epsilon}{3q} \to \frac{\epsilon}{3q^2} \to \frac{\epsilon}{3^2q^2} \to \frac{\epsilon}{3^2q^3} \to \frac{\epsilon}{(3q)^{\log n}}$$

Our result

Theorem

If $\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ *is an* $(\epsilon_\mathsf{G}, s_\mathsf{G})$*-secure PRG then (for any* n, q*)* $\mathsf{GGM}^\mathsf{G}\colon \{0,1\}^n \to \{0,1\}^\lambda$ *is a* **fully** (ϵ, s, q)*-secure constrained PRF for* $\mathcal{S}_{\mathsf{pre}}$*, where*

$$\epsilon = \epsilon_\mathsf{G} \cdot (3q)^{\log n} \qquad s = s_\mathsf{G} - O(q \cdot n \cdot |\mathsf{G}|)$$



The Boneh-Waters bit-fixing PRF

Leveled multilinear maps

- Groups $(\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$, each generated by g_i
- Maps $e_{i,j}$ s.t. $e(g_i^a, g_j^b) = g_{i+j}^{ab}$
- MDDH:

  given $g_1, g_1^{c_1}, \ldots, g_1^{c_{\kappa+1}}$, then $(g_\kappa)^{\prod_{j=1}^{\kappa+1} c_j}$ looks random

The Boneh-Waters bit-fixing PRF

Leveled multilinear maps

- Groups $(\mathbb{G}_1, \ldots, \mathbb{G}_\kappa)$, each generated by g_i
- Maps $e_{i,j}$ s.t. $e(g_i^a, g_j^b) = g_{i+j}^{ab}$
- MDDH:

  given $g_1, g_1^{c_1}, \ldots, g_1^{c_{\kappa+1}}$, then $(g_\kappa)^{\prod_{j=1}^{\kappa+1} c_j}$ looks random

Boneh-Waters PRF

- Key: $\left\{ \begin{array}{ll} \alpha, & d_{1,0}, \ldots, d_{n,0} \\ & d_{1,1}, \ldots, d_{n,1} \end{array} \right\}$
- PRF: $\begin{array}{rl} \mathsf{F}\colon \mathcal{K} \times \{0,1\}^n & \to \mathbb{G}_\kappa = \mathbb{G}_{n+1} \\ (k, x) & \mapsto (g_\kappa)^{\alpha \prod_{i=1}^{n} d_{i,x_i}} \end{array}$

The Boneh-Waters bit-fixing PRF

Boneh-Waters PRF

- Key: $\left\{ \begin{array}{ll} \alpha, & d_{1,0}, \ldots, d_{n,0} \\ & d_{1,1}, \ldots, d_{n,1} \end{array} \right\}$
- PRF: $\begin{array}{rl} \mathsf{F}\colon \mathcal{K} \times \{0,1\}^n & \to \mathbb{G}_\kappa = \mathbb{G}_{n+1} \\ (k, x) & \mapsto (g_\kappa)^{\alpha \prod_{i=1}^{n} d_{i,x_i}} \end{array}$

α	$d_{1,0}$	$d_{2,0}$	$d_{3,0}$	$d_{4,0}$	$d_{5,0}$	$\ldots$	$d_{n,0}$
	$d_{1,1}$	$d_{2,1}$	$d_{3,1}$	$d_{4,1}$	$d_{5,1}$	$\ldots$	$d_{n,1}$

The Boneh-Waters bit-fixing PRF

Boneh-Waters PRF

- Key: $\left\{ \begin{array}{ll} \alpha, & d_{1,0}, \ldots, d_{n,0} \\ & d_{1,1}, \ldots, d_{n,1} \end{array} \right\}$
- PRF: $\begin{array}{rl} \mathsf{F}\colon \mathcal{K} \times \{0,1\}^n & \to \mathbb{G}_\kappa = \mathbb{G}_{n+1} \\ (k, x) & \mapsto (g_\kappa)^{\alpha \prod_{i=1}^{n} d_{i,x_i}} \end{array}$

$$\boxed{\alpha} \quad \boxed{d_{1,0}} \quad d_{2,0} \quad \boxed{d_{3,0}} \quad \boxed{d_{4,0}} \quad d_{5,0} \quad \ldots \quad d_{n,0}$$

$$d_{1,1} \quad \boxed{d_{2,1}} \quad d_{3,1} \quad d_{4,1} \quad \boxed{d_{5,1}} \quad \ldots \quad \boxed{d_{n,1}}$$

$x = (0, 1, 0, 0, 1, \ldots, 1)$

The Boneh-Waters bit-fixing PRF

Boneh-Waters PRF

- Key: $\left\{ \begin{array}{ll} \alpha, & d_{1,0}, \ldots, d_{n,0} \\ & d_{1,1}, \ldots, d_{n,1} \end{array} \right\}$
- PRF: $\begin{array}{rl} \mathsf{F}\colon \mathcal{K} \times \{0,1\}^n & \to \mathbb{G}_\kappa = \mathbb{G}_{n+1} \\ (k, x) & \mapsto (g_\kappa)^{\alpha \prod_{i=1}^{n} d_{i,x_i}} \end{array}$

$\boxed{\alpha}$	$\boxed{d_{1,0}}$	$d_{2,0}$	$\boxed{d_{3,0}}$	$d_{4,0}$	$d_{5,0}$	$\ldots$	$d_{n,0}$
	$d_{1,1}$	$\boxed{d_{2,1}}$	$d_{3,1}$	$d_{4,1}$	$\boxed{d_{5,1}}$	$\ldots$	$\boxed{d_{n,1}}$

$x = (0, 1, 0, ?, 1, \ldots, 1)$ key: $g_{\kappa-1}^{\alpha \prod_{i \neq 4} d_{i,x_i}}$ and $g_1^{d_{4,0}}, g_1^{d_{4,1}}$

The Boneh-Waters bit-fixing PRF

Boneh-Waters PRF

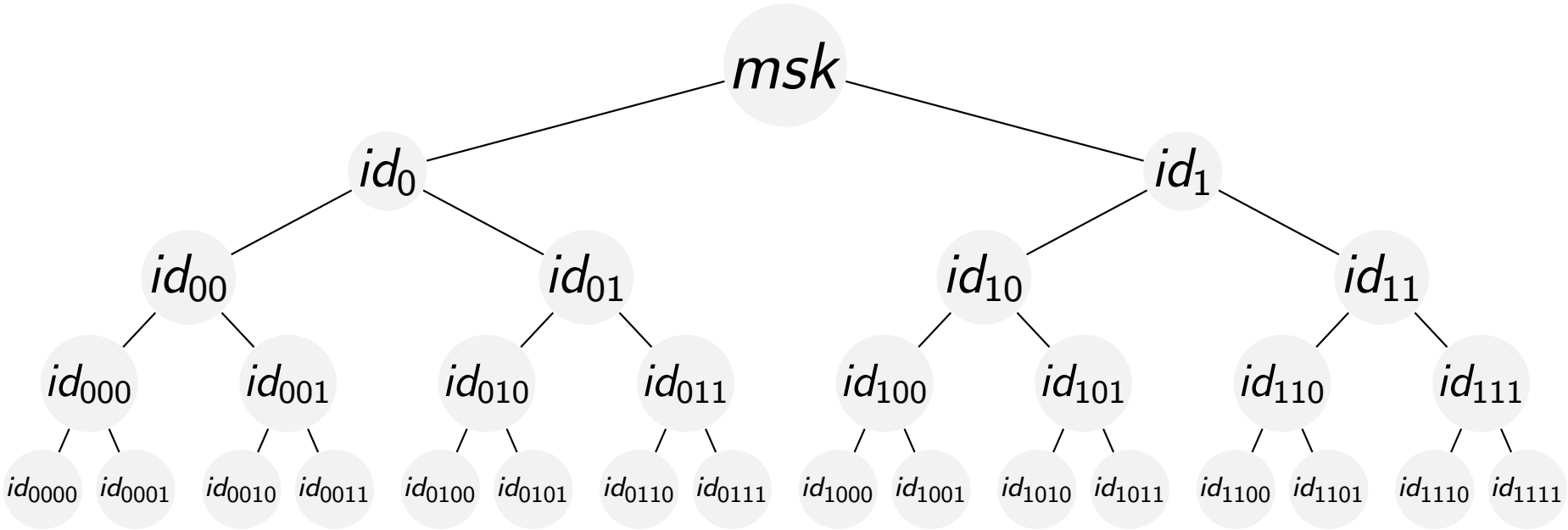
- Key: $\left\{ \begin{array}{ll} \alpha, & d_{1,0}, \ldots, d_{n,0} \\ & d_{1,1}, \ldots, d_{n,1} \end{array} \right\}$
- PRF: $\begin{array}{rl} \mathsf{F}\colon \mathcal{K} \times \{0,1\}^n & \to \mathbb{G}_\kappa = \mathbb{G}_{n+1} \\ (k, x) & \mapsto (g_\kappa)^{\alpha \prod_{i=1}^n d_{i,x_i}} \end{array}$
- Constrained key for set $v \in \{0,1,?\}^n$, set $V := \{i \in [n] \mid v_i \neq ?\}$ and output

$$K_v := (g_{|V|+1})^{\alpha \prod_{i \in V} d_{i,v_i}} \qquad \{D_{i,b} := g^{d_{i,b}}\}_{i \notin V,\, b \in \{0,1\}}$$

- Evaluate F on $x \in S_v$:
  - compute $T := (g_{n-|V|})^{\prod_{i \notin V} d_{i,x_i}}$ by pairing elements D_{i,x_i}
  - output $e(T, K_v) = (g_\kappa)^{\alpha \prod_{i=1}^n d_{i,x_i}} = \mathsf{F}(k, x)$.

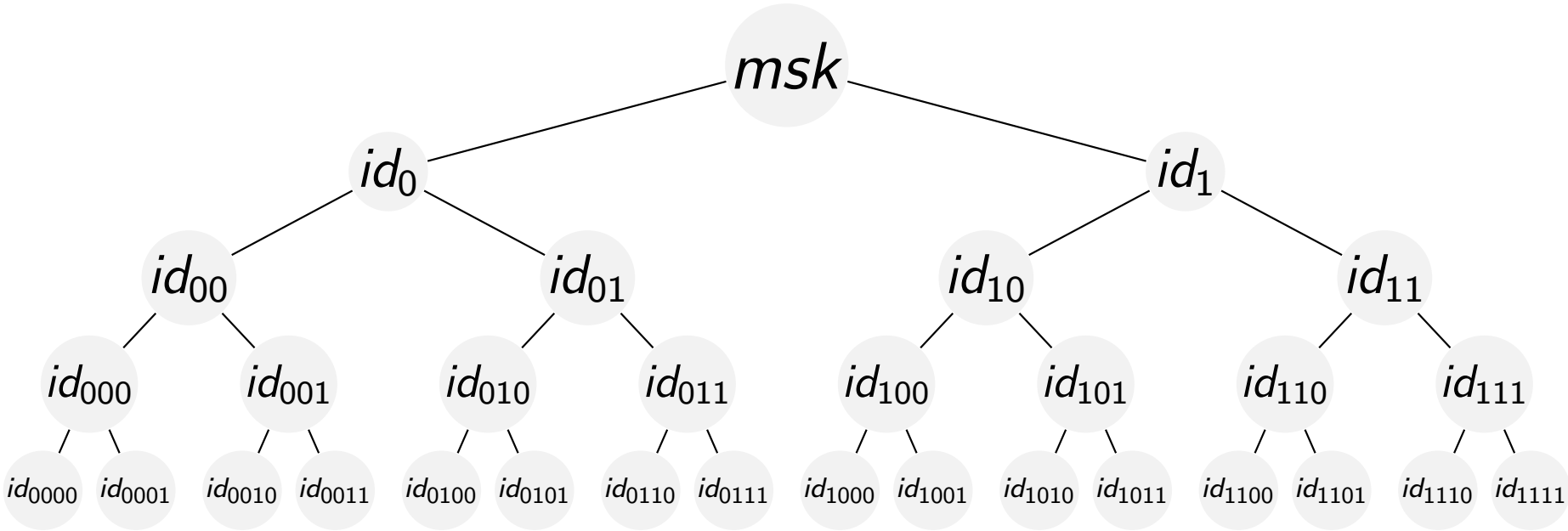
Lewko-Waters: Why proving HIBE is difficult

Hierarchical IBE:



Lewko-Waters: Why proving HIBE is difficult

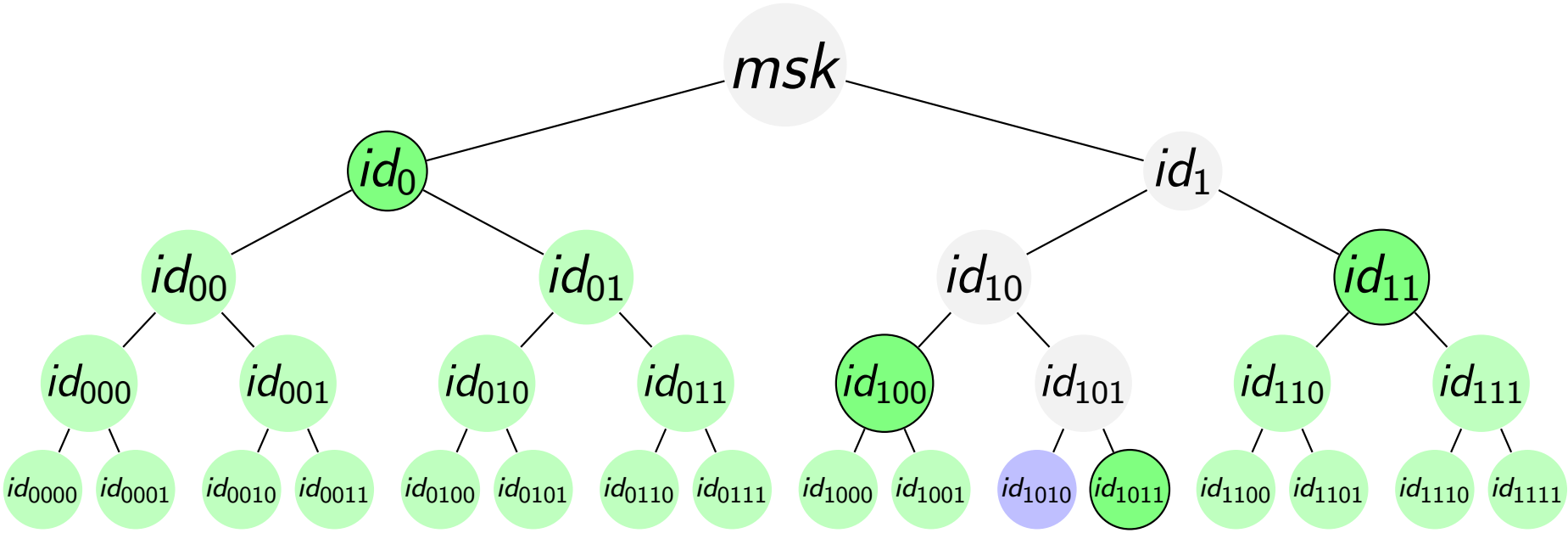
Hierarchical IBE:



- [LW14] show that partitioning proofs must lose exponential factor in depth of HIBE:

Lewko-Waters: Why proving HIBE is difficult

Hierarchical IBE:



- [LW14] show that partitioning proofs must lose exponential factor in depth of HIBE:
- A could query keys to decrypt for any but one *id*.
- $\Rightarrow$ reduction must guess *id*

Lewko-Waters: Why proving HIBE is difficult

Generalization: proof by meta-reduction:

- Assume reduction R:
  - get challenge
  - run A once
  - break challenge
- Construct (inefficient) A breaking HIBE

Lewko-Waters: Why proving HIBE is difficult

Generalization: proof by meta-reduction:

- Assume reduction R:
  - get challenge
  - run A once
  - break challenge
- Construct (inefficient) A breaking HIBE
- Simulate R and A efficiently
  1. run R to get pk
  2. A picks random id, queries all other keys, challenge on id
  3. $\leftarrow$ rewind, do (2) again
  4. use key from previous round to decrypt

Lewko-Waters: Why proving HIBE is difficult

Generalization: proof by meta-reduction:

- Assume reduction R:
  - get challenge
  - run A once
  - break challenge
- Construct (inefficient) A breaking HIBE
- Simulate R and A efficiently
  1. run R to get pk
  2. A picks random id, queries all other keys, challenge on id
  3. $\leftarrow$ rewind, do (2) again
  4. use key from previous round to decrypt

Condition

Keys have to be checkable w.r.t. pk

Lewko-Waters: Why proving HIBE is difficult

Generalization: proof by meta-reduction:

- Assume reduction R:
  - get challenge
  - run A once
  - break challenge
- Construct (inefficient) A breaking HIBE
- Simulate R and A efficiently
  1. run R to get pk
  2. A picks random id, queries all other keys, challenge on id
  3. $\leftarrow$ rewind, do (2) again
  4. use key from previous round to decrypt

Result

Simple reductions for HIBE must lose factor exp. in depth

FKPV: Why proving Boneh-Waters is difficult

Applied to Boneh-Waters PRF

- There is no public key!

$\Rightarrow$ A makes 2 fingerprint queries for constrained keys

$\Rightarrow$ these fix the secret key

- Keys are checkable w.r.t. fingerprint keys using pairings

FKPV: Why proving Boneh-Waters is difficult

Applied to Boneh-Waters PRF

- There is no public key!

$\Rightarrow$ A makes 2 fingerprint queries for constrained keys

$\Rightarrow$ these fix the secret key

- Keys are checkable w.r.t. fingerprint keys using pairings

Theorem

Let $\Pi(\lambda)$ be a decisional problem such that no algorithm running in time $t = poly(\lambda)$ has an advantage non-negligible in λ. Let R be a simple $(t, \epsilon, q, \delta, t')$ reduction from Π to unpredictability of the Boneh-Waters prefix-constrained PRF with domain $\{0,1\}^n$, with both t, t' polynomial in λ, and $q \geq n - 1$. Then δ vanishes exponentially as a function of n (up to terms that are negligible in λ).

Thank you! ☺