Be Adaptive, Avoid Overcommitting

Zahra Jafargholi, Chethan Kamath, Karen Klein,
Ilan Komargodski, Krzysztof Pietrzak and Daniel Wichs

AARHUS UNIVERSITY

IST AUSTRIA Institute of Science and Technology

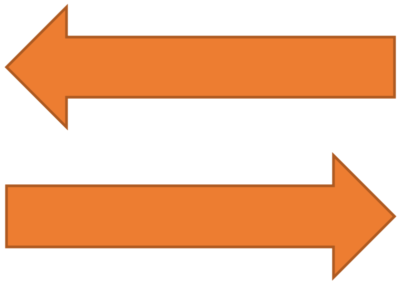
CORNELL TECH

Northeastern University

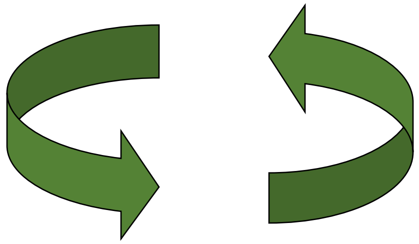
NY Crypto Day, September 2017

Selective vs. Adaptive Security

- **Selective security:**
  - Adversary has to commit to some or all of its choices ahead of time
  - Not very realistic
  - Easier to get



- **Adaptive security:**
  - Adversary can make various choices during the course of the attack
  - More realistic
  - Harder to get



Recent Work & Our Results

Modular reduction to pebbling & guessing arguments

Several recent works showing that schemes actually satisfy adaptive security:

- Generalized selective decryption (GSD) [Panjwani07,FJP15]
- Constrained PRFs [FKPR14]
- Garbled circuits [JW16]

Very long and technical

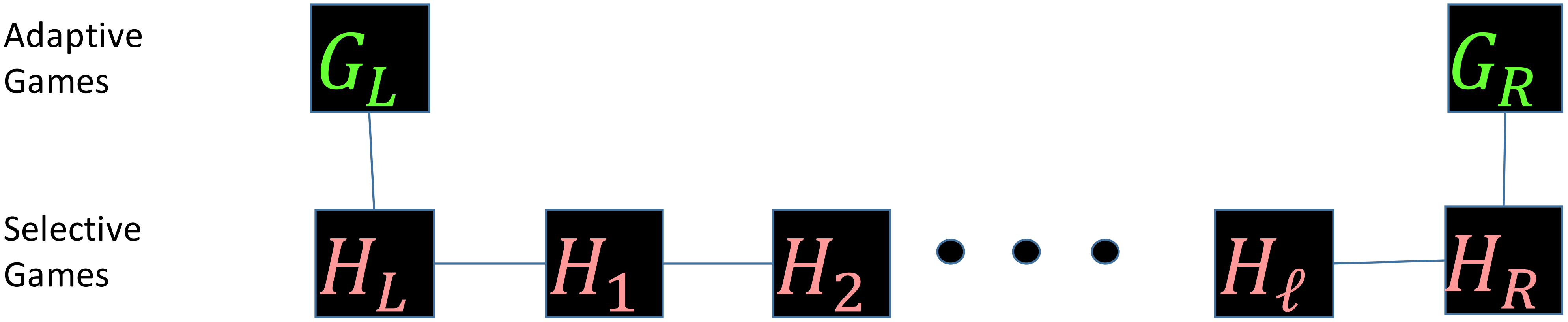
Vague consensus that proof techniques are related but no clear understanding

Similar framework by Ananth et al [TCC 2016]

- A framework that connects these works and allows us to present them in a unified and simplified fashion
- New result for adaptive security of Yao's secret sharing scheme

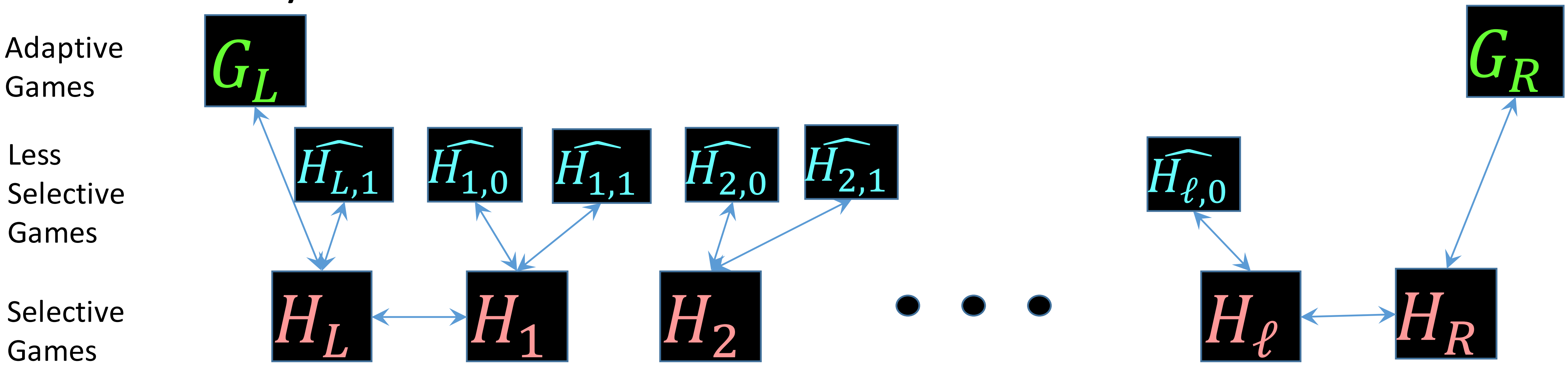
The Hybrid Argument & Random Guessing

- Let G_L and and G_R be two adaptive game
- Let H_L and H_R be their *selectivized* versions where the adversary commits to $w \in \{0,1\}^n$
- Assume that there is some sequence $H_L = H_0, H_1, \dots, H_\ell = H_R$ such that we can show that H_i and H_{i+1} are indistinguishable.
- Then, G_L and G_R are indistinguishable with security loss $2^n \ell$.



The Main Idea Underlying Our Framework

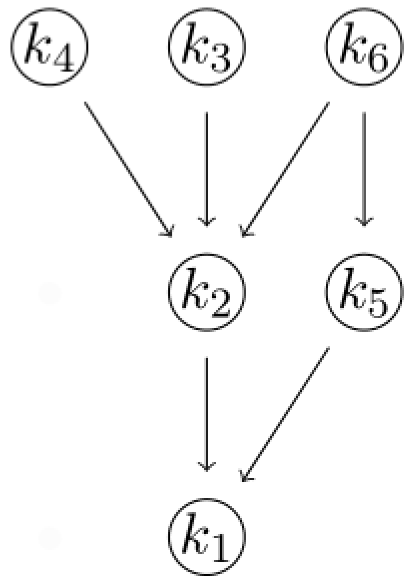
- Devise a sequence of hybrids such that to prove their indis. it is enough for the adversary to commit to $h(w) \in \{0,1\}^m$, $m \ll n$
  - May be a different h for every pair of hybrids
  - Across all hybrids we may need to know all of w
- Security loss is $2^m \ell \ll 2^n \ell$



The GSD Problem [Panjwani07]

- Have many secret keys $k_1, \dots, k_n$ and adversary can:
  - Ask for $\mathrm{Enc}(k_i, k_j)$ --- Encryption query
  - Ask to get k_i --- Corruption query
  - Make a challenge on key k_i --- Challenge

  - Decide whether it's real or random

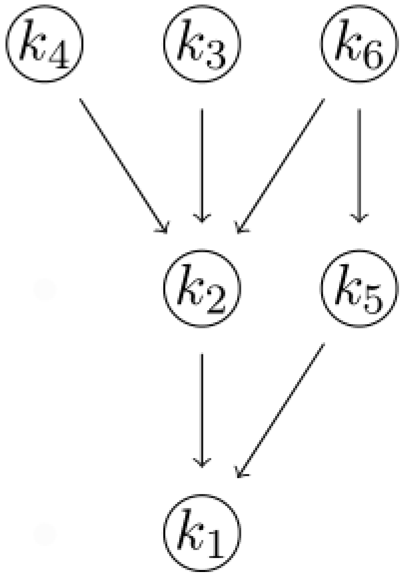


- Goal: distinguish between the two cases
  - No cycles
  - Key is not corrupted

Directly corrupted or is reachable from such

The GSD Problem – Selective Security

- Graph and all queries are known ahead of time
- Design n^2 hybrids where in each replace one honest encryption with a bogus one
  - Each pair is indistinguishable by IND-CPA
- Security loss is n^2



What about adaptive security?

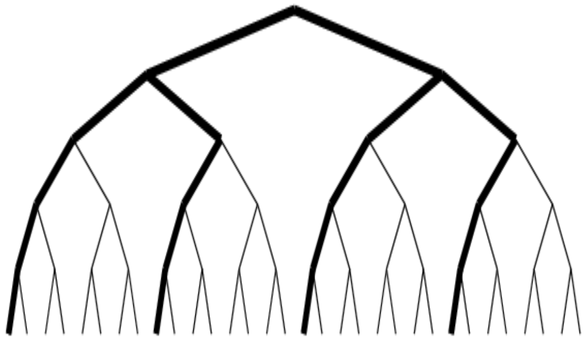
The GSD Problem – Adaptive Security

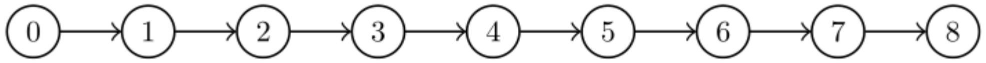
- Can reduce to the selective case by guessing the graph (n^2 bits)
- Security loss is $n^2 \cdot 2^{n^2}$

Can prove adaptive security without losing so much?

Known results:

- The graph is of depth d
  - Loss is $(2n)^d$ [Panjwani07]
- The graph is a tree
  - Loss is $n^{3\log n}$ [FJP15]
- The graph is a path
  - Loss is $n^{\log n}$ [FJP15]

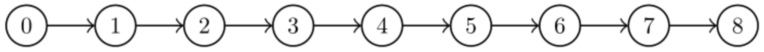




GSD on a PATH

- There is a path of length n & some permutation σ
- Adversaries queries are of the form $\mathrm{Enc}\big(\sigma(i-1), \sigma(i)\big)$
- The challenge is for $k_{\sigma(n)}$

- Know the permutation => know all queries.
- Know the order in which we replace ciphertext with bogus ones.



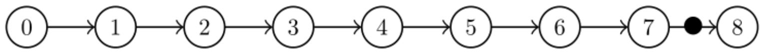
GSD on a PATH

- Any hybrid is defined by a path where some edges have black pebbles
- A pebble means that the corr. encryption query is replied with bogus

$Enc(k_{\sigma(i)}, k_{\sigma(i+1)}) \Rightarrow Enc(k_{\sigma(i)}, r)$

....

- Goal is to move from no pebble to the case that only the $(n-1, n)$ edge has a pebble
  - This is exactly the "random" game
- Pebbling rules:
  - Put/remove pebble on the source $(0,1)$ edge
  - Put/remove pebble on $(i, i+1)$ if $(i-1, i)$ has one.



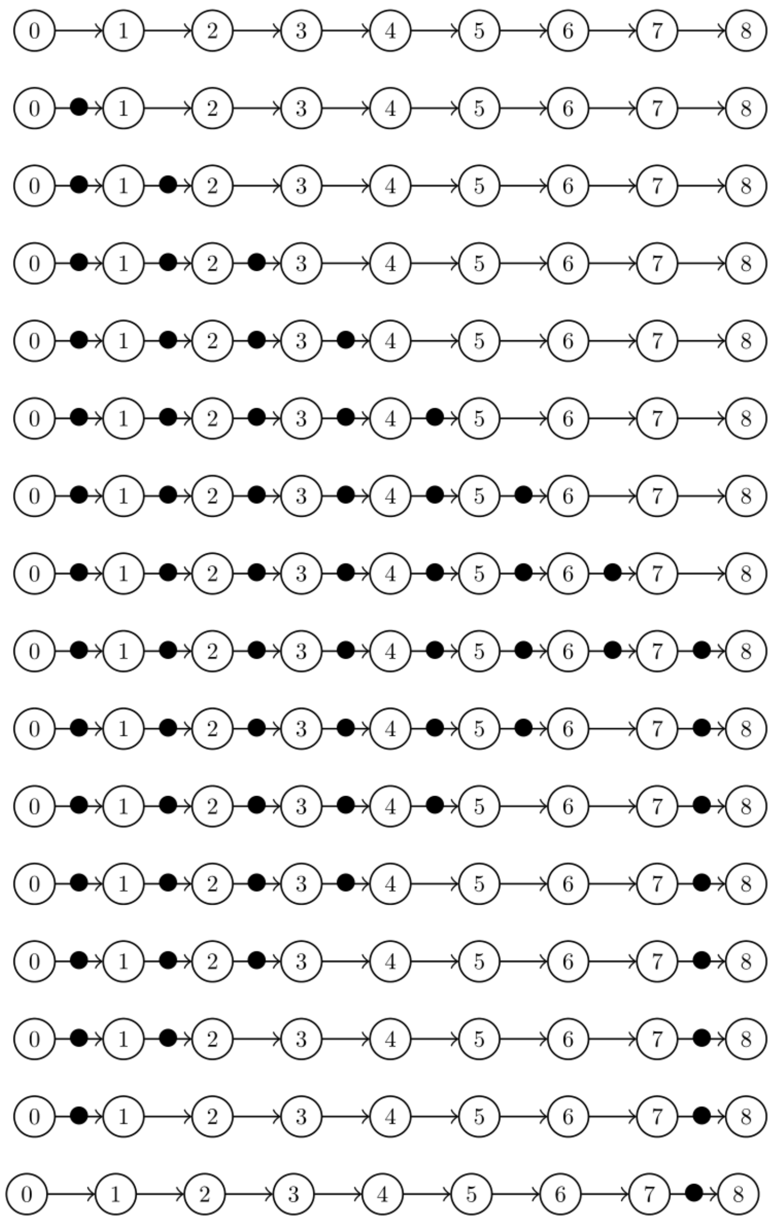
GSD on a PATH

- In the adaptive case we don't know the permutation.
  - Need to guess the edge where there's a pebble

- Unfortunately, # of pebbles is too large so guessing is too expensive

Goal: Find a pebbling strategy with not so many moves and as few as possible pebbles.

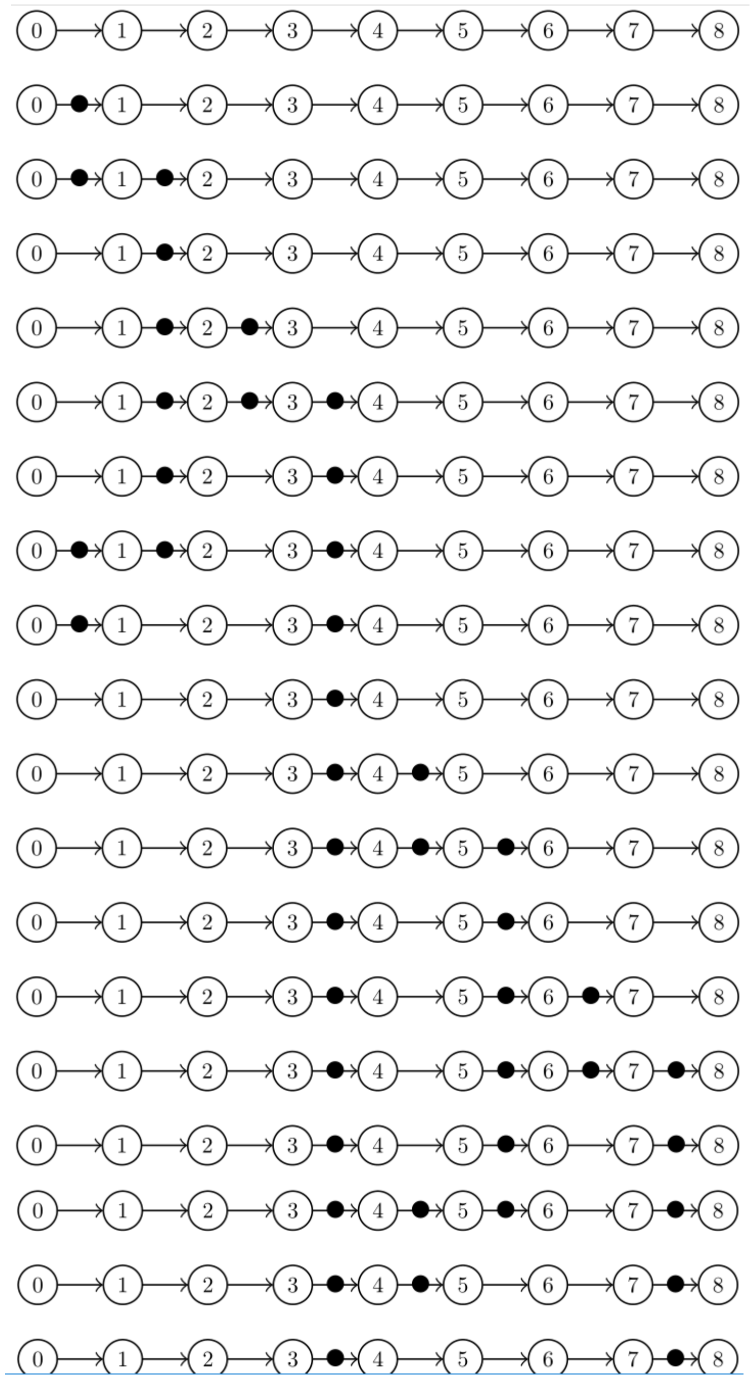
Loss will be ℓn^p
ℓ - # of hybrids
p – max # of pebbles

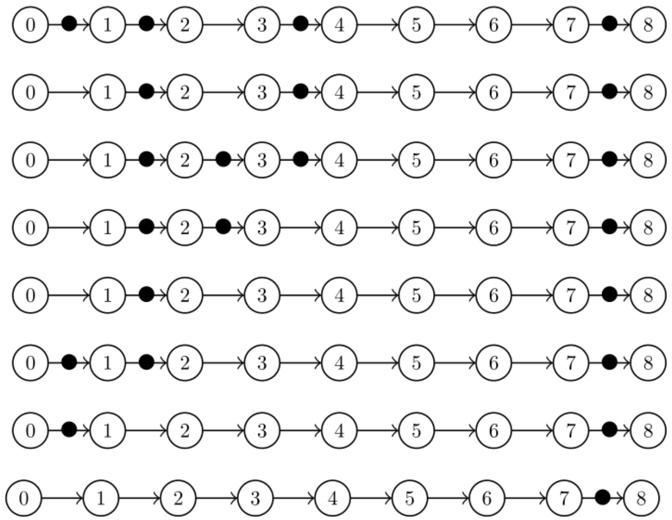


GSD on a PATH

- Recursive pebbling:
  - Pebble the middle
  - Pebble the right-most vertex
  - Remove the middle pebble
- $\log n + 1$ pebbles & $3^{\log n}$ moves

Loss is $\approx n^{\log n} \cdot 3^{\log n}$

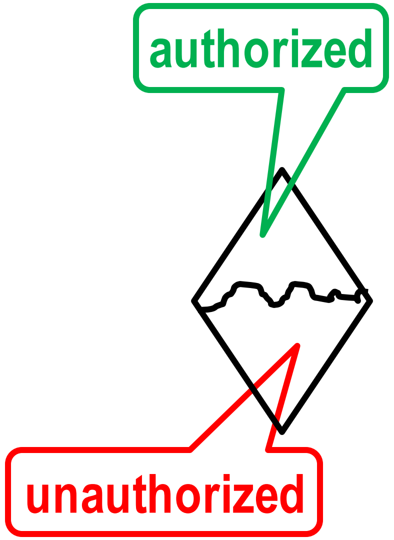




Secret Sharing

- **Dealer** has a secret S
- Gives to users shares $\Pi_1, \ldots, \Pi_n$
  - The shares are a probabilistic function of S

- A subset of users X is either **qualified** or **unqualified**
- Authorized sets form a *monotone* access structure



Goal:
- A qualified X can reconstruct S based on their shares.
- An unqualified X **cannot** gain *any* knowledge about S.

Perfect / Computational

Selective Secu ve Security

Objective

$\Pr[\text{Adv wins \& unqualified}] \leq \frac{1}{2} + \text{negl}(\lambda)$

Dealer		Adv
Sample $S \leftarrow \{0,1\}$ Generate shares $\Pi_1, \dots, \Pi_n$		Choose $\{i_1, \dots, i_k\}$
	← $i_1, \dots, i_k$	
	$\Pi_{i_1}, \dots, \Pi_{i_k}$ →	
Adv wins if $S' = S$	← S'	

Dealer		Adv
Sample $S \leftarrow \{0,1\}$ Generate shares $\Pi_1, \dots, \Pi_n$		
	← $i \in [n]$	
	Π_i →	
Adv wins if $S' = S$	← S'	

Our Result For Yao's Scheme

Theorem [Adaptive Security Loss in Yao's Scheme]:

Given an access structure described by a Boolean circuit with **fanin** k_{in} and **fanout** k_{out} with s **gates** and **depth** d,

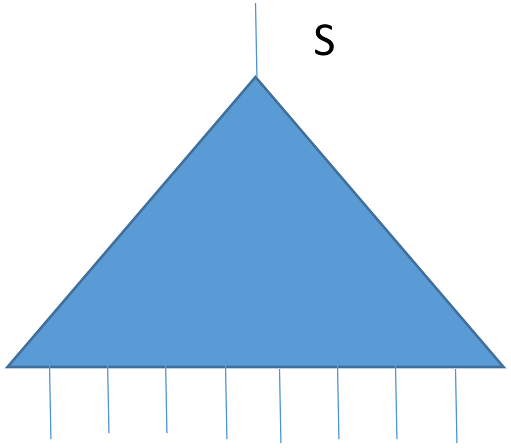
the loss in Yao's scheme is

$$2^{d(\log s+\log k_{\text{in}})} \cdot (2k_{\text{in}})^{2d} \cdot k_{\text{out}}$$

$$\approx$$

$$s^{O(d)}$$

Yao's Scheme

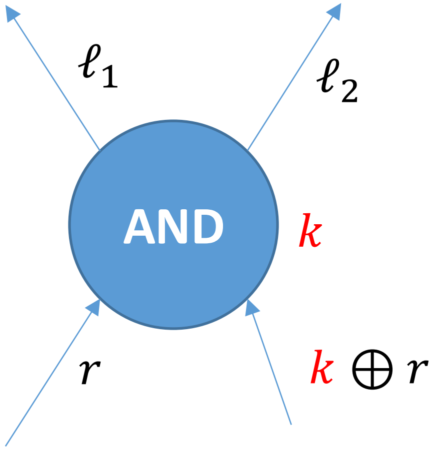


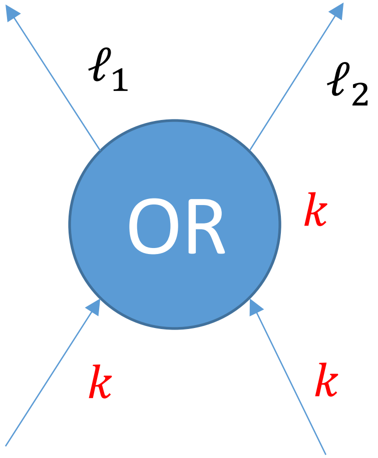
Assume fanin and fanout 2.

- Label the output wire with the secret
- Label all wires in the circuit from root to inputs
- The labels of the inputs are given to the corresponding parties

1. Sample SKE key k
2. If AND:
   One-time pad k
3. If OR:
   Duplicate k
4. Encrypt the out labels under k

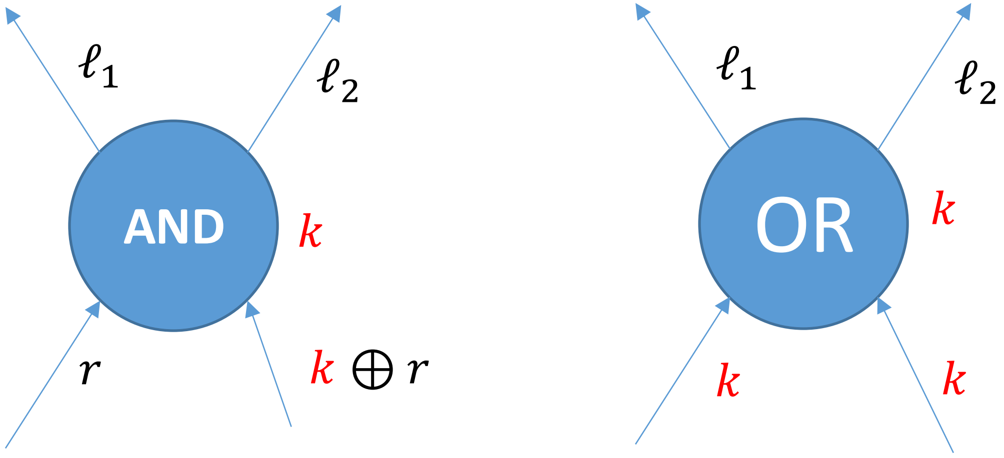
Give each party:
$\mathrm{Enc}_k(\ell_1)$
$\mathrm{Enc}_k(\ell_2)$





Proof of Selective Security

Give each party:
$\mathrm{Enc}_k(\ell_1)$
$\mathrm{Enc}_k(\ell_2)$



- Via a sequence of hybrids.
- Slowly replace ciphertexts with bogus ones
- We can do this for every gate for which the adv cannot compute the corresponding key
- When we do this to output gate $\Rightarrow$ shares are indep. of secret
- How do we know who are these gates?
- **Selective security:** adv commits to his set of parties ahead of time!

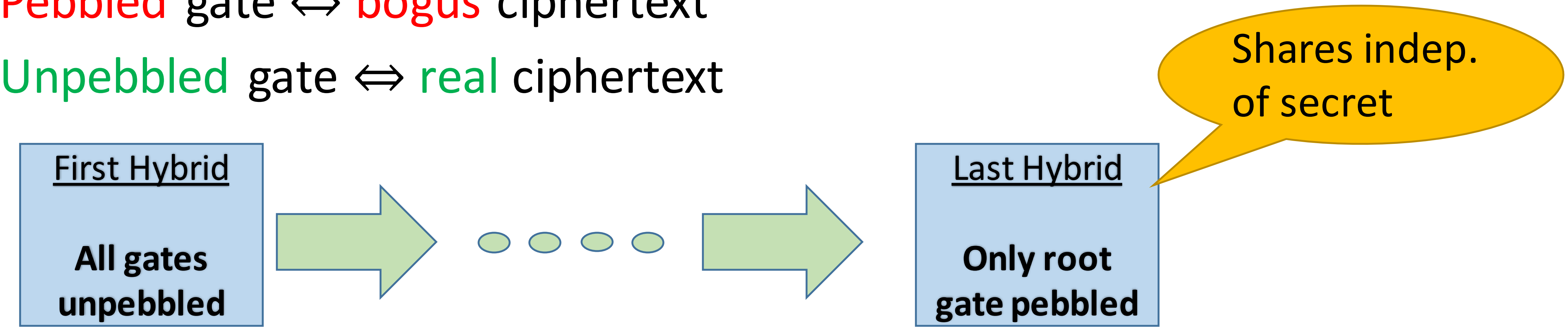
Seems inherent to know the set in order to devise such a sequence

- Such a sequence exists since chosen set is unqualified

Proof of Adaptive Security

- Devise a new sequence of hybrids.
- Hybrid H_i corresponds to a *pebbling configuration* in which every gate is either pebbled or not.

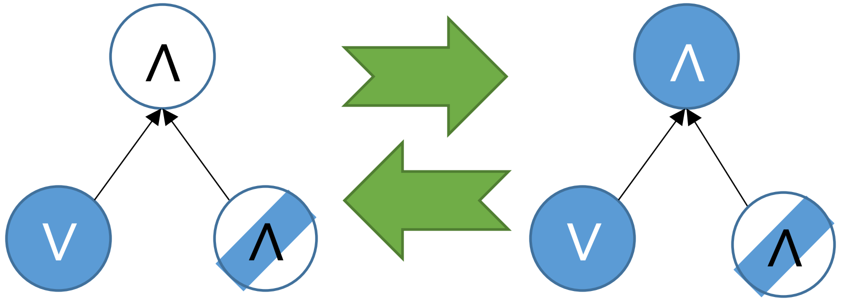
- Pebbled gate ⇔ bogus ciphertext
- Unpebbled gate ⇔ real ciphertext

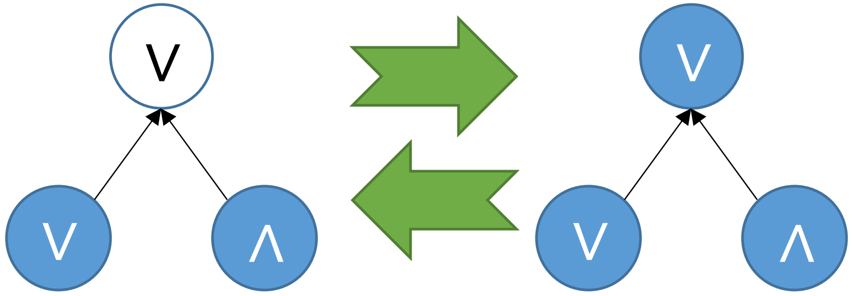


Proof of Adaptive Security

- Hybrid H_i corresponds to a *pebbling configuration* in which every gate is either pebbled or not.

- From hybrid H_i to hybrid H_{i+1} via pebbling rules:
  - Place/remove a pebble on AND gate for which **at least one** input is connected to a pebbled gate
  - Place/remove a pebble on OR gate for which **all** inputs are connected to pebbled gates.



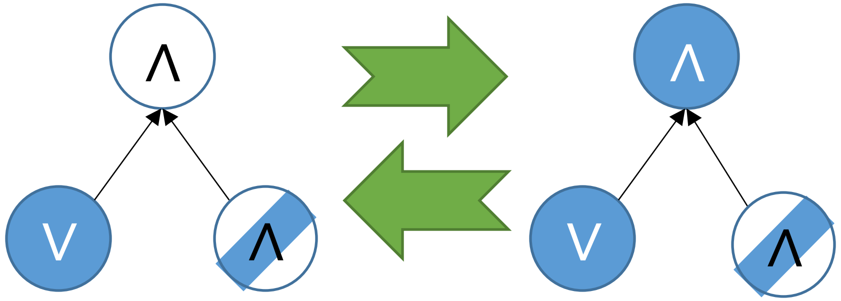


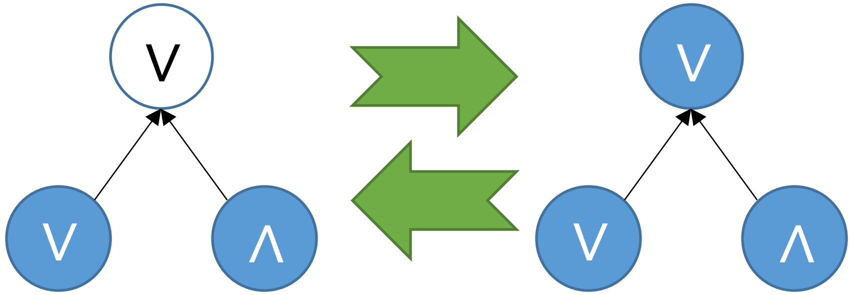
Proof of Adaptive Security

Main idea:

- In order to move from H_i to H_{i+1}, no need to know the corrupted set, but only the *pebble configurations* in these two hybrids
- If, in addition, each pebbling configuration requires *few* bits to describe, we can guess it.

Goal: Find a pebbling strategy with not so many moves that can be **described** with few bits.



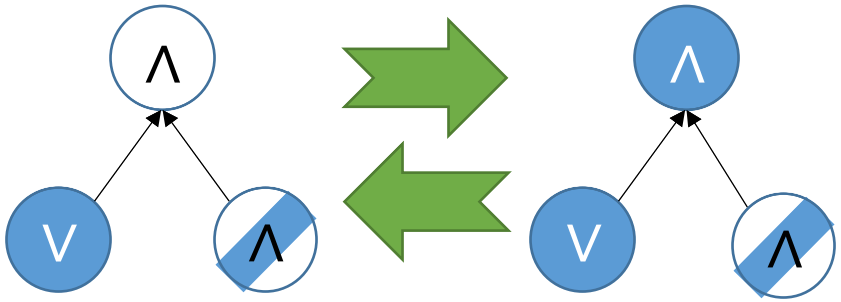


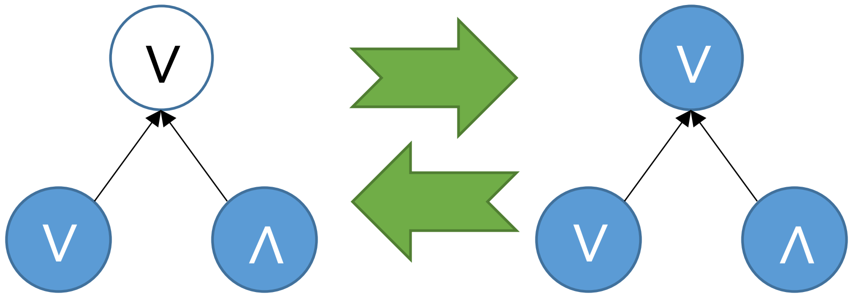
Proof of Adaptive Security

We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.

Pebbling Configuration:

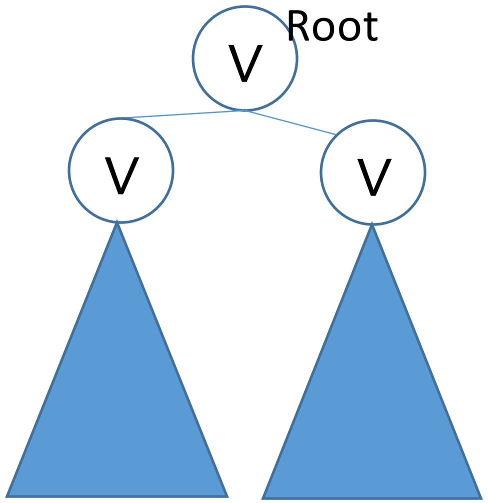
- Pairs of the form (GateName, Bit)
  - Bit will say if only left child is pebbled or both
- Another bit to specify whether root is pebbled





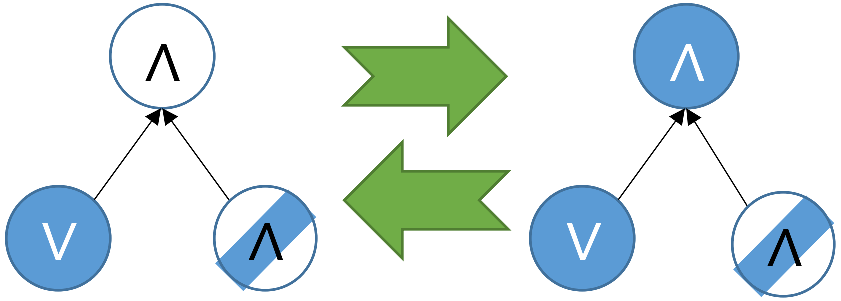
Proof of Adaptive Security

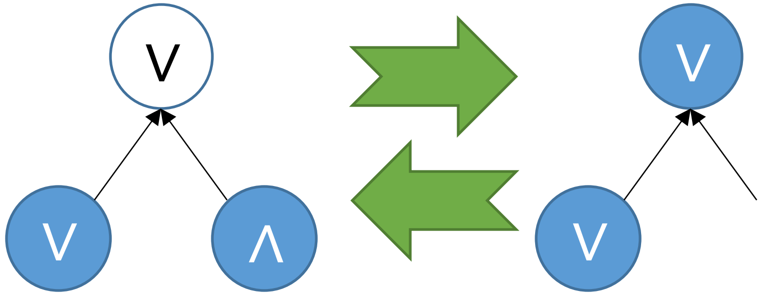
We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



Recursively pebble the left child of the root.

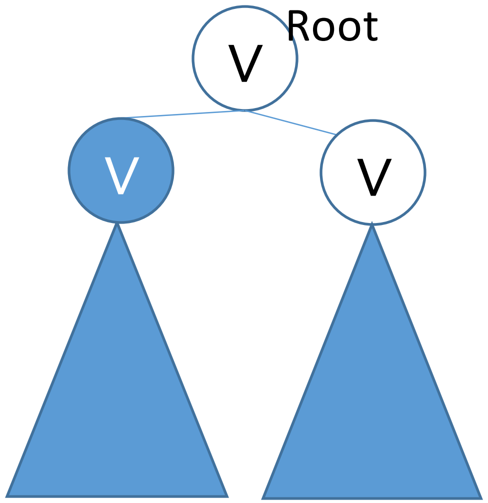
Add (RootGate,0) to configuration





Proof of Adaptive Security

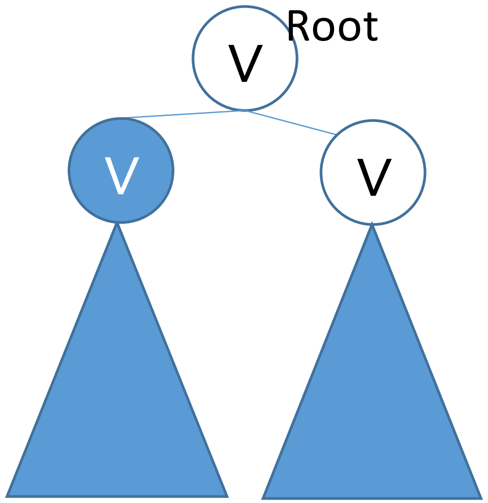
We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



Recursively pebble the left child of the root.
Add (RootGate,0) to configuration

Proof of Adaptive Security

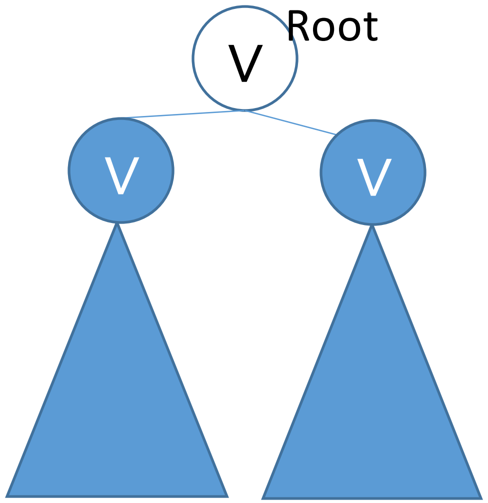
We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



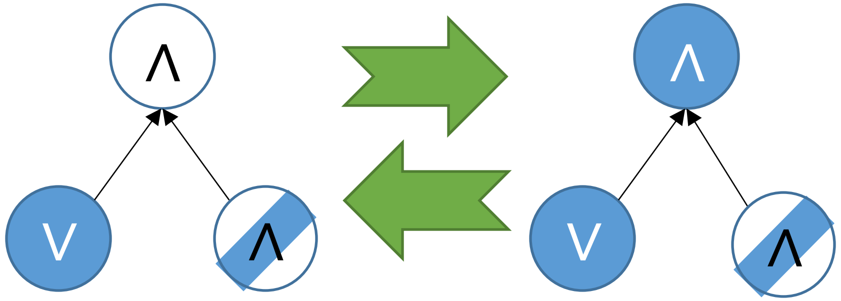
Recursively pebble the right child of the root.
Update (RootGate,1) in configuration

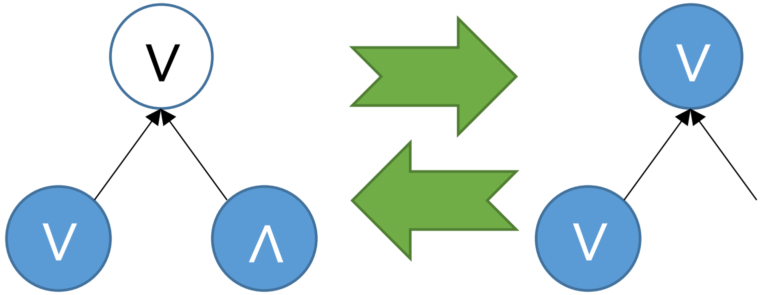
Proof of Adaptive Security

We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



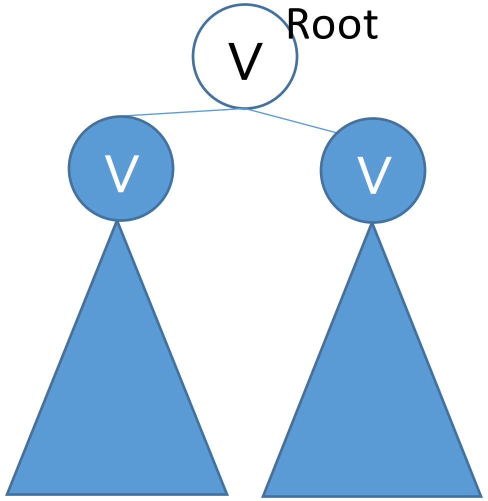
Recursively pebble the right child of the root.
Update (RootGate,1) in configuration





Proof of Adaptive Security

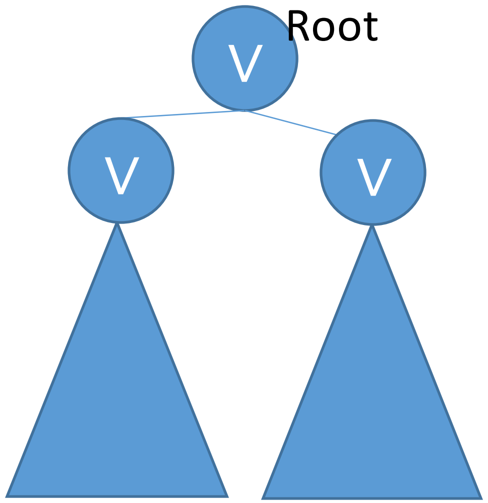
We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



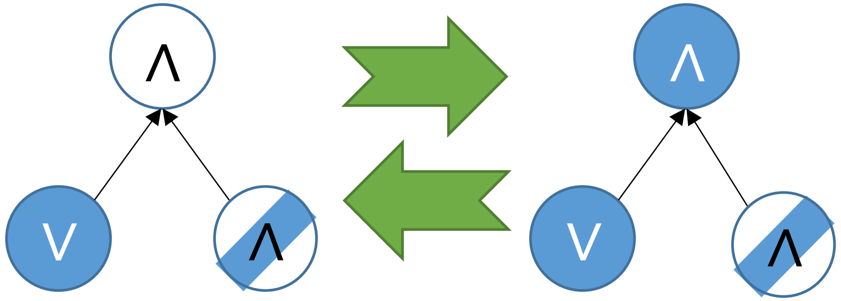
Put a pebble on the root
Update Bit in configuration

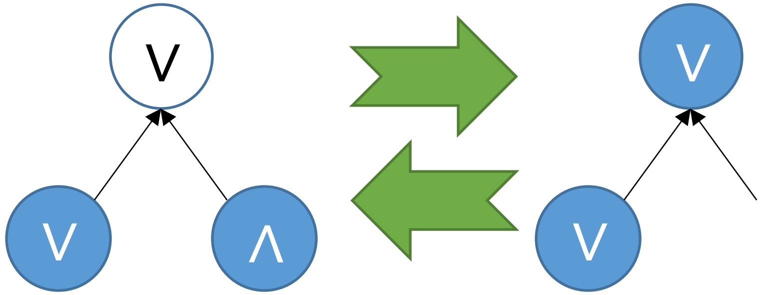
Proof of Adaptive Security

We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



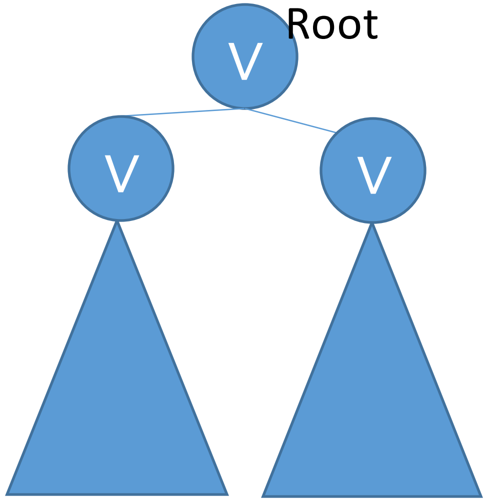
Put a pebble on the root
Update Bit in configuration



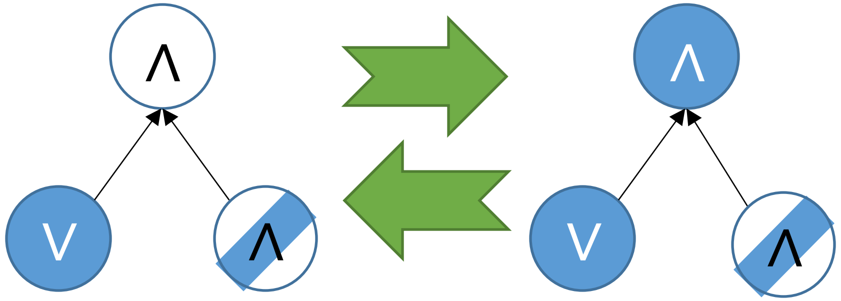


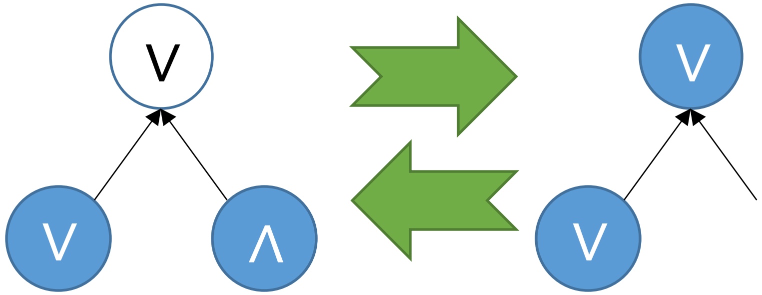
Proof of Adaptive Security

We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



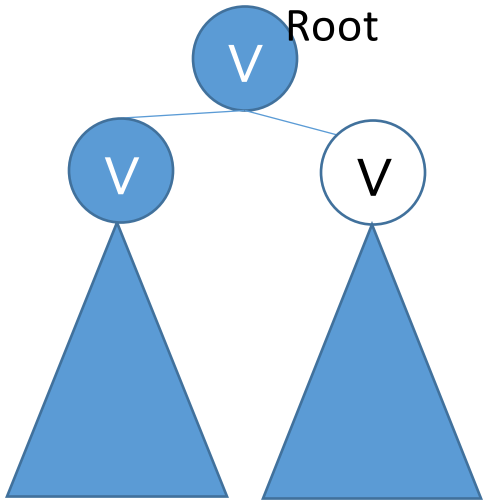
Unpabble right subtree of root
Update (RootGate,1) in configuration





Proof of Adaptive Security

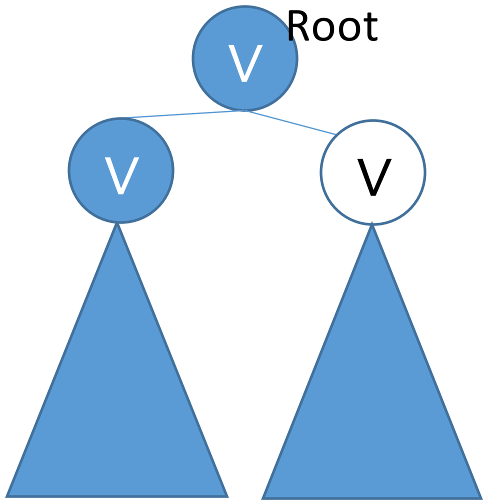
We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



Unpabble right subtree of root
Update (RootGate,1) in configuration

Proof of Adaptive Security

We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.

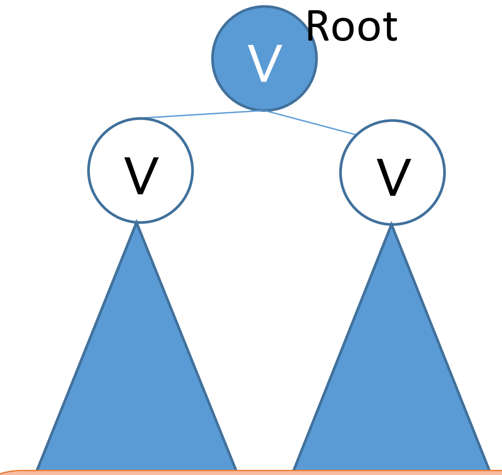


Unpabble left subtree of root

Remove (RootGate,1) from configuration

Proof of Adaptive Security

We give a pebbling strategy that requires $2^{O(d)}$ steps and each configuration can be described by $d \cdot \log s$ bits.



Unpabble left subtree of root
Remove (RootGate,1) from configuration

$T(d)$ = # of pebbling rules

$$T(d) = 2 \cdot 2 \cdot T(d-1)$$

$L(d) =$ size of conf.

$$L(d) = L(d-1) + \log s + 2$$

Conclusions

Thank You!

- A new framework for proving adaptive security
  - Simplified proof of adaptive security:
    GSD, Constrained PRFs, Yao's Garbled circuits
  - New result for Yao's secret sharing scheme.

- Find more applications where this framework applies
- Find better pebbling strategies
- Is there a connection in the other direction between pebbling strategies and the security loss?
  - Can we use lower bounds for pebbling strategies to devise attacks?