

Advanced Cryptography

CS 655

Week 1:

- Course Overview & Policy
- Review
- Concrete Security Analysis

Topic 0: Course Overview

Course Resources

Instructor: Jeremiah Blocki

Office Hours: Thursdays from 4:30-6:30PM

TA: Mohammad Hassan Ameri

Office Hours: TBD (Poll)

Course Web Page: Slides, homeworks and schedule

https://www.cs.purdue.edu/homes/jblocki/courses/655_Spring23/index.html

Technology

- **Brightspace**

- Syllabus (You are responsible for reading and understanding course policies)
- Grades

- **Gradescope**

- Submit homework assignments
- View Graded Assignments and Exams

- **Piazza**

- Course Discussion Board
- Announcements/Questions
- Preferred method of communication
- <https://piazza.com/purdue/spring2023/cs655/home>

Grades

- Course Project: 30%
- Homework: 20%
- Midterm Exam: 15%
- Final Exam: 25%
- Course Presentation: 10%

Collaboration is permitted on homework assignments, but you completely understand your solutions and you must write the solutions entirely in your own words.

No collaboration on exams

Expected Background

- **Prerequisites:** CS555 (or equivalent) and CS 526
- We will assume mathematical maturity
 - Ability to Write Proofs
 - Ability to Understand/Interpret Crypto Definitions
 - Security Reductions
 - Probability Theory
 - Combinatorics
 - Counting

Course Topics (Tentative)

- The course will review several cutting edge research topics in the field of cryptography.
- Cryptography is a big field → There will be many interesting topics that we cannot cover.
- **Tentative List of Topics:** Concrete Security Analysis, Idealized Models, Preprocessing Attacks and Lower Bounds, Proofs of Space, Proofs of Sequential Work, Verifiable Delay Functions, Memory Tight Reductions, Memory Hard Functions, Oblivious RAM, Obfuscation + Applications, Differential Privacy, Functional Secret Sharing, Quantum Random Oracle Model + Compressed Oracles, Differential Privacy, Fully Homomorphic Encryption

Homeworks (20%)

- There will be between 4 to 6 homework assignments. Homeworks must be submitted via Gradescope before the stated deadline.
- You should typeset your assignment (preferably using LaTeX). It is ok to include handdrawn figures in your pdf.
- You are encouraged to collaborate with others, but you must completely understand your solution and you should cite any resources that you use
- **Late Policy:**
 - On Time: No penalty
 - < 1 Day Late: 10-point penalty
 - < 2 Days Late: 25-point penalty
 - > 2 days late: no credit

Course Presentation (10%)

- Pick a recent paper (last 15 years) from a cryptography conference and give a 15 minute presentation in class
 - 12 minutes for presentation + 3 minutes for questions
- Suggested conferences include CRYPTO, EUROCRYPT, TCC, ASIACRYPT.
- If you would like to pick a crypto paper from another venue (e.g., S&P, CCS) you will need to verify that the focus of the paper is crypto and then request permission from the instructor.

Course Project (30%)

- Project Proposal: 5%
 - Progress Report: 5%
 - Final Project Report: 10%
 - Final Project Presentation: 10%
-
- We will provide some ideas for projects. You are also welcome to propose your own topics.
 - **Example:** New security analysis of ____ construction
 - **Example Twists:** Tighter Security Analysis in Ideal Model? Memory-Tight Reduction? Security proof in Quantum Random Oracle Model?
 - **Implementation:** pick a recent crypto construction and implement it

Topic 1: Review

Asymptotic Approach to Security

A scheme is secure if every *probabilistic polynomial time* (ppt) adversary “succeeds” with *negligible* probability.

- Two Key Concepts
 - Polynomial time algorithm
 - Negligible Function

Definition: A function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is negligible if for every positive polynomial p there is an integer $N > 0$ such that for all $n > N$ we have

$$f(n) < \frac{1}{p(n)}$$

Asymptotic Approach to Security

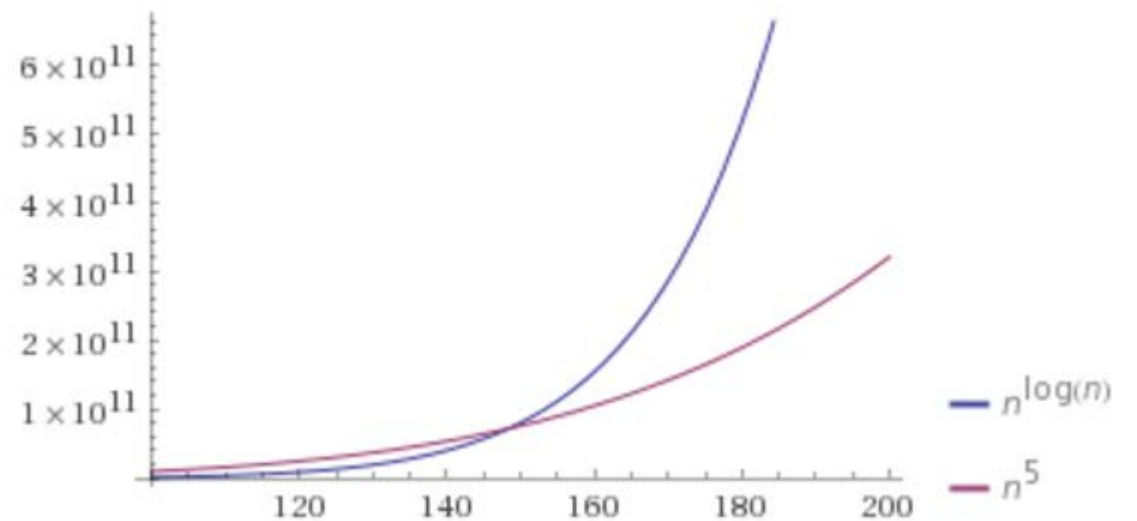
Definition: A function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is negligible if for every positive polynomial p there is an integer $N > 0$ such that for all $n > N$ we have

$$f(n) < \frac{1}{p(n)}$$

Which functions below are negligible?

- $f(n) = 2^{-n}$
- $f(n) = n^{-5}$
- $f(n) = 2^{-1000} 1000n^{1000}$
- $f(n) = 2^{100} 2^{-\sqrt{n}}$
- $f(n) = 2^{-\log n}$
- $f(n) = n^{-\log n}$

Plot:



Concrete Security

“A scheme is (t, ϵ) -secure if **every** adversary running for time at most t succeeds in breaking the scheme with probability at most ϵ ”

- Example: $t = 2^{60}$ CPU cycles
 - 9 years on a 4GHz processor
 - < 1 minute on fastest supercomputer (in parallel)
- Full formal definition needs to specify “break”
- Important Metric in Practice
 - **Caveat 1:** difficult to provide/prove such precise statements
 - **Caveat 2:** hardware improves over time

Asymptotic Approach to Security

Definition: A function $f: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is negligible if for every positive polynomial $p(\cdot) > 0$ there is an integer $N > 0$ such that for all $n > N$ we have

$$f(n) < \frac{1}{p(n)}$$

Intuition: If we choose the security parameter n to be sufficiently large then we can make the adversaries success probability very small (negligibly small).

Symmetric Key Building Blocks

- Pseudorandom Generator
- Pseudorandom Functions
- Pseudorandom Permutations + Block Ciphers
- Message Authentication Codes
- Hash Functions
- CPA-Secure Encryption
- Authenticated Encryption

Pseudorandom Generator (PRG) G

- **Input:** *Short* random seed $s \in \{0,1\}^n$
- **Output:** Longer “pseudorandom” string $G(s) \in \{0,1\}^{\ell(n)}$ with $\ell(n) > n$
 - $\ell(n)$ is called expansion factor
- **PRG Security:** For all PPT attacker A there is a negligible function $\text{negl}(\cdot)$ s.t

$$\left| \Pr_{s \in \{0,1\}^n} [A(G(s)) = 1] - \Pr_{R \in \{0,1\}^{\ell(n)}} [A(R) = 1] \right| \leq \text{negl}(n)$$

- **Concrete Security:** We say that $G(\cdot)$ is a $(t(n), \varepsilon(n))$ -secure PRG if for all attackers running in time at most $t(n)$ we have

$$\left| \Pr_{s \in \{0,1\}^n} [A(G(s)) = 1] - \Pr_{R \in \{0,1\}^{\ell(n)}} [A(R) = 1] \right| \leq \varepsilon(n)$$

Pseudorandom Function (PRF)

A keyed function $F: \{0,1\}^{\ell_{key}(n)} \times \{0,1\}^{\ell_{in}(n)} \rightarrow \{0,1\}^{\ell_{out}(n)}$, which “looks random” without the secret key k .

- $\ell_{key}(n)$ - length of secret key k
 - $\ell_{in}(n)$ - length of input
 - $\ell_{out}(n)$ - length of output
-
- Typically, $\ell_{key}(n)=\ell_{in}(n)=\ell_{out}(n)=n$ (unless otherwise specified)
 - Computing $F_k(x)$ is efficient (polynomial-time)

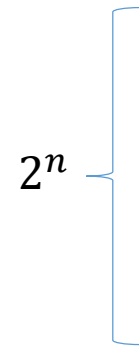
Truly Random Function

- Let **Func_n** denote the set of all functions $f: \{0,1\}^n \rightarrow \{0,1\}^n$.

- Question:** How big is the set **Func_n**?

- Hint:** Consider the lookup table.

- 2^n entries in lookup table
- n bits per entry ($f(x)$)
- $n2^n$ bits to encode $f \in \mathbf{Func}_n$



x	$f(x)$
0 ... 00	$f(0 \dots 00)$
0 ... 01	$f(0 \dots 01)$
0 ... 10	$f(0 \dots 10)$
...	...
1 ... 11	$f(1 \dots 11)$

- Answer:** $|\mathbf{Func}_n| = 2^{n2^n}$ (by comparison only $|\mathcal{K}| = 2^n$ n -bit keys)

PRF Security

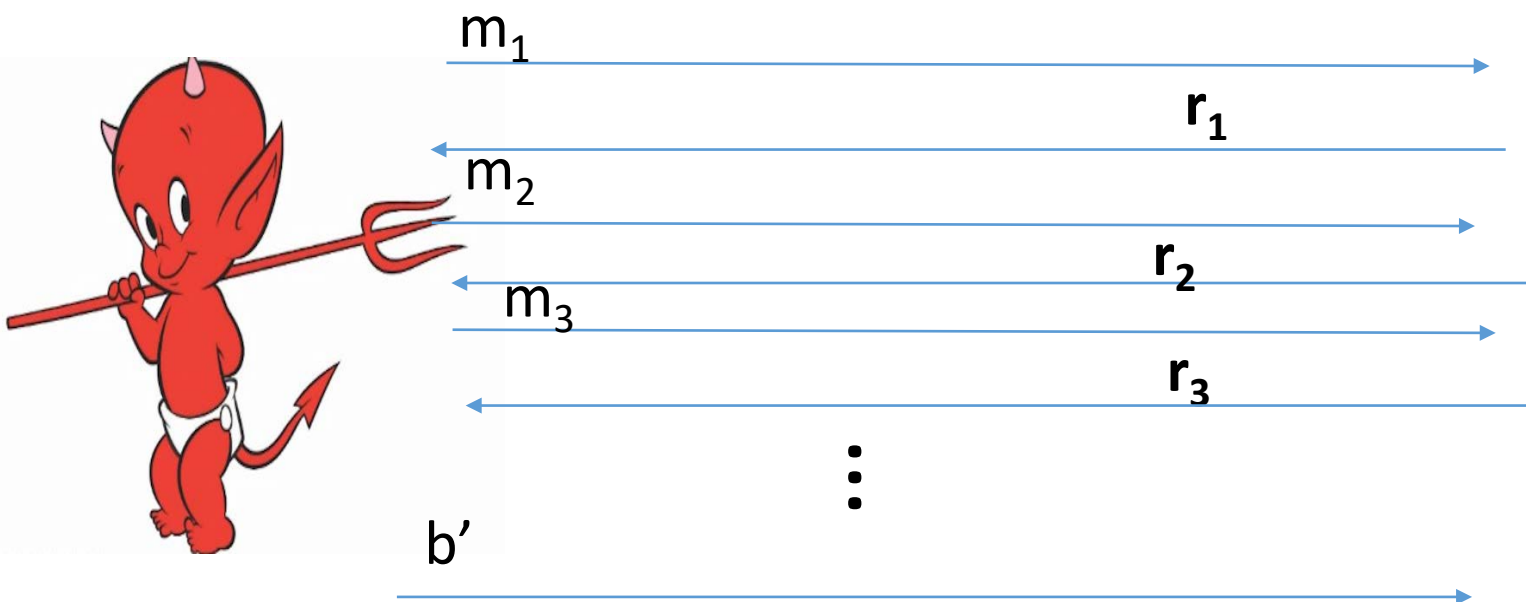
Definition 3.25: A keyed function $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a pseudorandom function if for all PPT distinguishers D there is a negligible function μ s.t.

$$|Pr[D^{F_k(\cdot)}(1^n)] - Pr[D^{f(\cdot)}(1^n)]| \leq \mu(n)$$

Notes:

- the first probability is taken over the uniform choice of $k \in \{0,1\}^n$ as well as the randomness of D .
- the second probability is taken over uniform choice of $f \in \mathbf{Func}_n$ as well as the randomness of D .
- D is *not* given the secret k in the first probability (otherwise easy to distinguish...how?)

PRF-Security as a Game



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$

$$\Pr[A \text{ Guesses } b' = b] \leq \frac{1}{2} + \mu(n)$$

Random bit b
 $K \leftarrow \text{Gen}(1^n)$
 Truly random func R
 $r_i = F_K(m_i) \quad \text{if } b=1$
 $R(m_i) \quad \text{o.w}$

PRF Security Concrete Version

Definition 3.25: A keyed function $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a $(t(n), q(n), \varepsilon(n))$ -secure pseudorandom if

$$ADV_{n,t,q} := \max_D ADV_{D,n,PRF} \leq \varepsilon(n)$$

Where

$$ADV_{D,n,PRF} := \left| Pr[D^{F_k(\cdot)}(1^n)] - Pr[D^{f(\cdot)}(1^n)] \right| \leq \varepsilon(n)$$

And the maximum is taken over all distinguishers D **running in time at most $t(n)$ and making at most $q(n)$ queries** we have

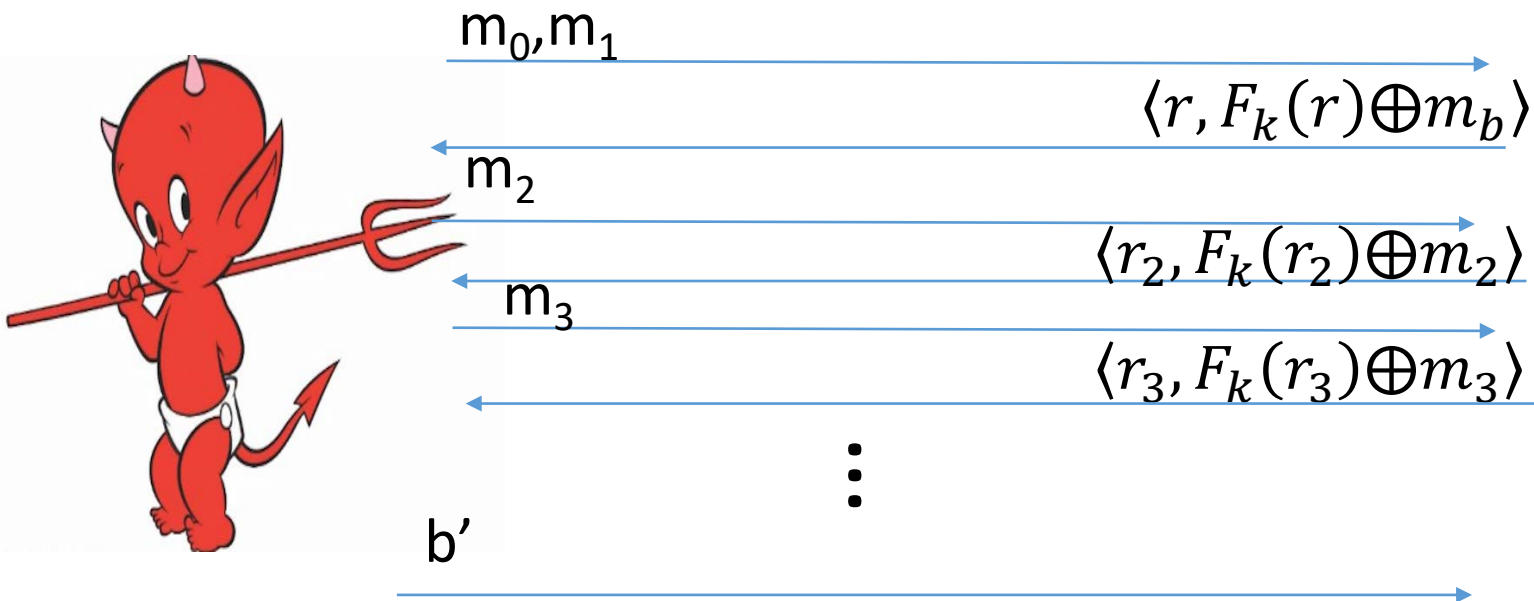
CPA-Secure Encryption

- **Gen:** on input 1^n pick uniform $k \in \{0,1\}^n$
- **Enc:** Input $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$
Output $c = \langle r, F_k(r) \oplus m \rangle$ for uniform $r \in \{0,1\}^n$
- **Dec:** Input $k \in \{0,1\}^n$ and $c = \langle r, s \rangle$
Output $m = F_k(r) \oplus s$

How to begin proof?

Theorem: If F is a pseudorandom function, then $(\text{Gen}, \text{Enc}, \text{Dec})$ is a CPA-secure encryption scheme for messages of length n .

CPA-Security Game (Single Message Version)



Random bit b
 $K \leftarrow \text{Gen}(1^n)$

$(t(n), q(n), \varepsilon(n))$ -secure if any attacker A running in time t and making at most q queries wins with probability at most $\frac{1}{2} + \varepsilon(n)$

Security Reduction

- **Step 1:** Assume for contraction that we have an attacker A that violates CPA-Security.
- **Step 2:** Construct a PPT distinguisher D which breaks PRF security.
- Distinguisher D^O (oracle O --- either f or F_k)
 - Simulate A
 - Whenever A queries its encryption oracle on a message m
 - Select random r and query O(r)
 - Return $c = \langle r, O(r) \oplus m \rangle$
 - Whenever A outputs messages m_0, m_1
 - Select random r and bit b
 - Return $c = \langle r, O(r) \oplus m_b \rangle$
 - Whenever A outputs b'
 - Output 1 if $b=b'$
 - Output 0 otherwise

Analysis: Suppose that $O = f$ then

$$\Pr[D^{F_k} = 1] = \Pr[\text{PrivK}_{A,\Pi}^{cpa} = 1]$$

Suppose that $O = f$ then

$$\Pr[D^f = 1] = \Pr[\text{PrivK}_{A,\tilde{\Pi}}^{cpa} = 1]$$

where $\tilde{\Pi}$ denotes the encryption scheme in which F_k is replaced by truly random f.

Security Reduction

- **Step 1:** Assume for contraction that we have a PPT attacker A that breaks CPA-Security.
- **Step 2:** Construct a PPT distinguisher D which breaks PRF security.
- Distinguisher D^O (oracle O --- either f or F_k)
 - Simulate A
 - Whenever A queries its encryption oracle on a message m
 - Select random r and query $O(r)$
 - Return $c = \langle r, O(r) \oplus m \rangle$
 - Whenever A outputs messages m_0, m_1
 - Select random r and bit b
 - Return $c = \langle r, O(r) \oplus m_b \rangle$
 - Whenever A outputs b'
 - Output 1 if $b=b'$
 - Output 0 otherwise

Analysis: By PRF security, we have

$$\begin{aligned} & \left| \Pr[PrivK_{A,\Pi}^{cpa} = 1] - \Pr[PrivK_{A,\tilde{\Pi}}^{cpa} = 1] \right| \\ &= \left| \Pr[D^{F_k} = 1] - \Pr[D^f = 1] \right| \leq ADV_{D,n,PRF} \end{aligned}$$

Implies: $\Pr[PrivK_{A,\tilde{\Pi}}^{cpa} = 1] \geq \Pr[PrivK_{A,\Pi}^{cpa} = 1] - ADV_{D,n,PRF}$

Security Reduction

- **Fact:** $\Pr \left[\text{PrivK}_{A, \tilde{\Pi}}^{cpa} = 1 \right] \geq \Pr \left[\text{PrivK}_{A, \Pi}^{cpa} = 1 \right] - \text{ADV}_{D, n, PRF}$
- **Claim:** For any attacker A making at most $q(n)$ queries we have

$$\Pr \left[\text{PrivK}_{A, \tilde{\Pi}}^{cpa} = 1 \right] \leq \frac{1}{2} + \frac{q(n)}{2^n}$$

Conclusion: For any attacker A making at most $q(n)$ queries we have

$$\Pr \left[\text{PrivK}_{A, \Pi}^{cpa} = 1 \right] \leq \frac{1}{2} + \frac{q(n)}{2^n} + \text{ADV}_{D, n, PRF}$$

where $\frac{q(n)}{2^n} + \text{ADV}_{D, n, PRF}$ is negligible.

Finishing Up

Claim: For any attacker A making at most $q(n)$ queries we have

$$\Pr \left[\text{PrivK}_{A, \tilde{\Pi}}^{cpa} = 1 \right] \leq \frac{1}{2} + \frac{q(n)}{2^n}$$

Proof: Let m_0, m_1 denote the challenge messages and let r^* denote the random string used to produce the challenge ciphertext

$$c = \langle r^*, f(r^*) \oplus m_b \rangle$$

And let r_1, \dots, r_q denote the random strings used to produce the other ciphertexts $c_i = \langle r_i, f(r_i) \oplus m_i \rangle$.

If $r^* \neq r_1, \dots, r_q$ then c leaks no information about b (information theoretically).

Finishing Up

Claim: For any attacker A making at most $q(n)$ queries we have

$$\Pr \left[\text{PrivK}_{A, \tilde{\Pi}}^{cpa} = 1 \right] \leq \frac{1}{2} + \frac{q(n)}{2^n}$$

Proof: If $r^* \neq r_1, \dots, r_q$ then c leaks no information about b (information theoretically). We have

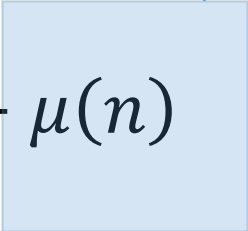
$$\begin{aligned} & \Pr \left[\text{PrivK}_{A, \tilde{\Pi}}^{cpa} = 1 \right] \\ & \leq \Pr \left[\text{PrivK}_{A, \tilde{\Pi}}^{cpa} = 1 \mid r^* \neq r_1, \dots, r_q \right] + \Pr \left[r^* \in \{r_1, \dots, r_q\} \right] \\ & \leq \frac{1}{2} + \frac{q(n)}{2^n} \end{aligned}$$

Conclusion

$$\text{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$$

$$\text{Dec}_k(\langle r, s \rangle) = F_k(r) \oplus s$$

For any attacker A making at most $q(n)$ queries we have

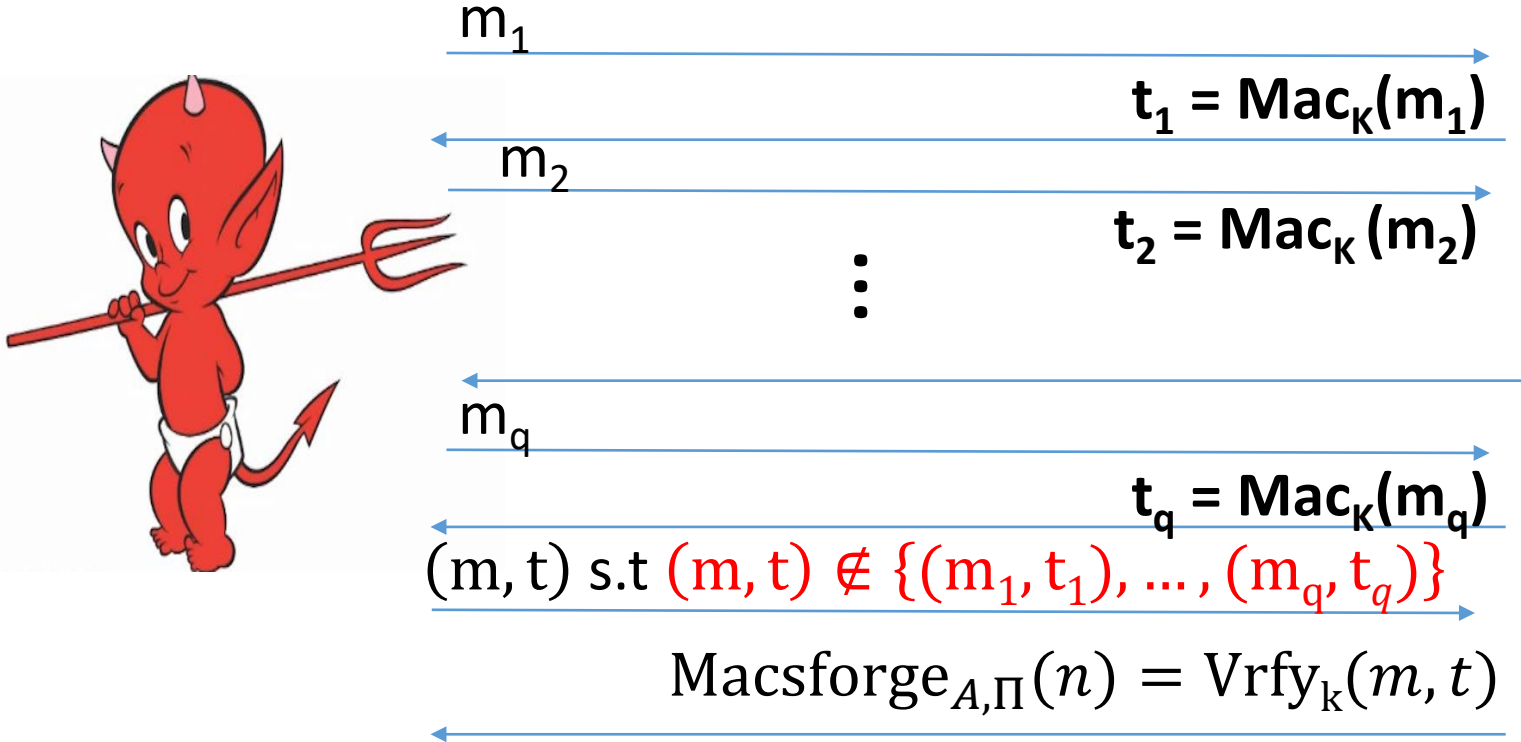
$$\Pr[\text{PrivK}_{A,\Pi}^{\text{cpa}} = 1] \leq \frac{1}{2} + \frac{q(n)}{2^n} + \mu(n)$$


PRF Security



Suggested Exercise: Work out concrete version of security proof

Concrete Version: $(t(n), q(n), \varepsilon(n))$ -secure MAC



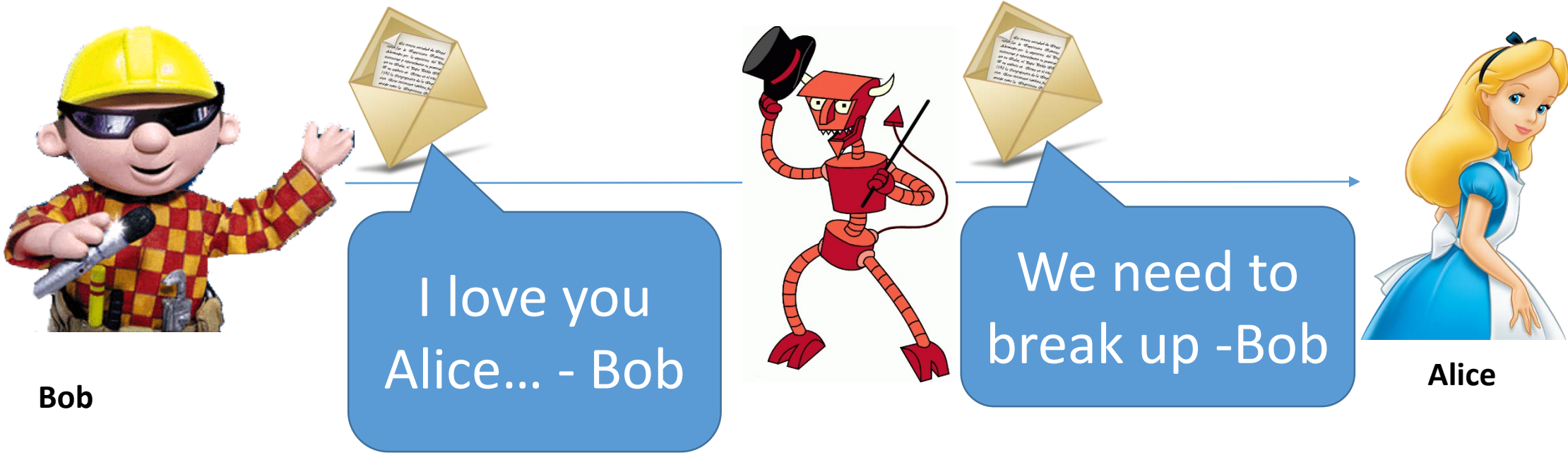
$K = \text{Gen}(\cdot)$



$\forall A$ with $(\text{time}(A) \leq t(n), \text{queries}(A) \leq q(n))$
 $\Pr[\text{Macsforge}_{A, \Pi}(n) = 1] \leq \varepsilon(n)$

What Does It Mean to “Secure Information”

- Confidentiality (Security/Privacy)
 - Only intended recipient can see the communication
- Integrity (Authenticity)
 - The message was actually sent by the alleged sender



Message Authentication Codes

- CPA-Secure Encryption: Focus on Secrecy
 - But does not promise integrity
 - Suppose Mallory intercepts $c = \langle r, F_k(r) \oplus m \rangle$
 - How can Mallory generate ciphertext for m' ?
- Message Authentication Codes: Focus on Integrity
 - But does not promise secrecy
- Authenticated Encryption: Requires Integrity and Secrecy

Message Authentication Code Syntax

Definition 4.1: A message authentication code (MAC) consists of three algorithms $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$

- $\text{Gen}(1^n; R)$ (Key-generation algorithm)
 - Input: security parameter 1^n (unary) and random bits R
 - Output: Secret key $k \in \mathcal{K}$
- $\text{Mac}_k(m; R)$ (Tag Generation algorithm)
 - Input: Secret key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$ and random bits R
 - Output: a tag t
- $\text{Vrfy}_k(m, t)$ (Verification algorithm)
 - Input: Secret key $k \in \mathcal{K}$, a message m and a tag t
 - Output: a bit b ($b=1$ means “valid” and $b=0$ means “invalid”)

$$\text{Vrfy}_k(m, \text{Mac}_k(m; R)) = 1$$

Security Goal (Informal): Attacker should not be able to forge a valid tag t' for new message m' that s/he wants to send.

Strong MAC Construction (Fixed Length)

$$\text{Mac}_k(m) = F_K(m)$$

$$\text{Vrfy}_k(m, t) = \begin{cases} 1 & \text{if } t = F_K(m) \\ 0 & \text{otherwise} \end{cases}$$

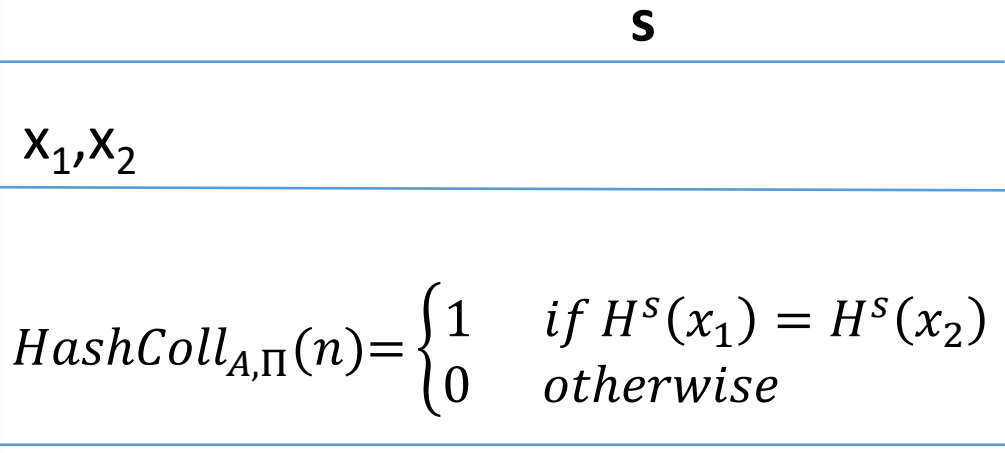
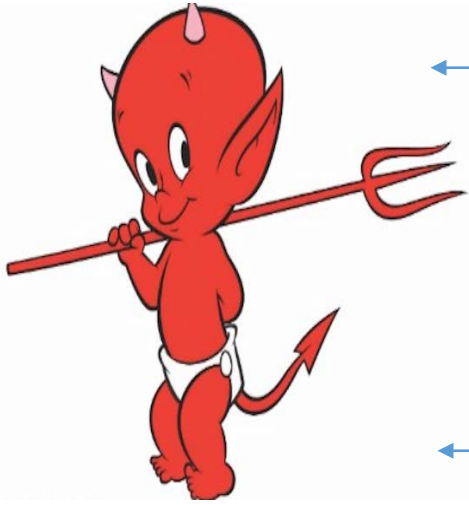
Theorem (Informal): If F is a PRF then this is a secure (fixed-length) MAC for messages of length n .

Proof: Start with attacker who breaks MAC security and build an attacker who breaks PRF security (contradiction!)

Sufficient to start with attacker who breaks regular MAC security (why?)

Concrete security?

Collision Experiment ($HashColl_{A,\Pi}(n)$)



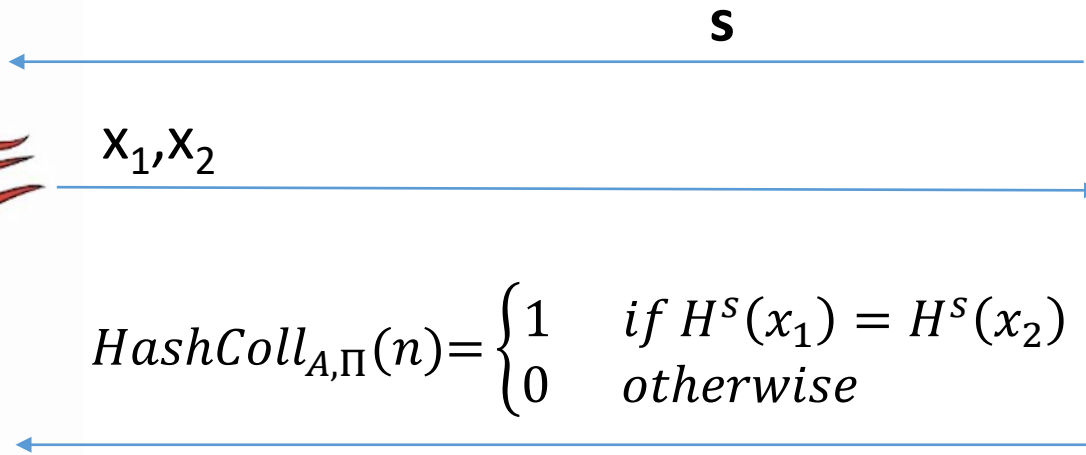
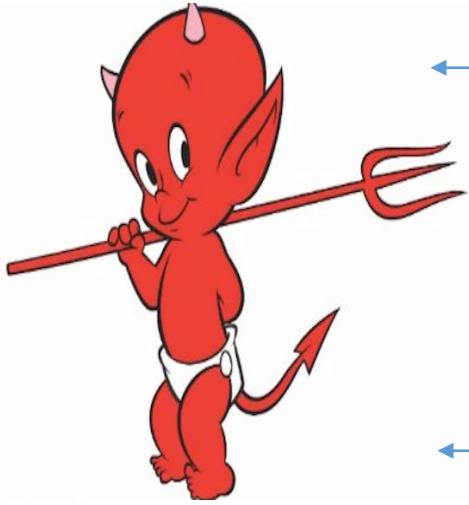
$$s = \text{Gen}(1^n; R)$$



Definition: (Gen, H) is a collision resistant hash function if

$$\forall PPT A \exists \mu \text{ (negligible) s.t.} \\ \Pr[HashColl_{A,\Pi}(n)=1] \leq \mu(n)$$

Concrete Security ($HashColl_{A,\Pi}(n)$)



$$s = \text{Gen}(1^n; R)$$



Definition: (Gen, H) is a (t, ε) –collision resistant hash function if \forall attackers A running in time at most $t(n)$

$$\Pr[HashColl_{A,\Pi}(n)=1] \leq \varepsilon(n)$$

Theory vs Practice

- Most cryptographic hash functions used in practice are un-keyed
 - Examples: MD5, SHA1, SHA2, SHA3, Blake2B
- Tricky to formally define collision resistance for keyless hash function
 - There is a PPT algorithm to find collisions
 - We just usually can't find this algorithm 😊
 - Guarantee for protocol using H

If we know an explicit efficient algorithm A breaking our protocol then there is an efficient blackbox reduction transforming A into an efficient collision finding algorithm.

Formalizing Human Ignorance:
Collision-Resistant Hashing without the Keys

Phillip Rogaway

Department of Computer Science, University of California,
Davis, California 95616, USA, and
Department of Computer Science, Faculty of Science,
Chiang Mai University, Chiang Mai 50200, Thailand
rogaway@cs.ucdavis.edu

31 January 2007

Abstract. There is a foundational problem involving collision-resistant hash-functions: common constructions are keyless, but formal definitions are keyed. The discrepancy stems from the fact that a function $H: \{0,1\}^* \rightarrow \{0,1\}^n$ *always* admits an efficient collision-finding algorithm, it's just that us human beings might be unable to write the program down. We explain a simple way to sidestep this difficulty that avoids having to key our hash functions. The idea is to state theorems in a way that prescribes an explicitly-given reduction, normally a black-box one. We illustrate this approach using well-known examples involving digital signatures, pseudorandom functions, and the Merkle-Damgård construction.

Hash and MAC Construction

Start with (Mac,Vrfy) a MAC for messages of fixed length and (Gen_H,H) a collision resistant hash function

$$Mac'_{\langle K_M, S \rangle}(m) = Mac_{K_M}(H^S(m))$$

Theorem 5.6: Above construction is a secure MAC.

Proof Intuition: If attacker successfully forges a valid MAC tag t' for unseen message m' then either

- **Case 1:** $H^S(m') = H^S(m_i)$ for some previously requested message m_i
- **Case 2:** $H^S(m') \neq H^S(m_i)$ for every previously requested message m_i

Hash and MAC Construction

Start with $(Mac, Vrfy)$ a MAC for messages of fixed length and (Gen_H, H) a collision resistant hash function

$$Mac'_{\langle K_M, S \rangle}(m) = Mac_{K_M}(H^S(m))$$

Theorem 5.6 (Concrete Version): If Mac is $(t, q_{MAC}, \epsilon_{MAC})$ – secure and (Gen_H, H) is (t, ϵ_{Hash}) – collision resistant then $Mac'_{\langle K_M, S \rangle}$ is $(O(t), q_{MAC}, \epsilon_{MAC} + \epsilon_{Hash})$ – secure

Proof Intuition: When A succeeds we either get a hash collision (case 1) or a Mac_{K_M} forgery (case 2)

if $\Pr[\text{case 2}] > \epsilon_{MAC}$ **we could violate** $(t, q_{MAC}, \epsilon_{MAC})$ – secure for Mac_{K_M}

Simulate $Mac'_{\langle K_M, S \rangle}$ attacker A

when attacker makes a query $Mac'_{\langle K_M, S \rangle}(m)$ we

1. compute $H^S(m)$ and
2. forward $H^S(m)$ to Mac_{K_M} oracle to get back $Mac_{K_M}(H^S(m))$

A's tag yields a Mac_{K_M} forgery for new message with probability at least $\Pr[\text{case 2}] > \epsilon_{MAC}$

Similar argument **if** $\Pr[\text{case 1}] > \epsilon_{HASH}$ **we could violate** (t, ϵ_{Hash}) – collision resistance for $H^S(.)$

Therefore, A succeeds with probability at most $\epsilon_{MAC} + \epsilon_{Hash}$

Merkle-Damgård Transform

Construction: (Gen,h) fixed length hash function from $2n$ bits to n bits

$$H^S(x) =$$

1. Break x into n bit segments x_1, \dots, x_d (pad last block by 0's)
2. $z_0 = 0^n$ (initialization)
3. For $i = 1$ to d
 1. $z_i = h^S(z_{i-1} \parallel x_i)$
4. Output $z_{d+1} = h^S(z_d \parallel L)$ where L encodes $|x|$ as an n -bit string

Random Oracle Model

- Model hash function H as a truly random function
- Algorithms can only interact with H as an oracle
 - **Query:** x
 - **Response:** $H(x)$
- If we submit the same query you see the same response
- If x has not been queried, then the value of $H(x)$ is uniform
- **Real World:** H instantiated as cryptographic hash function (e.g., SHA3) of fixed length (no Merkle-Damgård)

Random Oracle Model: Pros

- It is easier to prove security in Random Oracle Model
- Provably secure constructions in random oracle model are often much more efficient (compared to provably secure construction is “standard model”)
- Sometimes we only know how to design provably secure protocol in random oracle model

Random Oracle Model: Cons

- Lack of formal justification
- Why should security guarantees translate when we instantiate random oracle with a real cryptographic hash function?
- We can construct (contrived) examples of protocols which are
 - Secure in random oracle model...
 - But broken in the real world

Random Oracle Model: Justification

“A proof of security in the random-oracle model is significantly better than no proof at all.”

- **Evidence of sound design** (any weakness involves the hash function used to instantiate the random oracle)
- **Empirical Evidence for Security**
 - “there have been no successful real-world attacks on schemes proven secure in the random oracle model”

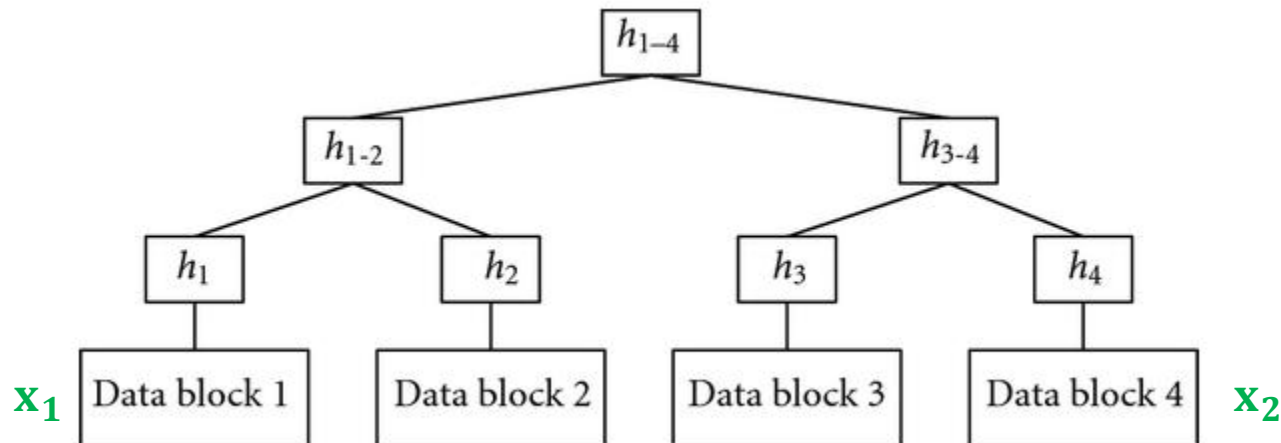
Merkle Trees

$$\mathbf{MT}^s(x) := h^s(x)$$

$$\mathbf{MT}^s(x_1, \dots, x_{2i}) :=$$

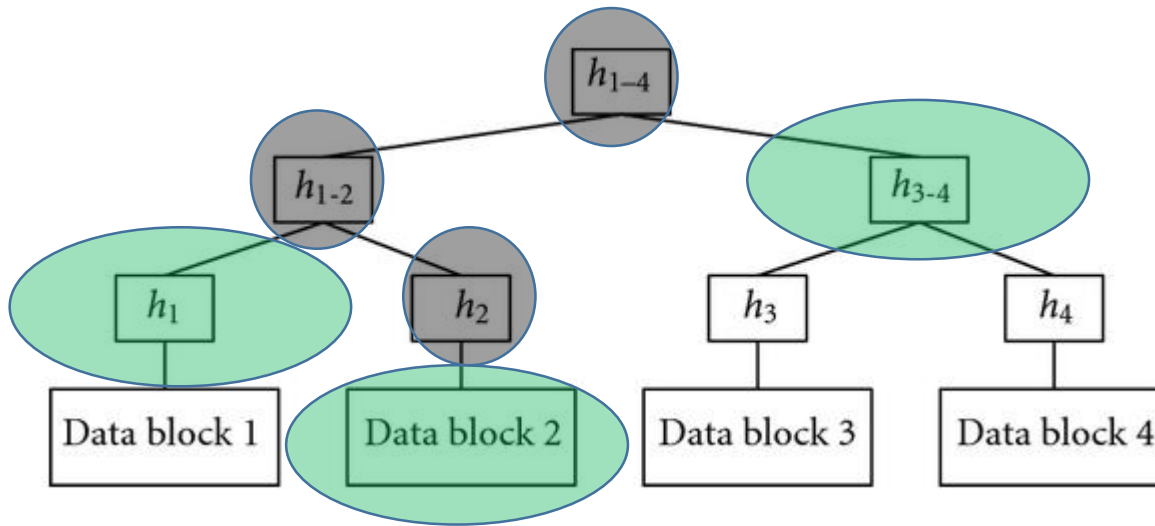
$$h^s\left(\mathbf{MT}^s(x_1, \dots, x_{2i-1}), \mathbf{MT}^s(x_{2i-1+1}, \dots, x_{2i})\right)$$

Theorem: Let (Gen, h^s) be a collision resistant hash function then \mathbf{MT}^s is collision resistant.



Merkle Trees

- **Proof of Correctness for data block 2**



- **Verify that root matches**
- **Proof consists of just $\log(n)$ hashes**
 - Verifier only needs to permanently store only one hash value



Ideal Cipher Model

- For each n -bit string K we pick a truly random permutation F_K
- Public Oracles
 - $O(K, x) = F_K(x)$
 - $O^{-1}(K, y) = F_K^{-1}(y)$
- Real World: Instantiate Ideal Cipher with a modern block cipher like AES
- Similar Pros/Cons to Random Oracle Model
 - Pro: Powerful evidence of sound design
 - Con: No blockcipher is an ideal cipher (even AES)

Hash Functions from Ideal Block Ciphers

- Davies-Meyer Construction from block cipher F_K

$$H(K, x) = F_K(x) \oplus x$$

Theorem: If $F: \{0,1\}^\lambda \times \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ is modeled as an ideal block cipher then Davies-Meyer construction is a collision-resistant hash function

(**Concrete:** Need roughly $q \approx 2^{\lambda/2}$ queries to find collision)

- **Ideal Cipher Model:** For each key K model F_K as a truly random permutation which may only be accessed in black box manner.
 - (Equivalent to Random Oracle Model)

Hash Functions from Block Ciphers

$$H(K, x) = F_K(x) \oplus x$$

Analysis: Suppose we have already made queries to the ideal cipher

- Submitted $(K_1, x_1), \dots, (K_q, x_q)$ to F_K to get $F_{K_1}(x_1), \dots, F_{K_q}(x_q)$
- Submitted $(K_{q+1}, y_1), \dots, (K_{2q}, y_q)$ to $F_K^{-1}(\cdot)$ to get $x_{q+1} := F_{K_{q+1}}^{-1}(y_1), \dots, x_{2q} := F_{K_{2q}}^{-1}(y_q)$.

$H(K_i, x_i)$ is known for all $i \leq 2q$ (but $H(K, x)$ is unknown at all other input points).

Now suppose we make a new query $(K, x) \notin \{(K_1, x_1), \dots, (K_{2q}, x_{2q})\}$: $F_K(x)$ sampled uniformly from $2^\lambda - 2q$ possible choices.

- ➔ Collides with $H(K_i, x_i)$ with probability at most $\frac{1}{2^\lambda - 2q}$
- ➔ Collides with $H(K_{q+i}, x_{q+i})$ with probability at most $\frac{1}{2^\lambda - 2q}$
- ➔ $H(K, x)$ Collides with prior query with probability at most $\frac{2q}{2^\lambda - 2q}$

Hash Functions from Block Ciphers

$$H(K, x) = F_K(x) \oplus x$$

Analysis:

Fact 1: Query $q+1$ to ideal cipher yields collision (with prior query) with probability at most $\frac{q}{2^\lambda - q}$

Fact 2: The probability of finding a collision within q queries is at most $\sum_{i \leq q} \frac{i}{2^\lambda - i} \leq \frac{q(q-1)/2}{2^\lambda - q}$

A Broken Attempt

$$H(K_1, K_2, x_1, x_2) = F_{K_1}(x_1) \oplus F_{K_2}(x_2) \oplus K_1 \oplus K_2$$

Collision Attack: Pick arbitrary keys $K_0 \neq K_1$

Step 1: Query $x_1 := F_{K_0}^{-1}(0^n)$ and $x_2 := F_{K_0}^{-1}(1^n)$

Step 2: Query $w_1 := F_{K_1}^{-1}(0^n)$ and $w_2 := F_{K_1}^{-1}(1^n)$

$$\begin{aligned} H(K_0, K_0, x_1, x_2) &= F_{K_0}(x_1) \oplus F_{K_0}(x_2) \oplus K_0 \oplus K_0 = 0^n \oplus 1^n \\ &= F_{K_1}(w_1) \oplus F_{K_1}(w_2) = H(K_1, K_1, w_1, w_2) \end{aligned}$$

Exploits the fact that we can query inverse oracle F_K^{-1}

Pseudorandom Permutation

Definition 3.28: A keyed function $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a **strong pseudorandom permutation** if for all PPT distinguishers D there is a negligible function μ s.t.

$$\left| \Pr \left[D^{F_k(\cdot), F_k^{-1}(\cdot)}(1^n) \right] - \Pr \left[D^{f(\cdot), f^{-1}(\cdot)}(1^n) \right] \right| \leq \mu(n)$$

Notes:

- the first probability is taken over the uniform choice of $k \in \{0,1\}^n$ as well as the randomness of D .
- the second probability is taken over uniform choice of $f \in \mathbf{Perm}_n$ as well as the randomness of D .
- D is *never* given the secret k
- However, D is given oracle access to keyed permutation and inverse

PRFs from PRGs

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

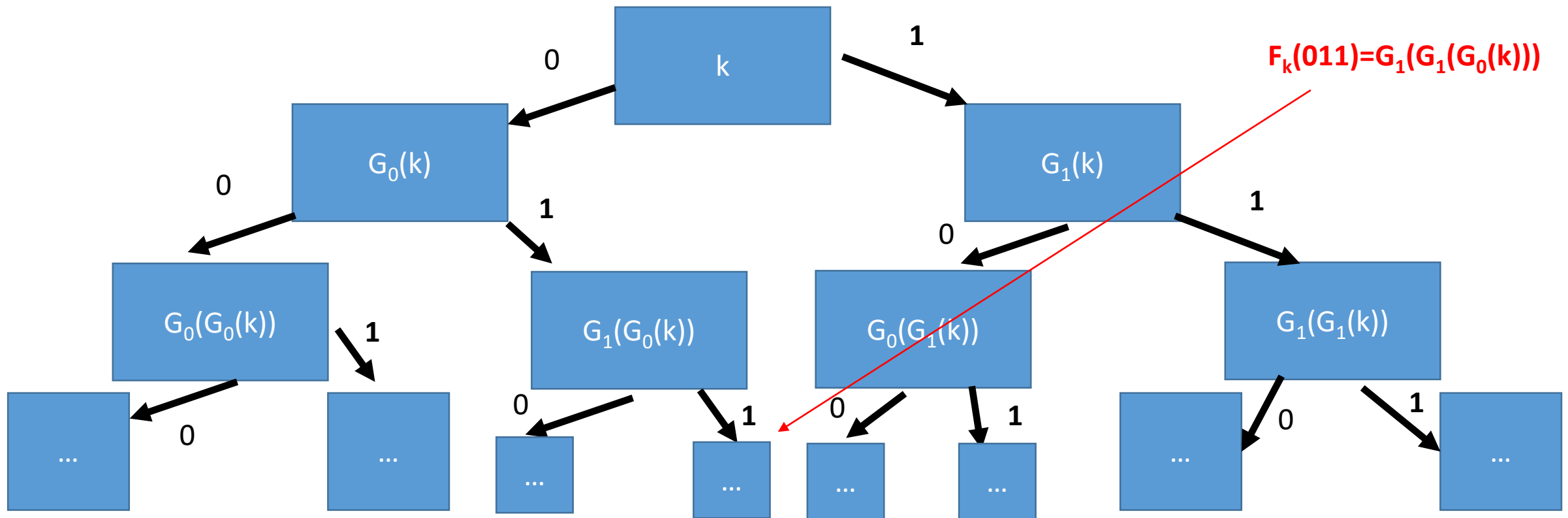
Let $G(x) = G_0(x) || G_1(x)$ (first/last n bits of output)

$$F_K(x_1, \dots, x_n) = G_{x_n} \left(\dots \left(G_{x_2} \left(G_{x_1}(K) \right) \right) \dots \right)$$

PRFs from PRGs

$$\mathbf{G}(\mathbf{x}) := \overbrace{\mathbf{G}_0(\mathbf{x})}^{n\text{-bits}} \parallel \overbrace{\mathbf{G}_1(\mathbf{x})}^{n\text{-bits}}$$

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.



PRFs from PRGs

Theorem: Suppose that there is a PRG G with expansion factor $\ell(n) = 2n$. Then there is a secure PRF.

Proof:

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \text{negl}(n)$$

PRFs from PRGs

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \text{negl}(n)$$

Proof Sketch (by Triangle Inequality): Fix j

$$\begin{aligned} & \text{Adv}_j \\ &= \left| \Pr[A(r_1 \parallel \cdots \parallel r_{j+1} \parallel G(s_{j+2}) \parallel \cdots \parallel G(s_{t(n)}))] \right| \end{aligned}$$

PRFs from PRGs

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \textit{negl}(n)$$

Proof Sketch

$$\begin{aligned} & \left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| \\ & \leq \sum_{j < t(n)} \textit{Adv}_j \\ & \leq t(n) \times \textit{negl}(n) = \textit{negl}(n) \quad (\textit{QED}) \end{aligned}$$

PRFs from PRGs

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \text{negl}(n)$$

Proof

$$\begin{aligned} & \left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| \\ & \leq \sum_{j < t(n)} \text{Adv}_j \\ & \leq t(n) \times \text{negl}(n) = \text{negl}(n) \end{aligned}$$

PRFs from PRGs

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \text{negl}(n)$$

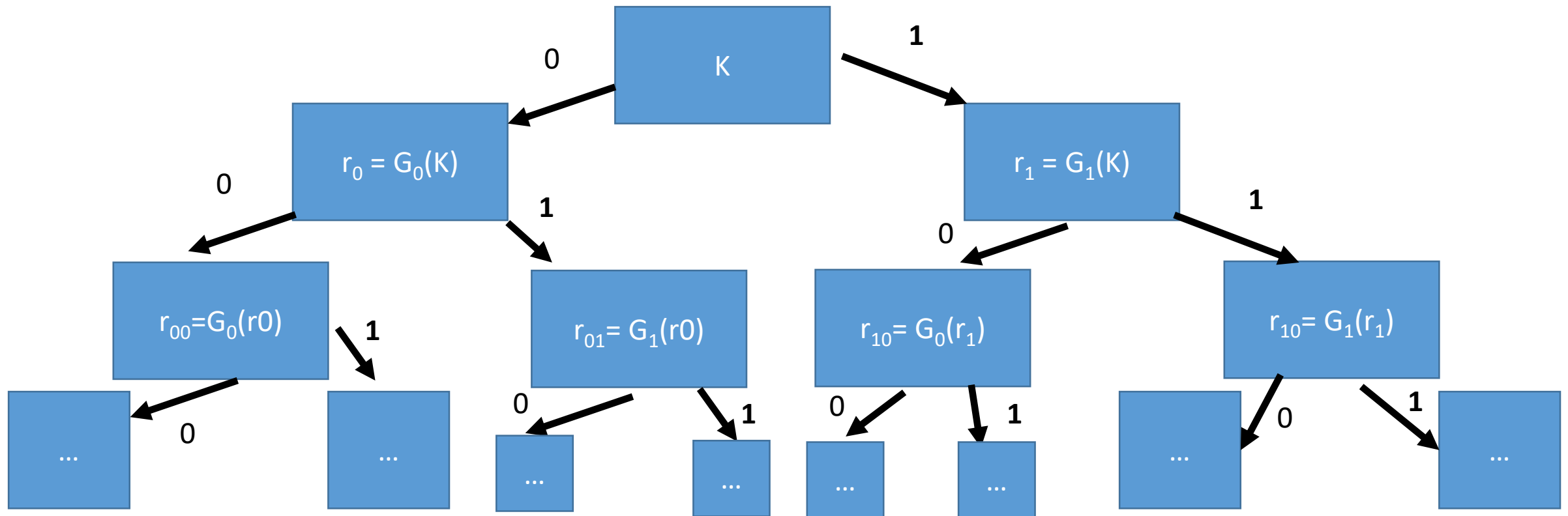
Proof

$$\begin{aligned} & \left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| \\ & \leq \sum_{j < t(n)} \text{Adv}_j \\ & \leq t(n) \times \text{negl}(n) = \text{negl}(n) \end{aligned}$$

(QED, Claim 1)

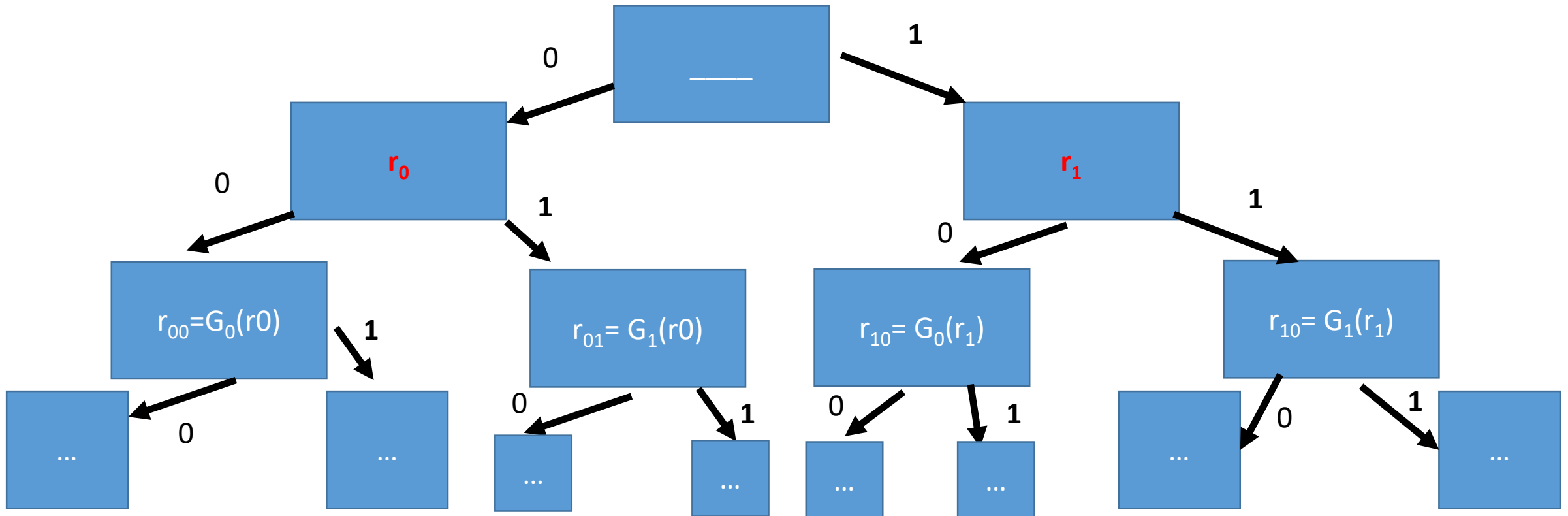
Hybrid H_1 and H_2

- Original Construction: Hybrid H_1



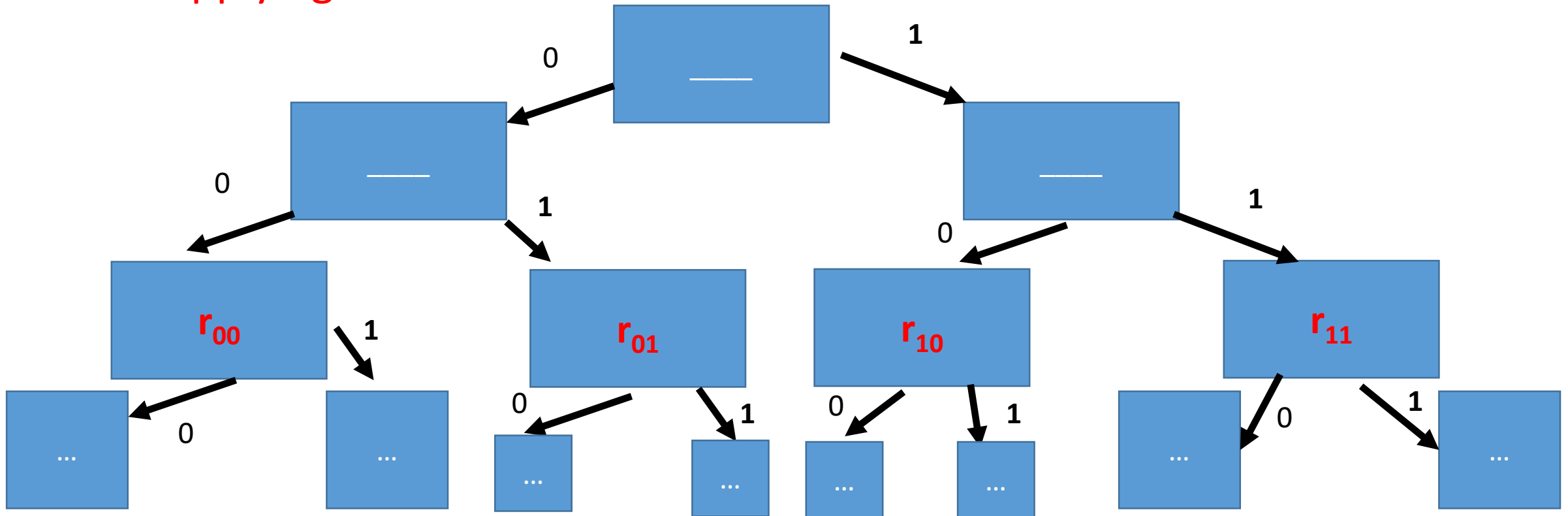
Hybrid H_1 and H_2

- Modified Construction H_2 : Pick r_0 and r_1 randomly instead of $r_i = G_i(K)$



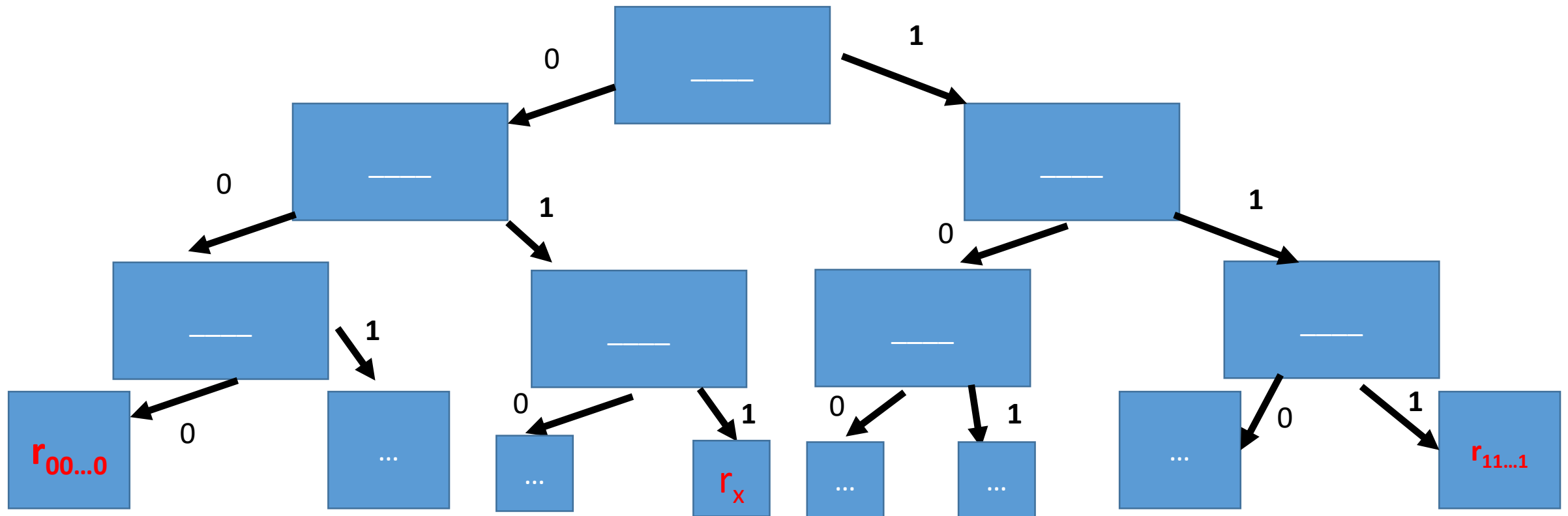
Hybrid H₃

- **Modified Construction H_3** : Pick r_{00} , r_{01} , r_{10} and r_{11} randomly instead of applying PRG



Hybrid H_n

- Truly Random Function: All output values r_x are picked randomly



Hybrid H_1 vs H_2

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \text{negl}(n)$$

Claim 2: *Attacker who makes $t(n)$ queries to F_k (or f) cannot distinguish H_2 from the real game (except with negligible probability).*

Proof Intuition: Follows by Claim 1

Hybrid H_i vs H_{i-1}

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \dots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \dots \parallel G(s_{t(n)}))] \right| < \text{negl}(n)$$

Claim 3: Attacker who makes $t(n)$ queries to F_k (or f) cannot distinguish H_i from H_{i-1} the real game (except with negligible probability).

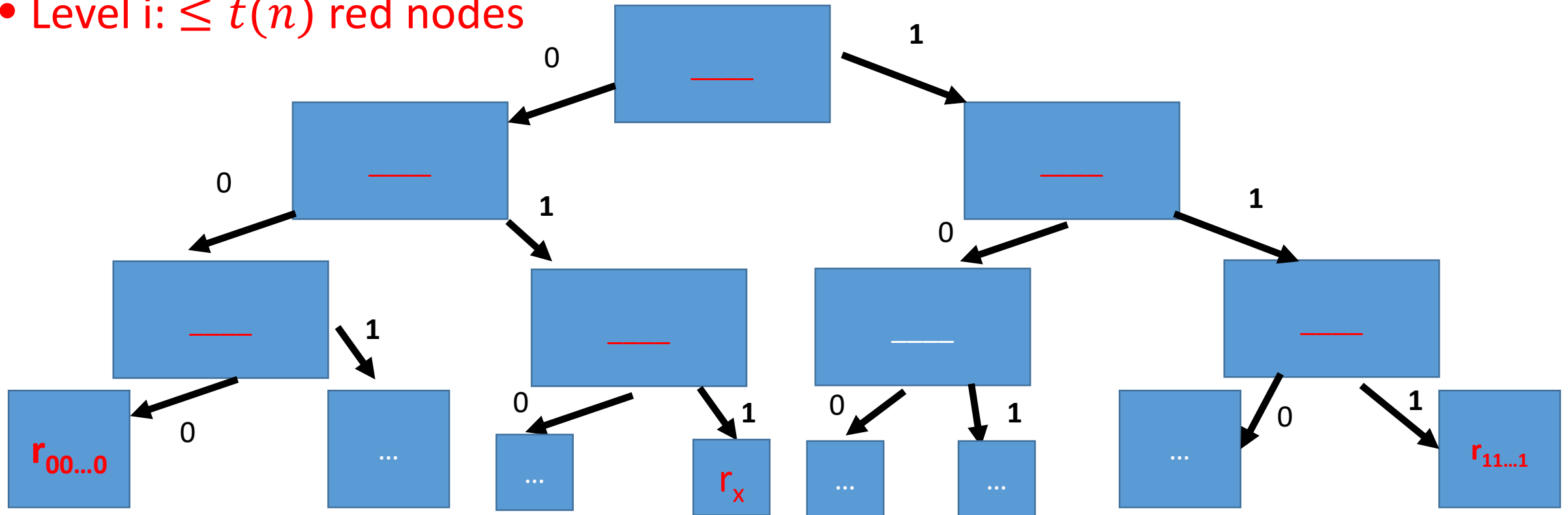
Challenge: Cannot replace 2^i pseudorandom values with random strings at level i

$2^i \text{negl}(n)$ is not necessarily negligible if $i = \frac{n}{2}$

Key Idea: Only need to replace $t(n)$ values (note: $t(n)\text{negl}(n)$ is negligible).

Hybrid H_i

- Red Leaf Nodes: Queried $F_k(x)$ (at most $t(n)$ red leaf nodes)
- Red Internal Nodes: On path from red leaf node to root
- Level i : $\leq t(n)$ red nodes



Hybrid H_1 vs H_2

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \text{negl}(n)$$

Claim 2: *Attacker who makes $t(n)$ oracle queries to our function cannot distinguish H_i from H_{i+1} (except with negligible probability).*

Proof: Indistinguishability follows by Claim 1

Let x_1, \dots, x_t denote the t queries. Let y_1, \dots, y_t denote first i bits of each query.

(H_{i+1} vs H_i : replaced $G(r_{y_i})$ with $r_{y_i \parallel 0} \parallel r_{y_i \parallel 1}$)

Hybrid H_i vs H_i

Claim 1: For any $t(n)$ and any PPT attacker A we have

$$\left| \Pr[A(r_1 \parallel \cdots \parallel r_{t(n)})] - \Pr[A(G(s_1) \parallel \cdots \parallel G(s_{t(n)}))] \right| < \textit{negl}(n)$$

Claim 3: Attacker who makes $t(n)$ queries to F_k (or f) cannot distinguish H_i from H_{i-1} the real game (except with negligible probability).

Triangle Inequality: Attacker who makes $t(n)$ queries to F_k (or f) *cannot* distinguish H_1 (real construction) from H_n (truly random function) except with negligible probability.

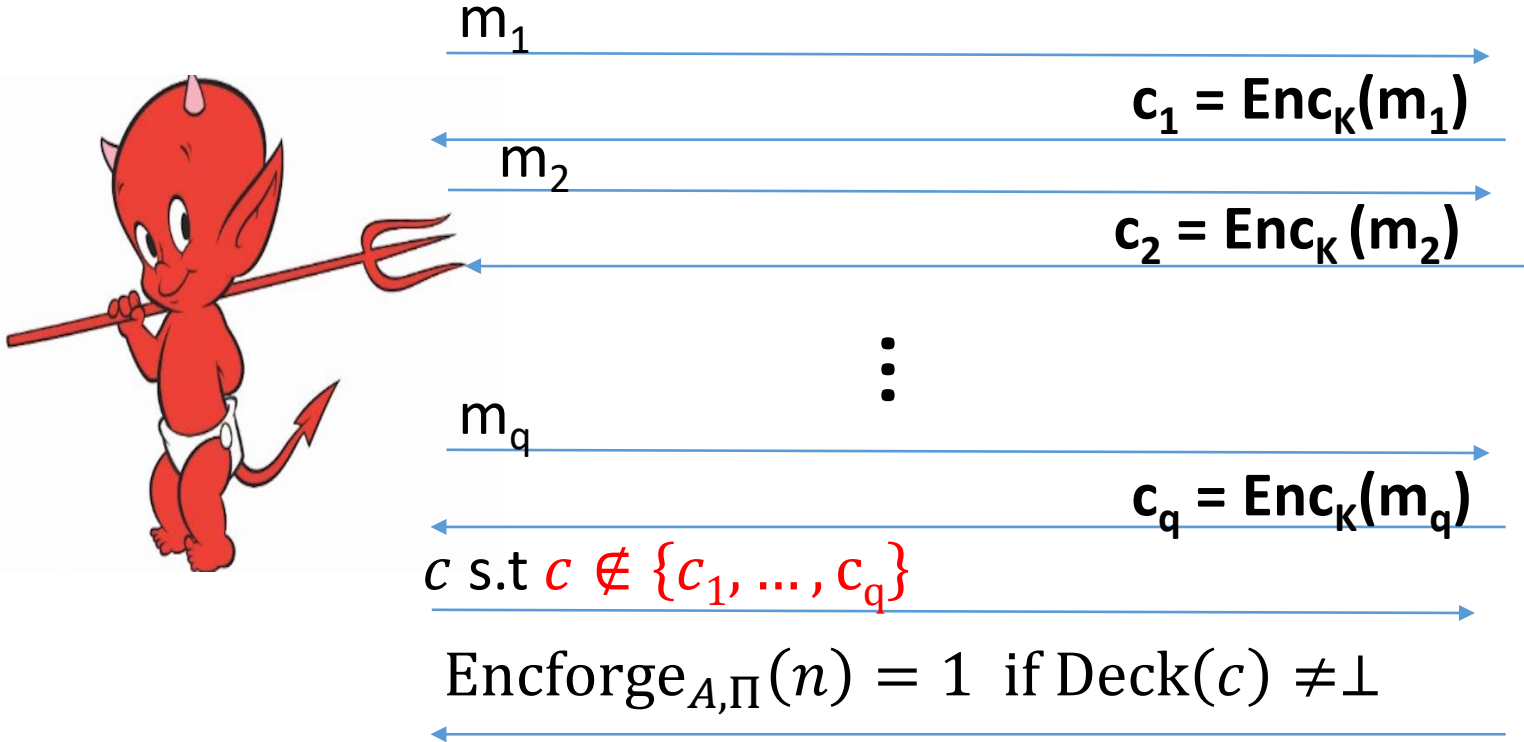
Authenticated Encryption

Encryption: Hides a message from the attacker

Message Authentication Codes: Prevents attacker from tampering with message



Unforgeable Encryption Experiment ($\text{Encforge}_{A,\Pi}(n)$)



$K = \text{Gen}(\cdot)$



$$\forall PPT A \exists \mu \text{ (negligible) s.t.}$$
$$\Pr[\text{Encforge}_{A,\Pi}(n) = 1] \leq \mu(n)$$

Unforgeable Encryption Experiment ($\text{Encforge}_{A,\Pi}(n)$)



Call Π an **authenticated encryption scheme** if it is CCA-secure and any PPT attacker wins Encforge with negligible probability

$$\forall \mu(n) = \text{negligible} \text{ s.t. } \Pr[\text{Encforge}_{A,\Pi}(n) = 1] \leq \mu(n)$$

$$c_1 = \text{Enc}_k(m_1)$$

$$(m_q)$$

$$\neq \perp$$

Game is very similar to MAC-Forge game



Building Authenticated Encryption

Attempt 4: (Encrypt-then-authenticate) Let $\text{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\text{Mac}'_{K_M}(m)$ be a secure MAC. Let $K = (K_E, K_M)$ then

$$\text{Enc}_K(m) = \langle c, \text{Mac}'_{K_M}(c) \rangle \text{ where } c = \text{Enc}'_{K_E}(m)$$

Secure?



Building Authenticated Encryption

Theorem: (Encrypt-then-authenticate) Let $\text{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\text{Mac}'_{K_M}(m)$ be a secure MAC. Then the following construction is an authenticated encryption scheme.

$$\text{Enc}_K(m) = \langle c, \text{Mac}'_{K_M}(c) \rangle \text{ where } c = \text{Enc}'_{K_E}(m)$$

Proof?

Two Tasks:

$\text{Encforge}_{A,\Pi}$
CCA-Security

Building Authenticated Encryption

Theorem: (Encrypt-then-authenticate) Let $\text{Enc}'_{K_E}(m)$ be a CPA-Secure encryption scheme and let $\text{Mac}'_{K_M}(m)$ be a secure MAC. Then the following construction is an authenticated encryption scheme.

$$\text{Enc}_K(m) = \langle c, \text{Mac}'_{K_M}(c) \rangle \text{ where } c = \text{Enc}'_{K_E}(m)$$

Proof Intuition: Suppose that we have already shown that any PPT attacker wins $\text{Encforge}_{A,\Pi}$ with negligible probability.

Why does CCA-Security now follow from CPA-Security?

CCA-Attacker has decryption oracle, but cannot exploit it! Why?

Always sees \perp “invalid ciphertext” when he query with unseen ciphertext

Proof Sketch

1. Let ValidDecQuery be event that attacker submits new/valid ciphertext to decryption oracle
2. Show $\Pr[\text{ValidDecQuery}]$ is $\text{negl}(n)$ for any PPT attacker
 - Hint: Follows from strong security of MAC since
$$\text{Enc}_K(m) = \langle c, \text{Mac}'_{K_M}(c) \rangle$$
 - This also implies unforgeability (even if we gave the attacker K_E !).
3. Show that attacker who does not issue valid decryption query wins CCA-security game with probability $\frac{1}{2} + \text{negl}(n)$
 - Hint: otherwise we can use A to break CPA-security
 - Hint 2: simulate decryption oracle by always returning \perp when given new ciphertext