Advanced Cryptography
CS 655

Week 4:

- Generic Group Model/Ideal Permutation
- Pre-Computation Attacks
- Bit-Fixing Model/Auxiliary Input
- Compression Arguments

Spring 2023

Idealized Models of Computation

- Random Oracle Model
  - All parties have oracle access to a truly random function $H(\cdot)$
- Ideal Permutation Model
  - All parties have access to a truly random permutation $f(\cdot)$ and its inverse $f^{-1}(\cdot)$
- Ideal Cipher Model
  - All parties have oracle access to $E(\cdot,\cdot)$ and $E^{-1}(\cdot\ ,\ \cdot)$
  - For any fixed key K the function $E_K(x) \coloneqq E(K,x)$ is a truly random permutation and $E_K^{-1}(x) \coloneqq E^{-1}(K,x)$ is the inverse

- Generic Group Model [Shoup 97]

Warm-Up

- In the random oracle model we are given $\mathrm{y} = H(x)$ for a random value x. The attacker can make q queries to the random oracle. What is the probability that the attacker can find a pre-image of $\mathrm{y} \in \{0,1\}^{\lambda}$?

- In the ideal-permutation model we are given $\mathrm{y} = f(x)$ for a random value x. What is the probability that the attacker can find a pre-image of $\mathrm{y} \in \{0,1\}^{\lambda}$ after at most q oracle queries?

Warm-Up

- In the random oracle model we are given $\mathrm{y} = H(x)$ for a random value x. The attacker can make q queries to the random oracle. What is the probability that the attacker can find a pre-image of $\mathrm{y} \in \{0,1\}^{\lambda}$?
  - **Answer:** At most $2\mathrm{q} \times 2^{-\lambda}$. Let $x_1, \dots, x_q$ denote the queries the attacker makes. The probability one of the q queries is x is $\Pr\left[\mathrm{x} \in \{x_1, \dots, x_q\}\right] \leq \mathrm{q} \times 2^{-\lambda}$. Given that $\mathrm{x} \notin \{x_1, \dots, x_q\}$ we can view each $H(x_1)$ as a uniformly random string. Thus, we have $\Pr\left[\mathrm{y} \in \{H(x_1), \dots, H(x_q)\}|\right] \leq \mathrm{q} \times 2^{-\lambda}$

- In the ideal-permutation model we are given $\mathrm{y} = f(x)$ for a random value x. What is the probability that the attacker can find a pre-image of $\mathrm{y} \in \{0,1\}^{\lambda}$ after at most q oracle queries?
  - **Answer:** There is a trivial attack using q=1 queries!

$$\mathrm{x} = f^{-1}(y)$$

Warm-Up: Part 2

- In the Ideal-Cipher Model we are given $\left(m, E_K(m)\right)$ where $\mathrm{K} \in \{0,1\}^{\lambda}$ is random. The attacker may make q queries to the ideal cipher. What is the probability that the attacker can find K?

Warm-Up: Part 2

- In the Ideal-Cipher Model we are given $(m, E_K(m))$ where $\mathrm{K} \in \{0,1\}^\lambda$ is random. The attacker may make q queries to the ideal cipher. What is the probability that the attacker can find K?
  - **Answer:** At most $\mathrm{q} \times 2^{-\lambda} + \frac{1}{2^\lambda - q}$ (the probability of making a query of the form $E(K,.)$ plus the probability of guessing the correct key out of the remaining $2^\lambda - q$ options if this query does not happen).

- **Challenge:** In the ideal-permutation model we are given y_1 where $\mathrm{y} = (y_1, y_2) = f(x)$ for a random value x. What is the probability that the attacker can finds x (or y_2) after at most q oracle queries?

What Can We Do with Ideal Permutation?

- **Answer 1: Build a Block-Cipher**
- Evan-Mansor Block Cipher
  - Key: $K = (K_1, K_2)$
  - $EM_{f,K}(x) \coloneqq f(K_1 \oplus x) \oplus K_2$
  - $EM_{f,K}{}^{-1}(y) \coloneqq f^{-1}(K_2 \oplus y) \oplus K_1$
  - Dunkelman et al. observed that one can safely use a single key $K_1 = K_2$ (see https://eprint.iacr.org/2011/541.pdf)

  - Security Game for Block-Cipher:
    - B=0 (real world): Attacker is given oracle access to $f(.)$, $f^{-1}(y)$ and $EM_K^f(\cdot)$ and $EM_{f,K}{}^{-1}(\cdot)$ but not the secret key $K = (K_1, K_2)$
    - B=0 (ideal world) Attacker is given oracle access to $f(.)$, $f^{-1}(y)$ and $\pi(.)$ $\pi^{-1}(.)$ where $\pi(.)$ is truly random permutation (independent of $f(.)$)

What Can We Do with Ideal Permutation?

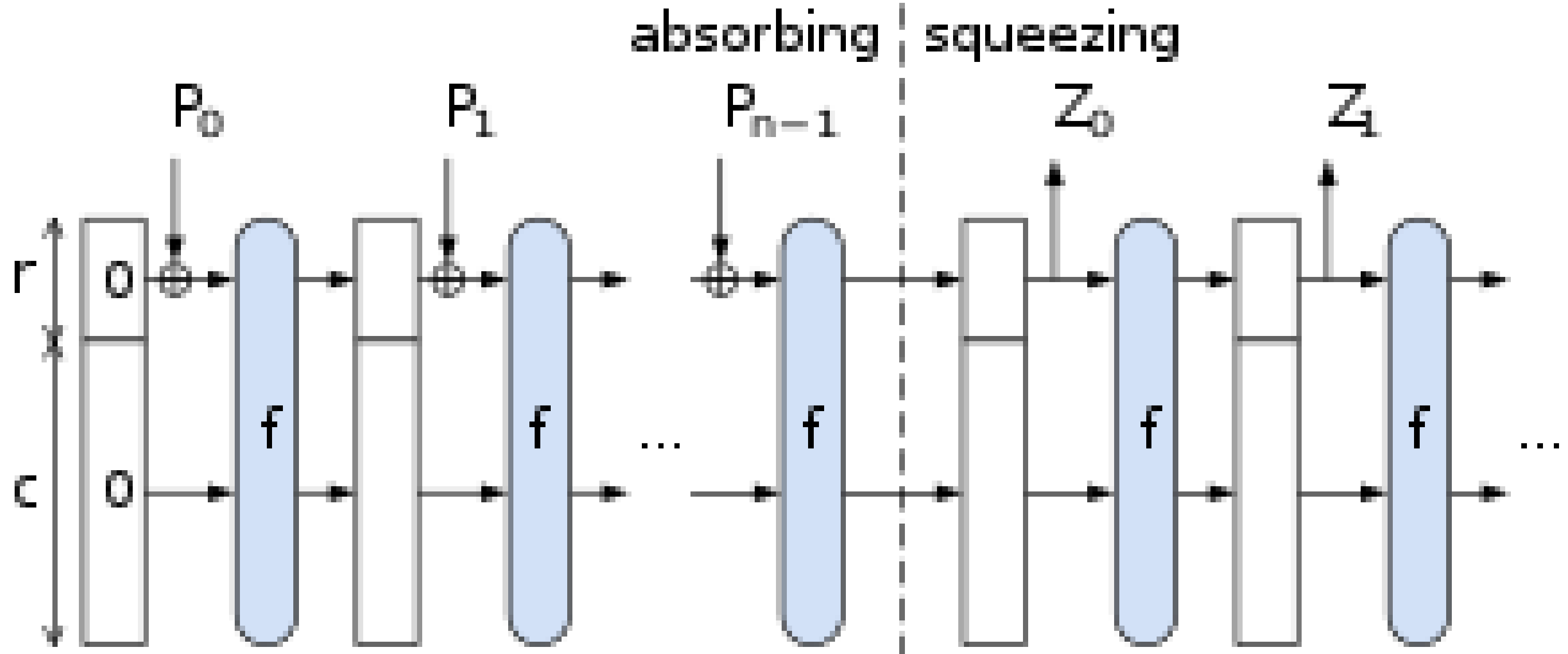
- Evan-Mansor Block Cipher
  - Key: $K = (K_1, K_2)$
  - $EM_{f,K}(x) \coloneqq f(K_1 \oplus x) \oplus K_2$
  - $EM_{f,K}^{-1}\quad(y) \coloneqq f^{-1}(K_2 \oplus y) \oplus K_1$
  - Dunkelman et al. observed that one can safely use a single key $K_1 = K_2$

  - Security Game for Block-Cipher:
    - B=0 (real world): Attacker is given oracle access to $f(.)$, $f^{-1}(y)$ and $EM_{f,K}(\cdot)$ and $EM_{f,K}^{-1}\quad(\cdot)$ but not the secret key $K = (K_1, K_2)$
    - B=0 (ideal world) Attacker is given oracle access to $f(.)$, $f^{-1}(y)$ and $\pi(.)\ \pi^{-1}(.)$ where $\pi(.)$ is truly random permutation (independent of $f(.)$)
  - Attacker's advantage is at most $O(\frac{q_f q_E}{2^\lambda})$ where q_f (resp.) denotes the number of queries to f or f^{-1} (resp. $EM_{f,K}$ or $EM_{f,K}^{-1}\quad$)

What Can We Do with Ideal Permutation?

- **Answer 2: Build a Collision-Resistant Hash Function**
  - Sponge Construction: SHA3
  - **Input:** $\mathrm{P} = (P_0, P_1, \ldots, P_{n-1})$ viewed as r-bit blocks e.g.,
  - Input “absorbed” in multiple rounds
  - Output squeezed out in subsequent rounds

Keccak (SHA3): |r|+|c|=1600-bit state
$c \in \{256, 512, \ldots\}$



Pre-Processing Attacks

- Often times the same cipher/permutation/group/hash function is used across multiple applications
- Adversary with nation-state level resources might spend a lot of time pre-computing hints to help break protocols using these building blocks

- Auxiliary-Input Attacker Model $A = (A_1, A_2)$
- Offline attacker A_1 is unbounded and outputs an S-bit hint for online attacker A_2
- A_2 will try to win security games using this hint

Pre-Processing Attacks: Trivial Example

- Auxiliary-Input Attacker Model $A = (A_1, A_2)$
- Offline attacker A_1 is unbounded, and can find collisions for our random oracle H by brute-force.
- **Output Hint:** x_1 and x_2 such that $\mathrm{H}(x_1)$ and $\mathrm{H}(x_2)$
- A_2 can trivially find a collision using this hint.

- However, we may still hope that $A_2(s, hint)$ cannot find x and x' such that $\mathrm{H}(s, x_1)$ and $\mathrm{H}(s, x_2)$ given a random salt s (picked after pre-processing)

Pre-Processing Lower Bounds

- Auxiliary-Input Attacker Model $A = (A_1, A_2)$
- Offline attacker A_1 is unbounded and outputs an S-bit hint for online attacker A_2
- A_2 will try to win security games using this hint

- Can be difficult! We can no longer assume that H(x) looks uniformly random to online attacker (due to hint)
- **Compression Technique:** If online attacker is too successful then we may be able to ``compress" H. (Compressing a random string is impossible). These arguments are very tricky!

Auxiliary-Input Attacker Model

- **Auxiliary-Input Attacker Model** $A = (A_1, A_2)$
- **Random Oracle Version:**
- Offline attacker A_1 is unbounded and outputs an S-bit hint for online attacker A_2 after viewing entire truth table $H(.)$
- A_2 will try to win security games using this hint

- (S,T,p)-attacker
    - A_1 outputs a S-bit hint
    - A_2 makes at most T random oracle queries
    - A_2 may be constrained in other ways (space/time/signing queries etc…) as specified by parameters p.
- $\left((S,T,p),\varepsilon\right)$-security ➔ Any (S,T,p) attacker wins with advantage at most ε

Generic Group Model (GGM)

- Models generic attacks [Shoup 97]
  - don't exploit structure of cyclic Group $G = \langle g \rangle$
  - WLOG assume $G = \mathbb{Z}_p$
- Attacker can only manipulate group elements $\mathrm{x} \in \mathbb{Z}_p$ via the following oracles:

$$\mathrm{mult}(\tau(x), \tau(y)) = \tau(x + y)$$

Input: handles for group elements $\mathrm{x, y} \in \mathbb{Z}_p$ **Output:** handles for group element $\mathrm{x + y} \in \mathbb{Z}_p$

Where $\tau\colon \mathbb{Z}_p \to \{0,1\}^{2k}$ is a random injective function mapping group elements to binary strings (handles)

Generic Group Model (GGM)

- Models generic attacks [Shoup 97]
  - Generic attacks don't exploit structure of cyclic Group $G = \langle g \rangle$
  - WLOG assume $G = \mathbb{Z}_p$
- Attacker can only manipulate group elements $\mathrm{x} \in \mathbb{Z}_p$ via the following oracles:

$$\mathrm{mult}(\tau(x), \tau(y)) = \tau(x+y)$$

$$\mathrm{inv}(\tau(x)\ ) = \tau(-x)$$

$$\mathrm{pow}(\tau(x), n\ ) = \tau(nx)$$

Output: handle for group element $(\mathrm{nx \ mod\ p}) \in \mathbb{Z}_p$

Input: handle for group elements $\mathrm{x} \in \mathbb{Z}_p$ and integer n

Where $\tau: \mathbb{Z}_p \rightarrow \{0,1\}^{2k}$ is a random injective function mapping group elements to binary strings (handles)

Sample GGM Result [Shoup 97] (Discrete Log): any attacker making T GGM queries solves discrete log problem with probability at most $\mathcal{O}\left(\frac{T^2}{2^{2k}}\right)$

Generic Group Model

- Typically we use an Elliptic Curve Group of prime order p for $p \approx 2^{2\lambda}$
  - Provides λ-bit security

- **Discrete Log Problem:** Given generator g and g^x find x
- **Generic Group Version:** Given $\tau(1)$ and $\tau(x)$ find x
  - Best Generic Attack (Baby-Step Giant Step):
    - 1) Compute $y_k = g^{k2^\lambda}$ for each $k \leq 2^\lambda$ (Time: $\tilde{O}(2^\lambda)$)
    - 2) For each $x_k = g^{x+k}$ for each $k \leq 2^\lambda$ (Time: $\tilde{O}(2^\lambda)$)
    - 3) Find intersection (i,j) such that $x_i = g^{x+i} = y_j = g^{j2^\lambda}$ and solve $x = j2^\lambda - i$
    - Note: $x = j2^\lambda - i$ for some pair $i, j \leq 2^\lambda$
- **Generic Group Version Lower Bound: Any generic attacker making q queries to GGM oracles succeeds with probability at most**

$$O\left(\frac{q^2}{2^{2\lambda}}\right)$$

Generic Group Lower Bound

- **Discrete Log Problem:** Given generator g and g^x find x
- **Generic Group Version:** Given $\tau(1)$ and $\tau(x)$ find x

- **Generic Group Version Lower Bound: Any generic attacker making q queries to GGM oracles succeeds with probability at most**

$$O\left(\frac{q^2}{2^{2\lambda}}\right)$$

- **Proof Sketch:** Initialize two sets $K = \{(1, \tau(1))\}$ and $U = \{(x, \tau(x))\}$
  - K ➔ Discrete Log Known
  - U ➔ Discrete Log Depends on Unknown x
  - $\mathrm{mult}(\tau(1), \tau(1)) = \tau(2)$ ➔ Add $(2, \tau(2))$ to K
  - $\mathrm{mult}(\tau(x), \tau(x)) = \tau(2x)$➔ Add $(2x, \tau(x+1))$ to U
  - $\mathrm{mult}(\tau(x), \tau(1)) = \tau(x+1)$➔ Add $(x+1, \tau(x+1))$ to U
  - Each new query adds item to K or U
  - Cannot learn x unless sets intersect e.g., $\mathrm{mult}(\tau(x), \tau(x+1)) = \tau(2x+1)$ is found in K
  - Sets remain disjoint with probability$\frac{|U||K|}{2^{2\lambda}} \le \frac{q^2}{2^{2\lambda}}$

Generic Group Model

- Typically we use an Elliptic Curve Group of prime order p for $p \approx 2^{2\lambda}$
  - Provides λ-bit security

- **Discrete Log Problem:** Given generator g and g^x find x
- **Generic Group Version:** Given $\tau(1)$ and $\tau(x)$ find x

- **Computational-Diffie Hellman:** Given g, g^x and g^y find g^{xy}
- **Generic Group Version:** Given $\tau(1), \tau(x)$ and $\tau(y)$ find $\tau(xy)$
- **Similar Proof: Any generic attacker making q queries to GGM oracles succeeds with probability at most**

$$O\left(\frac{q^2}{2^{2\lambda}}\right)$$

Generic Group Model with Preprocessing

- **Offline Attacker:** $\mathrm{A}(\tau) = \sigma$
  - Input: the secret/random encoding function τ for our group $\mathbb{Z}_p$
  - Output: S-bit hint $\sigma \in \{0,1\}^S$ for online attacker
  - No bound on the running time for the offline attacker

- **Online Attacker:** May use hint σ during attack
  - Bounded running time T, q_{GO} queries to generic group oracles etc…
  - May use hint σ during attack

- **Motivation:**
  - Handful of groups (NIST P-256, Curve25519 etc…) used by most real-world cryptosystems
  - Offline phase of preprocessing attack is only executed once

Sample Result [CK18] (Discrete Log with Preprocessing): any preprocessing attacker making solves discrete log problem with probability at most $\mathcal{O}\left(\frac{ST^2}{2^{2k}}\right)$

GGM + ROM with Preprocessing

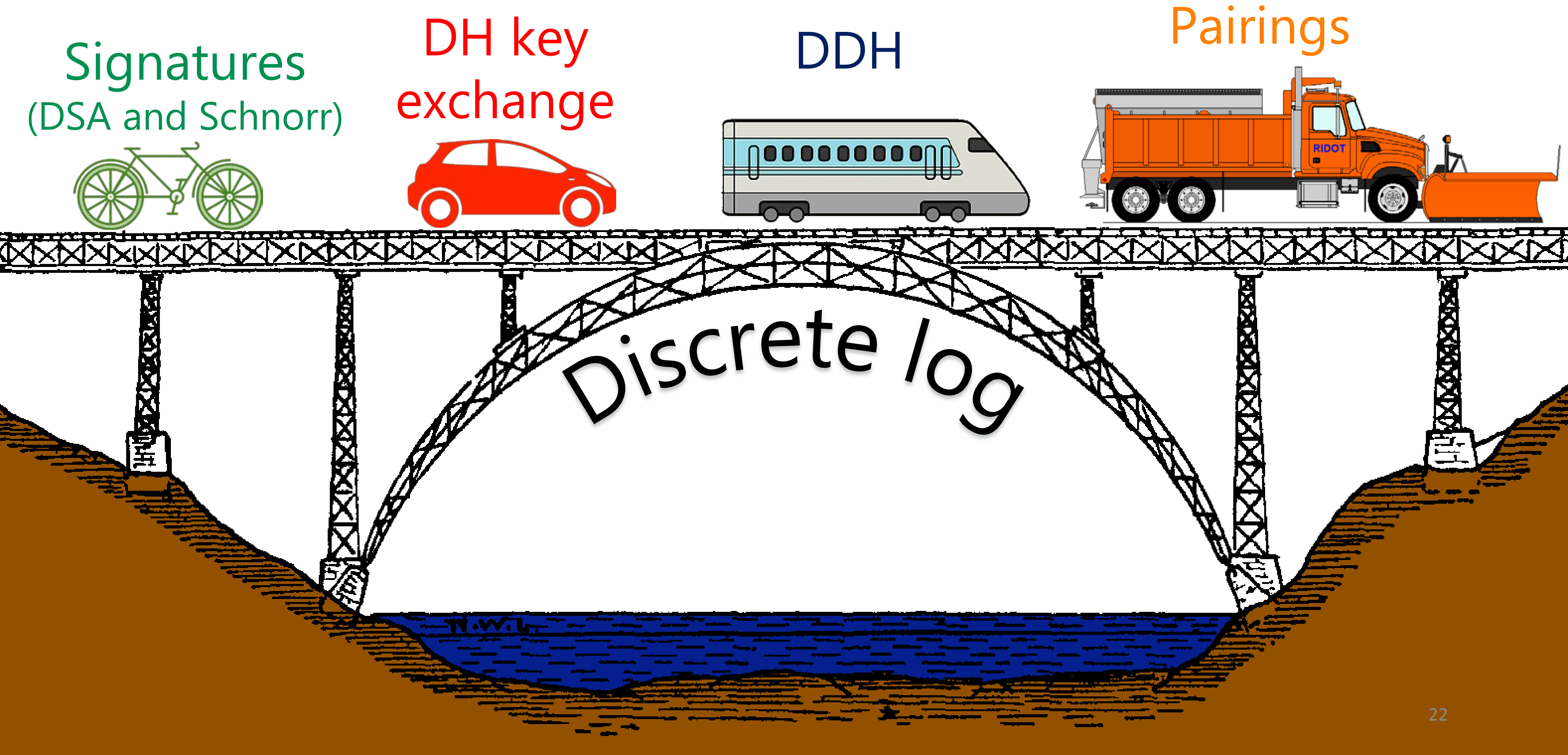
- **Offline Attacker:** $\mathrm{A}^{H}(\tau) = \sigma$
  - **Input:** the secret/random encoding function τ for our group $\mathbb{Z}_p$, oracle access to the random oracle H
  - **Output:** S-bit hint $\sigma \in \{0,1\}^S$ for online attacker
  - The offline attacker may make a very large number of random oracle queries e.g., 2^{3k}
  - Unbounded running time

- **Online Attacker:** May use hint σ during attack
  - Bounded running time T, q_{GO} (resp. q_{RO}) queries to group oracle (resp. random oracle) etc…
  - May use hint σ during attack

The Discrete Logarithm Problem with Preprocessing

Henry Corrigan-Gibbs and Dmitry Kogan
Stanford University

Eurocrypt – 1 May 2018
Tel Aviv, Israel


Signatures
(DSA and Schnorr)
DH key exchange
DDH
Pairings
RIDOT
Discrete log

The discrete-log problem

Group: $\mathbb{G} = \langle g \rangle$
of prime order N

Instance: $g^x \in \mathbb{G}$

Adversary $\mathcal{A}$

Solution:
$x \in \mathbb{Z}_N$

Why do we believe this problem is hard?

Generic lower bounds give us confidence

Theorem. [Shoup'97] Every **generic** discrete-log algorithm that

- operates in a group of prime order N and
- succeeds with probability at least ½

must run in time $\Omega(N^{1/2})$.

Generic attack in 256-bit group takes $\approx 2^{128}$ time.

Best attacks on standard EC groups are generic

Generic algorithms can only make "black-box" use of the group operation

Generic-group model:

- Group is defined by an injective **"labeling" function**
$$\sigma\colon\ \mathbb{Z}_N \to \{0,1\}^*$$
- Algorithm has access to a **group-operation oracle**:
$$\mathcal{O}_\sigma\big(\sigma(i),\sigma(j)\big) \ \mapsto \ \sigma(i+j)$$

Generic dlog algorithm takes as input $(\sigma(1)$ … make queries to $\mathcal{O}_\sigma$, outputs x.

[Measure running time by query complex…

Very useful way to understand hardness

[BB04,B05,M05,D06, B08,Y15,...]

Existing generic lower bounds **do not account for preprocessing**

- Premise of generic-group model: the adversary **knows nothing** about the structure of the group $\mathbb{G}$ in advance
- In reality: the adversary knows a lot about $\mathbb{G}$!
  - $\mathbb{G}$ is one of a small number of groups: NIST P-256, Curve25519, …
- A realistic adversary can perform $\mathbb{G}$-specific preprocessing!
- Existing generic-group lower bounds say nothing about preprocessing attacks! [H80, Yao90, FN91, …]

Preprocessing phase

Group: $\mathbb{G} = \langle g \rangle$

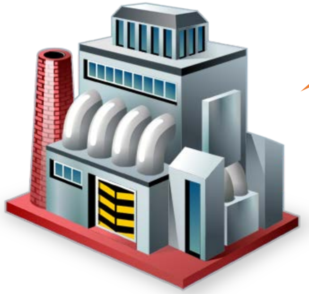
$\mathcal{A}_0$

Advice: $st_{\mathbb{G}}$

Online phase

Both algorithms are generic!

Instance: $g^x \in \mathbb{G}$



$\mathcal{A}_1$

Solution: $x \in \mathbb{Z}_N$

Initiated by Hellman (1980) in context of OWFs

Preprocessing phase

Preprocessing time P

Group: $\mathbb{G} = \langle g \rangle$

$\mathcal{A}_0$

Advice: $st_{\mathbb{G}}$

Advice size S

Online phase

Online time T

Instance: $g^x \in \mathbb{G}$

$\mathcal{A}_1$

Solution: $x \in \mathbb{Z}_N$

Success prob. ϵ

Initiated by Hellman (1980) in context of OWFs

- Preexisting $S = T = \tilde{O}(N^{1/3})$ generic attack on discrete log

- The $\tilde{O}(N^{1/3})$ generic dlog attack is optimal
- Any such attack must use <u>lots</u> of preprocessing: $\Omega(N^{2/3})$
- New $\tilde{O}(N^{1/5})$ preprocessing attack on DDH-like problem

A preexisting result…

Theorem. [Mihalcik 2010] [Lee, Cheon, Hong 2011] [Bernstein and Lange 2013]
There is a generic dlog algorithm with preprocessing that:

- uses S bits of group-specific advice,
- uses T online time, and
- succeeds with probability ϵ,

such that:

$$ST^2 = \tilde{O}(\epsilon N)$$

Will sketch the algorithm for $S = T = N^{1/3}$, constant ϵ.

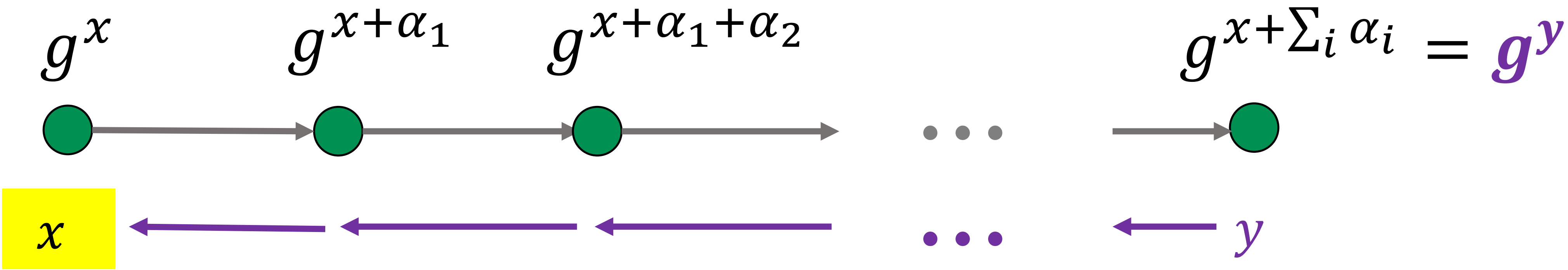
…. building on prior work on multiple-discrete-log algorithms [ESST99,KS01,HMCD04,BL12]

Preliminaries

Define a pseudo-random walk on $\mathbb{G}$:

$$g^x \mapsto g^{x+\alpha} \quad \text{where} \quad \alpha = \text{Hash}(g^x)$$

is a random function



If you know the dlog of the endpoint of a walk,
you know the dlog of the starting point!

[M10, LCH11, BL13]

Preprocessing phase

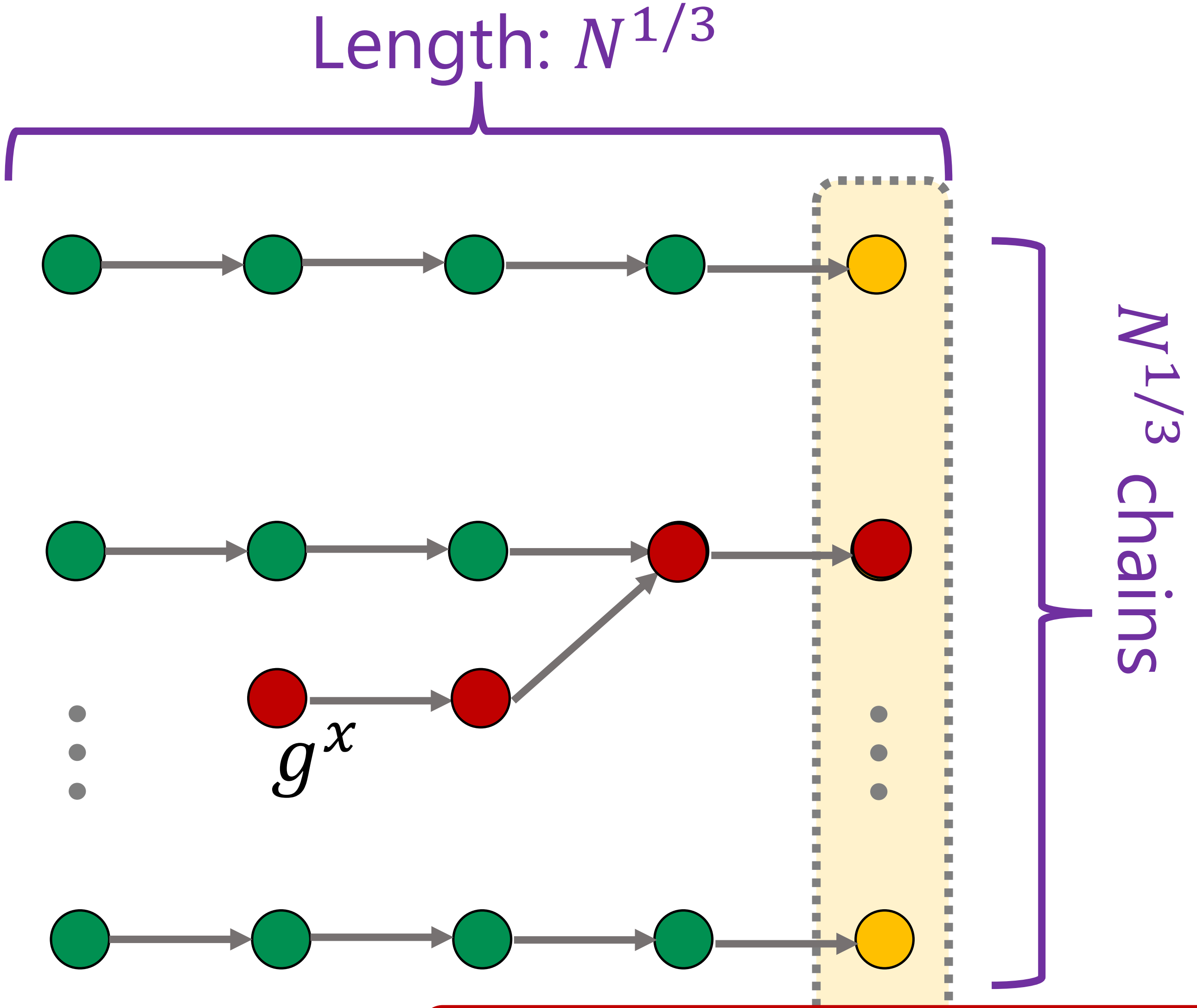
- Build $N^{1/3}$ chains of length $N^{1/3}$
- Store dlogs of chain endpoints

Advice: $\tilde{O}(N^{1/3})$ bits

Online phase

- Walk $O(N^{1/3})$ steps
- When you hit a stored point, output the discrete log

Time: $\tilde{O}(N^{1/3})$ steps



Preprocessing time: $\tilde{\Omega}(N^{2/3})$

[M10, LCH11, BL13]

Generic discrete log

→ Without preprocessing: $\Omega(N^{1/2})$

→ With preprocessing: $\tilde{O}(N^{1/3})$

256-bit ECDL

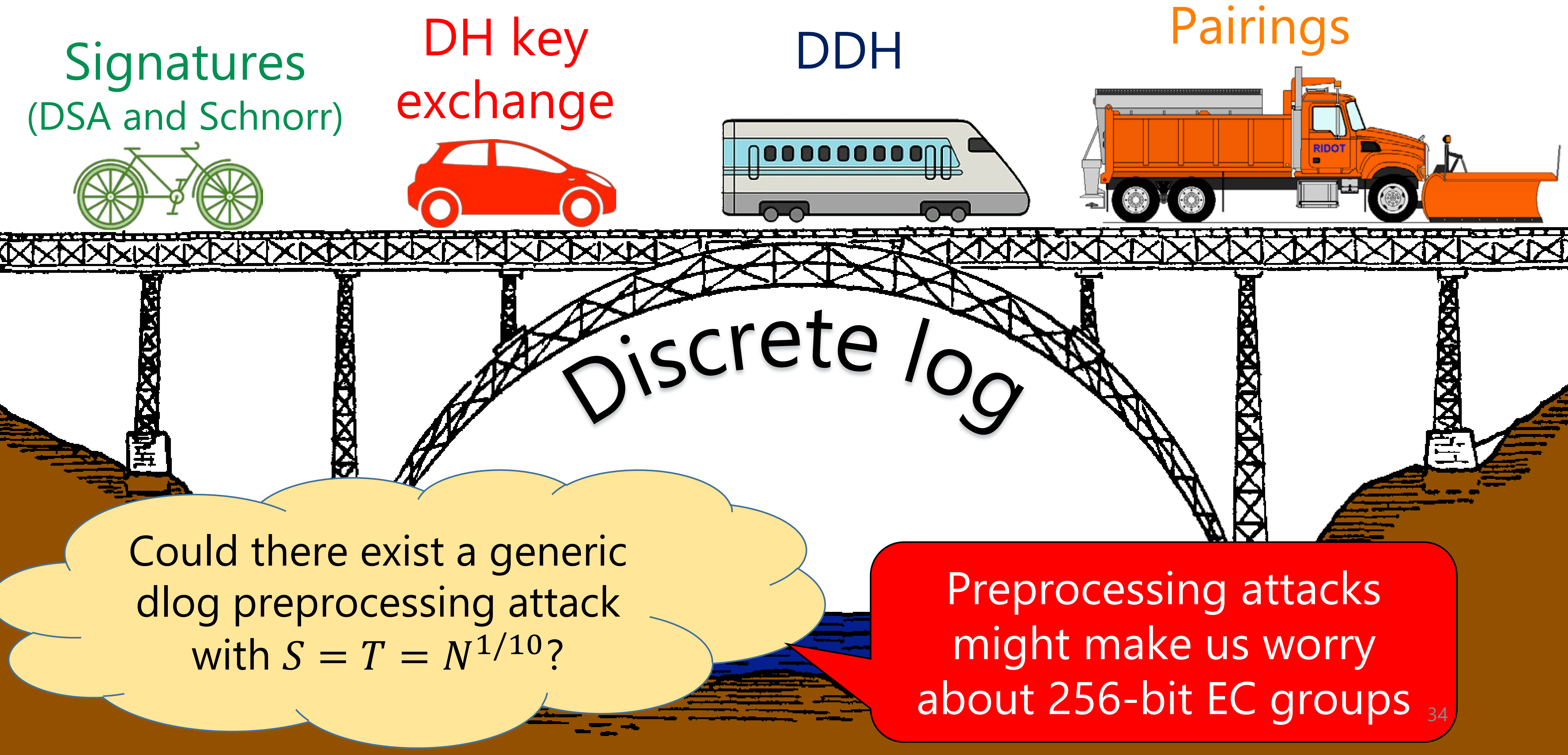
2^{128} time

2^{86} time

Is this dlog attack the best possible?!

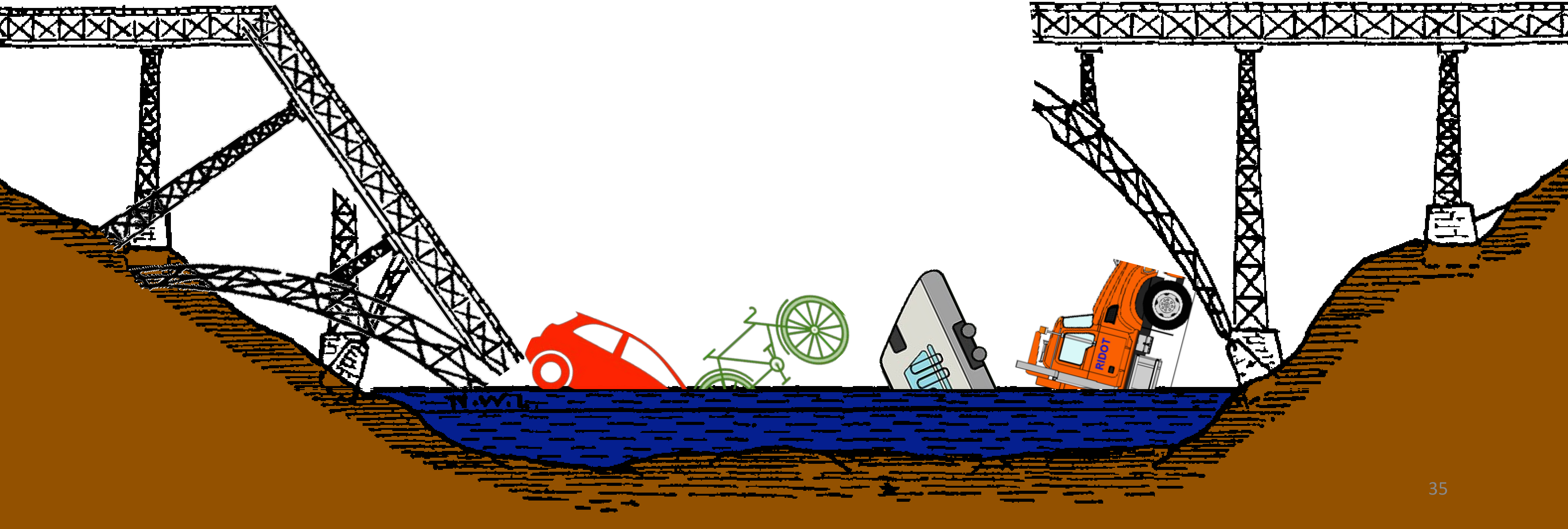
Related preprocessing attacks break:

- Multiple discrete log problem [This paper]
- One-round Even-Mansour cipher [FJM14]
- Merkle-Damgård hash with random IV [CDGS17]



Could there exist a generic dlog preprocessing attack with $S = T = N^{1/10}$?

Preprocessing attacks might make us worry about 256-bit EC groups


N.W.L.
RIDOT

- Preexisting $S = T = \tilde{O}(N^{1/3})$ generic attack on discrete log

- The $\tilde{O}(N^{1/3})$ generic dlog attack is optimal
- Any such attack must use lots of preprocessing: $\Omega(N^{2/3})$
- New $\tilde{O}(N^{1/5})$ preprocessing attack on DDH-like problem

Theorem. [Our paper]
Every generic dlog algorithm with preprocessing that:
- uses S bits of group-specific advice,
- uses T online time, and
- succeeds with probability ϵ,

must satisfy:

$$ST^2 = \tilde{\Omega}(\epsilon N)$$

This bound is tight for the full range of parameters (up to log factors)

Shoup's proof technique (1997) relies on $\mathcal{A}$ having no information about the group $\mathbb{G}$ when it starts running

→ Need different proof technique

Theorem. [Our paper]

Every generic dlog algorithm with preprocessing that:

- uses S bits of group-specific advice,
- uses T online time, and
- succeeds with probability ϵ,

must satisfy:

$$ST^2 = \tilde{\Omega}(\epsilon N)$$

Online time $N^{1/3}$ implies $\Omega(N^{2/3})$ preprocessing

Theorem. [Our paper]

Furthermore, the preprocessing time P must satisfy

$$PT + T^2 = \Omega(\epsilon N)$$

Open questions and recent progress

- Tightness of DDH upper/lower bounds?
  - Is it as hard as dlog or as easy as sqDDH?
- Non-generic preprocessing attacks on ECDL?
  - As we have for $\mathbb{Z}_p^*$

Coretti, Dodis, and Guo (2018)

- Elegant proofs of generic-group lower bounds using "presampling" (à la Unruh, 2007)
- Prove hardness of "one-more" dlog, KEA assumptions, …

Auxiliary-Input Attacker Model

- **Auxiliary-Input Attacker Model** $A = (A_1, A_2)$
- **Random Oracle Version:**
- Offline attacker A_1 is unbounded and outputs an S-bit hint for online attacker A_2 after viewing entire truth table $H(.)$
- A_2 will try to win security games using this hint

- (S,T,p)-attacker
  - A_1 outputs a S-bit hint
  - A_2 makes at most T random oracle queries
  - A_2 may be constrained in other ways (space/time/signing queries etc…) as specified by parameters p.
- $\left((S,T,p),\varepsilon\right)$-security ➔ Any (S,T,p) attacker wins with advantage at most ε

Bit-Fixing Model for Pre-Processing Attacks

- **Auxiliary-Input Attacker Model** $A = (A_1, A_2)$
- **Random Oracle Version:**
- Offline attacker A_1 fixes output of random oracle $H(.)$ at P locations and then outputs a S-bit hint.
- A_2 initially knows nothing about remaining unfixed values i.e., $H(x)$ picked randomly for $x \notin P$ after A_1 generates hint
- (P,T,p)-attacker
  - A_1 fixes H on at most P locations and outputs S-bit hint
  - A_2 makes at most T random oracle queries
  - A_2 may be constrained in other ways (space/time/signing queries etc…) as specified by parameters p.
- $\left((S,T,p),\varepsilon\right)$-security ➔ Any (S,T,p) attacker wins with advantage at most ε

Bit-Fixing Model (Unruh)

- **Pro:** Much easier to prove lower bounds in Bit-Fixing Model
- **Con: Bit-Fixing model is n**ot a compelling model for pre-processing attacks

- **Usage:** Lower bound in bit-fixing model ➔ Lower bound in Auxilliary-Input Model

- This approach yields tight lower-bounds in the Auxilliary-Input Model for some applications ☺
- Other applications require a different approach (e.g., compression)

Oracles. An oracle $\mathcal{O}$ has two interfaces $\mathcal{O}.\mathsf{pre}$ and $\mathcal{O}.\mathsf{main}$, where $\mathcal{O}.\mathsf{pre}$ is accessible only once before any calls to $\mathcal{O}.\mathsf{main}$ are made. Oracles used in this work are:

- **Random oracle** $\mathsf{RO}(N, M)$**:** Samples a random function table $F \leftarrow \mathcal{F}_{N,M}$, where $\mathcal{F}_{N,M}$ is the set of all functions from $[N]$ to $[M]$; offers no functionality at $\mathcal{O}.\mathsf{pre}$; answers queries $x \in [N]$ at $\mathcal{O}.\mathsf{main}$ by the corresponding value $F[x] \in [M]$.

- **Auxiliary-input random oracle** $\mathsf{AI\text{-}RO}(N, M)$**:** Samples a random function table $F \leftarrow \mathcal{F}_{N,M}$; outputs F at $\mathcal{O}.\mathsf{pre}$; answers queries $x \in [N]$ at $\mathcal{O}.\mathsf{main}$ by the corresponding value $F[x] \in [M]$.

- **Bit-Fixing random oracle** $\mathsf{BF\text{-}RO}(P, N, M)$**:** Samples a random function table $F \leftarrow \mathcal{F}_{N,M}$; takes a list at $\mathcal{O}.\mathsf{pre}$ of at most P query/answer pairs that override F in the corresponding positions; answers queries $x \in [N]$ at $\mathcal{O}.\mathsf{main}$ by the corresponding value $F[x] \in [M]$.

- **Standard model:** Neither interface offers any functionality.

Definition 2. *An* (S, T)-attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the $\mathcal{O}$-model *consists of two procedures*

- $\mathcal{A}_1$*, which is computationally unbounded, interacts with* $\mathcal{O}.\mathsf{pre}$*, and outputs an* S*-bit string, and*
- $\mathcal{A}_2$*, which takes an* S*-bit auxiliary input and makes at most* T *queries to* $\mathcal{O}.\mathsf{main}$*.*

Definition 3. *For an* indistinguishability *application G in the $\mathcal{O}$-model, the* advantage of an attacker $\mathcal{A}$ *is defined as*

$$\mathrm{Adv}_{G,\mathcal{O}}(\mathcal{A}) \ := \ 2\left|\mathrm{Succ}_{G,\mathcal{O}}(\mathcal{A}) - \frac{1}{2}\right| .$$

For an unpredictability *application G, the advantage is defined as*

$$\mathrm{Adv}_{G,\mathcal{O}}(\mathcal{A}) \ := \ \mathrm{Succ}_{G,\mathcal{O}}(\mathcal{A}) .$$

An application G is said to be $((S,T,p),\varepsilon)$-secure *in the $\mathcal{O}$-model if for every (S,T,p)-attacker $\mathcal{A}$,*

$$\mathrm{Adv}_{G,\mathcal{O}}(\mathcal{A}) \ \leq \ \varepsilon .$$

Relationship: BF-RO and AI-RO

Theorem 5. *For any $P \in \mathbb{N}$ and every $\gamma > 0$, if an application G is $((S,T,p),\varepsilon')$-secure in the* BF-RO(P)*-model, then it is $((S,T,p),\varepsilon)$-secure in the* AI-RO*-model, for*

$$\varepsilon \ \leq \ \varepsilon' + \frac{2(S + \log \gamma^{-1}) \cdot T_G^{\mathsf{comb}}}{P} + 2\gamma \ ,$$

where T_G^{comb} is the combined query complexity corresponding to G.

Example: Set $\gamma = 2^{-2\lambda}$ and the advantage is roughly $\varepsilon' + \frac{2(S+2\lambda)T}{P}$

Balancing: ε' usually increases with P i.e., as BF-attacker gets to fix more and more points.

Relationship: BF-RO and AI-RO

Theorem 6. *For any* $P \in \mathbb{N}$ *and every* $\gamma > 0$, *if an* unpredictability *application* G *is* $((S, T, p), \varepsilon')$*-secure in the* BF-RO$(P, N, M)$*-model for*

$$P \;\geq\; (S + 2\log\gamma^{-1}) \cdot T_G^{\mathsf{comb}} \;,$$

then it is $((S, T, p), \varepsilon)$*-secure in the* AI-RO(N, M)*-model for*

$$\varepsilon \;\leq\; 2\varepsilon' + 2\gamma \;,$$

where T_G^{comb} *is the combined query complexity corresponding to* G.

Application: Function-Inversion

- **Challenger:** Picks x in $\{0,1\}^{\lambda}$ and sends y=H(x) to online attacker where y in $\{0,1\}^{\lambda}$
- **Goal:** Find x' such that H(x')=y (online attacker may use hints)

- Bit-Fixing Attacker: $A = (A_1, A_2)$
  - Let $L = \{(x_1', y_1'), \dots, (x_P', y_P')\}$ denote set of points fixed by A_1
  - Let E' be the event that $x = x_i'$ or $x = y_i'$ for some $i \leq P$
  - $\Pr[\mathrm{E}'] \leq \Pr[\exists i. x = x_i'] + \Pr[\exists i. y = y_i' \mid \forall i. x \neq x_i'] \leq \frac{P}{2^\lambda} + \frac{P}{2^\lambda}$

x is random

y_i' = H(x_i') is uniformly random if not previously fixed i.e., $x \neq x_i'$ for all i

Application: Function-Inversion

- **Challenger:** Picks x in $\{0,1\}^\lambda$ and sends y=H(x) to online attacker where y in $\{0,1\}^\lambda$
- **Goal:** Find x' such that H(x')=y (may use hints)

- Bit-Fixing Attacker: $A = (A_1, A_2)$
    - Let $L = \{(x_1{}', y_1{}'), \ldots, (x_P{}', y_P{}')\}$ denote set of points fixed by A_1
    - Let E' be the event that $x = x_i{}'$ or $x = y_i{}'$ for some $i \leq P$
    - Let $Q = \{(x_1, y_1), \ldots, (x_T, y_T)\}$ denote queries made by A_2 with corresponding answers
    - Let E_i be the event that $x = x_i$ or $y = y_i$
    - $\Pr\left[\mathrm{E}_i \mid \overline{\mathrm{E}'} \cap \overline{\mathrm{E}_1} \cap \ldots \cap \overline{\mathrm{E}_{i-1}}\right] \leq \frac{1}{2^\lambda - P - (i-1)} + \frac{1}{2^\lambda}$

$\Pr[y = y_i | \ldots]$ note that y_i = H(x_i) is uniformly random if not previously fixed i.e., if $x_i \neq x$, x_i not in L and $x_i \neq x_j$ for all j<i,

$\Pr[x = x_i | \ldots]$ note that x is random, and there $2^\lambda - P - (i-1)$ remaining possible values

Application: Function-Inversion

- **Challenger:** Picks x in $\{0,1\}^\lambda$ and sends y=H(x) to online attacker where y in $\{0,1\}^\lambda$
- **Goal:** Find x' such that H(x')=y (may use hints)

- Bit-Fixing Attacker: $A = (A_1, A_2)$
  - Attacker wins with probability at most

$$\Pr[\mathrm{E}'] + \sum_{i \le q} \Pr\left[\mathrm{E}_i \mid \overline{\mathrm{E}'} \cap \overline{\mathrm{E}_1} \cap \ldots \cap \overline{\mathrm{E}_{i-1}}\right] \le \frac{2P+q}{2^\lambda} + \frac{q}{2^\lambda - P - q} \le \frac{2P+3q}{2^\lambda}$$

Assume $P + q \le 2^{\lambda-1}$

Application: Function-Inversion

- Bit-Fixing Attacker: $A = (A_1, A_2)$
  - Attacker wins with probability at most

$$\Pr[\mathrm{E}'] + \sum_{i \le q} \Pr[\mathrm{E}_i' | \ \overline{\mathrm{E}'} \cap \ \overline{\mathrm{E}_1} \cap \ldots \cap \overline{\mathrm{E}_{i-1}}] \le \frac{2P+q}{2^\lambda} + \frac{q}{2^\lambda - P - q}$$

Set $P \geq 6(S + 2\lambda)T$ ➔ Auxilliary-Input Attacker wins with Probability at most

$$2\left(\frac{6(S+2\lambda)T + T}{2^\lambda} + \frac{6(S+2\lambda)T}{2^\lambda - 6(S+2\lambda)T - T}\right) + 2^{-2\lambda} = O\left(\frac{ST + \lambda T}{2^\lambda}\right)$$

Review: Bit-Fixing vs Auxiliary Input

- Auxiliary-Input: (S,T,p)-attacker
  - A_1 outputs a S-bit hint based entire description of ideal-object
  - A_2 makes at most T oracle queries
  - A_2 may be constrained in other ways (space/time/signing queries etc…) as specified by parameters p.

- Bit-Fixing: (P,S,T,p)-attacker
  - A_1 fixes at most P input/output pairs and outputs a S-bit hint. The remaining ideal object is picked randomly subject to this restriction.
  - A_2 makes at most T oracle queries
  - A_2 may be constrained in other ways (space/time/signing queries etc…) as specified by parameters p.

Bit-Fixing Model (Unruh)

- **Pro:** Much easier to prove lower bounds in Bit-Fixing Model
- **Con: Bit-Fixing model is n**ot a compelling model for pre-processing attacks

- **Usage:** Lower bound in bit-fixing model ➔ Lower bound in Auxilliary-Input Model

- This approach yields tight lower-bounds in the Auxilliary-Input Model for some applications ☺
- Other applications require a different approach (e.g., compression)

Relationship Bit-Fixing and Auxilliary Input

Theorem 1. *Let* $P, K, N, M \in \mathbb{N}$, $N \geq 16$, *and* $\gamma > 0$. *Moreover, let*

$$(\mathsf{AI}, \mathsf{BF}) \in \{(\mathsf{AI\text{-}IC}(K,N), \mathsf{BF\text{-}IC}(P,K,N)), (\mathsf{AI\text{-}GG}(N,M), \mathsf{BF\text{-}GG}(P,N,M))\} \ .$$

Then,

1. *if an application* G *is* $((S,T,p),\varepsilon')$*-secure in the* BF*-model, it is* $((S,T,p),\varepsilon)$*-secure in the* AI*-model, where*
$$\varepsilon \ \leq \ \varepsilon' + \frac{6(S + \log \gamma^{-1}) \cdot T_G^{\mathsf{comb}}}{P} + \gamma\,;$$
2. *if an* unpredictability *application* G *is* $((S,T,p),\varepsilon')$*-secure in the* BF*-model for*
$$P \ \geq \ 6(S + \log \gamma^{-1}) \cdot T_G^{\mathsf{comb}},$$
it is $((S,T,p),\varepsilon)$*-secure in the* AI*-model for*
$$\varepsilon \ \leq \ 2\varepsilon' + \gamma\,,$$

where T_G^{comb} *is the combined query complexity corresponding to* G.

Generic Group Lower Bound

- **Discrete Log Problem:** Given generator g and g^x find x
- **Generic Group Version:** Given $\tau(1)$ and $\tau(x)$ find x

- **Generic Group Version Lower Bound: Any generic attacker making q queries to GGM oracles succeeds with probability at most**

$$O\left(\frac{q^2}{2^{2\lambda}}\right)$$

- **Proof Sketch:** Initialize two sets $K = \{(1, \tau(1))\}$ and $U = \{(x, \tau(x))\}$
  - K ➔ Discrete Log Known
  - U ➔ Discrete Log Depends on Unknown x
  - $\mathrm{mult}(\tau(1), \tau(1)) = \tau(2)$ ➔ Add $(2, \tau(2))$ to K
  - $\mathrm{mult}(\tau(x), \tau(x)) = \tau(2x)$➔ Add $(2x, \tau(x+1))$ to U
  - $\mathrm{mult}(\tau(x), \tau(1)) = \tau(x+1)$➔ Add $(x+1, \tau(x+1))$ to U
  - Each new query adds item to K or U
  - Cannot learn x unless sets intersect e.g., $\mathrm{mult}(\tau(x), \tau(x+1)) = \tau(2x+1)$ is found in K
  - Sets remain disjoint with probability $\approx \frac{|U||K|}{2^{2\lambda}} \leq \frac{q^2}{2^{2\lambda}}$
  - **Technical Note:** If attacker queries $\mathrm{mult}(\varkappa, .)$ for fresh $\varkappa$ which is not in K or U then can add $(\tau^{-1}(\varkappa), \varkappa)$ to K

Generic Group Lower Bound with Bit-Fixing

- **Generic Group Version Lower Bound: Any Bit-Fixing attacker making q queries to GGM oracles (online) and fixing at most P points succeeds with probability at most**

$$O\left(\frac{q^2 + qP}{2^{2\lambda}}\right)$$

Proof Sketch: Let $\boldsymbol{L} = \{(\boldsymbol{x_1}, \boldsymbol{y_1}), \dots, (\boldsymbol{x_P}, \boldsymbol{y_P})\}$ denote the fixed points where attacker fixed $\tau(x_i) = \boldsymbol{y_i}$

Initialize two sets $K = \{(1, \tau(1))\} \cup L$ and $U = \{(x, \tau(x))\}$

K ➔ Discrete Log Known

U ➔ Discrete Log Depends on Unknown x

Each new query adds item to K or U

Cannot learn x unless sets intersect

Sets remain disjoint with probability $\approx \frac{|U||K|}{2^{2\lambda}} \leq \frac{q^2}{2^{2\lambda}}$

Generic Group Lower Bound with Preprocessing Attacker

- **Generic Group Version Lower Bound: Any auxiliary-input attacker making q queries to GGM oracles (online) and with a S bit hint points succeeds with probability at most**

$$O\left(\frac{q^2 + q^2(S+\lambda)}{2^{2\lambda}}\right)$$

Proof Sketch: Set $P = O\big((S+2\lambda)q\big)$ for our bit-fixing attacker A bit-fixing attacker succeeds with probability at most

$$\varepsilon = O\left(\frac{q^2 + q^2(S+\lambda)}{2^{2\lambda}}\right)$$

It follows that the AI-attacker succeeds with probability at most $2\varepsilon + 2^{-2\lambda} = O\left(\frac{q^2+q^2(S+\lambda)}{2^{2\lambda}}\right)$

Block-Ciphers with Leakage

- **Ideal Cipher Model**
  - All parties have oracle access to $E(\cdot,\cdot)$ and $E^{-1}(\cdot\,,\ \cdot)$
  - For any fixed key K the function $E_K(x) \coloneqq\ E(K, x)$ is a truly random permutation and $E_K^{-1}(x) \coloneqq E^{-1}(K, x)$ is the inverse

- **Question:** Can we still safely use the block-cipher after S-bit leakage?

Block-Ciphers with Leakage

- **Question:** Can we still safely use the block-cipher after S-bit leakage?

- Leakage Security Game:
  - Offline Attacker A_1outputs S-bit hint
  - Online Attacker has to predict secret bit b
  - **Real World (b=0):** Online attacker may query $E(\cdot,\cdot)$, $E^{-1}(\cdot\ ,\ \cdot)$, $E(K,\cdot)$ and $E^{-1}(K,\ \cdot)$, where K is a random key picked by the challenger
  - **Ideal World (b=1):** Online attacker may query $E(\cdot,\cdot)$, $E^{-1}(\cdot\ ,\ \cdot)$, $f(\cdot)$ and $f^{-1}(\cdot)$ where f is a truly random permutation (independent of block-cipher + hint).

  - Online Attacker may make T queries to $E(\cdot,\cdot)$ or $E^{-1}(\cdot\ ,\ \cdot)$ and q queries to $E(K,\cdot)$ or $E^{-1}(K,\ \cdot)$ when b=0 (resp. $f(\cdot)$ or $f^{-1}(\cdot)$ when b=1)

Block-Ciphers with Leakage

- Leakage Security Game:
  - Offline Attacker A_1outputs S-bit hint
  - Online Attacker has to predict secret bit b
  - **Real World (b=0):** Online attacker may query $E(\cdot,\cdot)$, $E^{-1}(\cdot\ ,\ \cdot)$, $E(K,\cdot)$ and $E^{-1}(K,\ \cdot)$, where K is a random key picked by the challenger
  - **Ideal World (b=1):** Online attacker may query $E(\cdot,\cdot)$, $E^{-1}(\cdot\ ,\ \cdot)$, $f(\cdot)$ and $f^{-1}(\cdot)$ where f is a truly random permutation (independent of block-cipher + hint).

- **Analysis (Bit Fixing Attacker):** Let

$$L = \{K' \colon \exists x\ s.t. E(K,x) \text{ was fixed } by\ A_1\}$$

and observe that $\Pr[K \in L] \leq |L|2^{-\lambda}$.

Block-Ciphers with Leakage

- **Analysis (Bit Fixing Attacker):** Let

$$L = \{K' \colon \exists x \; s.t. E(K,x) \text{ was fixed } by \; A_1\}$$
$$\Pr[K \in L] \le |L| 2^{-\lambda} \le P 2^{-\lambda}$$

Let B_i denote event that $\mathrm{K} = K_i$ where K_i is the key used in the ith query to $E(.,.)$ or $E^{-1}(\cdot\,,\;\cdot)$

$$\Pr\left[B_i \;\middle|\; \overline{L} \cap \overline{B_1} \cap \ldots \cap \overline{B_{i-1}}\right] \le \frac{1}{2^\lambda - |P| - i} \le \frac{2}{2^\lambda}$$

$2^\lambda - |P| - i$ possible keys remain

WLOG assume $|P| + T \le 2^{\lambda-1}$
(otherwise upper bound on success rate of bit-fixing attacker becomes 1 holds trivially)

Block-Ciphers with Leakage

- **Analysis (Bit Fixing Attacker):** Let

$$L = \{K' \colon \exists x \; s.t.\, E(K, x) \text{ was fixed } by \; A_1\}$$

$$\Pr[K \in L] \le |L| 2^{-\lambda} \le P 2^{-\lambda}$$

Let B_i denote event that $\mathrm{K} = K_i$ where K_i is the key used in the ith query to $E(.,.)$ or $E^{-1}(\cdot\,,\,\cdot)$

If the attacker does not query K or fix an input for K then the attacker cannot distinguish between b=0 or b=1 since $E(K,\cdot)$ is a random permutation. Advantage is upper bounded by

$$\Pr[L] + \sum_{i \le T} \Pr\left[\mathrm{E}_i \mid \bar{L} \cap \overline{B_1} \cap \ldots \cap \overline{B_{i-1}}\right] \le \frac{P}{2^{\lambda}} + \frac{2T}{2^{\lambda}} = \frac{P + 2T}{2^{\lambda}}$$

Block-Ciphers with Leakage

- **Analysis (Pre-Processing Attacker):**

$T_{comb} = T + 1$ combined # of queries to ideal object

Advantage of pre-processing is upper bounded by

$$O\left(\frac{P+T}{2^{\lambda}} + \frac{(S+\lambda)(T+q)}{P}\right)$$

Set $P = \sqrt{(S+\lambda)T2^{\lambda}}$ ➔ $O\left(\frac{T}{2^{\lambda}} + \sqrt{\frac{(S+\lambda)(T+q)}{2^{\lambda}}}\right)$

Block-Ciphers with Leakage

- **Thm (Informal):** an ideal cipher is $\big((\boldsymbol{S}, \boldsymbol{T}, \boldsymbol{q}), \boldsymbol{\varepsilon}\big)$-secure against preprocessing attacks in the auxiliary-input model with

$$\boldsymbol{\varepsilon} = O\left(\frac{T}{2^\lambda} + \sqrt{\frac{(S+\lambda)(T+q)}{2^\lambda}}\right)$$

Best Attack: $\varepsilon = \Omega\left(\frac{T}{2^\lambda} + \sqrt{\frac{S}{2^\lambda}}\right)$

Open Question: Better attack or tighter lower-bound?

Note: Lower-bound likely requires different techniques (e.g., compression?)

Sponge-Construction

- ***Input:*** $m = (m_1, \dots, m_\ell)$ ***with*** $m_i \in \{0,1\}^r$
- $s_0 = s_0^{(1)} \| s_0^{(2)}$ where $s_0^{(1)} = 0^r$ and $s_0^{(2)} = 0^c$ and
- For $i = 1, \dots, \ell$; set $s_i = s_i^{(1)} \| s_i^{(2)} = \pi\left(\left(s_{i-1}^{(1)} \oplus m_i\right) \| s_{i-1}^{(2)}\right)$
- **Output:** $s_\ell^{(1)}$
- **Collision-Game:** Attacker A_1 outputs s-bit hint based on ideal permutation π. A_2 tries to find collision for sponge construction.

Sponge-Construction: $\mathrm{Sponge}_{\pi}(.)$

- ***Input:*** $m = (m_1, \dots, m_\ell)$ ***with*** $m_i \in \{0,1\}^r$

- $s_0 = s_0^{(1)} \| s_0^{(2)}$ where $s_0^{(1)} = 0^r$ and $s_0^{(2)} = 0^c$ and
- For $i = 1, \dots, \ell$; set $s_i = s_i^{(1)} \| s_i^{(2)} = \pi\left(\left(s_{i-1}^{(1)} \oplus m_i\right) \| s_{i-1}^{(2)}\right)$
- **Output:** $\mathrm{Sponge}_{\pi}(m_1, \dots, m_\ell) := s_\ell^{(1)}$

- **Pre-processing Attack:** Find m_1 and m_2 such that $\pi\left((m_1)\|0^c\right)$ and $\pi\left((m_2)\|0^c\right)$ match on first r-bits. A_1 outputs hint m_1 and m_2.

Salted Sponge-Construction: $\mathrm{Sponge}_{\pi,\mathrm{IV}}(.)$

- ***Input:*** $m = (m_1, \dots, m_\ell)$ ***with*** $m_i \in \{0,1\}^r$
- $s_0 = s_0^{(1)} \| s_0^{(2)}$ where $s_0^{(1)} = 0^r$ and $s_0^{(2)} = IV \in \{0,1\}^c$ (random salt) For $i = 1, \dots, \ell$; set $s_i = s_i^{(1)} \| s_i^{(2)} = \pi\left(\left(s_{i-1}^{(1)} \oplus m_i\right) \| s_{i-1}^{(2)}\right)$
- **Output:** $s_\ell^{(1)}$

- **Question:** Is the salted sponge-construction secure against pre-processing attacks?
- **Parameters:** Attacker gets S-bit hint, q queries to π or π^{-1} and outputs a collision of length at most ℓ.
- **First Step:** Analyze a bit-fixing attacker who can fix P input/outputs for π

Salted Sponge-Construction

- ***Input:*** $m = (m_1, \ldots, m_\ell)$ ***with*** $m_i \in \{0,1\}^r$
- $s_0 = s_0^{(1)} \| s_0^{(2)}$ where $s_0^{(1)} = 0^r$ and $s_0^{(2)} = IV \in \{0,1\}^c$ (random salt) For $i = 1, \ldots, \ell;$ set $s_i = s_i^{(1)} \| s_i^{(2)} = \pi\left(\left(s_{i-1}^{(1)} \oplus m_i\right) \| s_{i-1}^{(2)}\right)$
- **Output:** $s_\ell^{(1)}$
- **First Step:** Analyze a bit-fixing attacker who can fix P input/outputs for π
- At the cost of 2ℓ additional queries to π we can assume (WLOG) that the attacker who outputs m and m' has queried π at all points needed to evaluate $\mathrm{Sponge}_\pi(m)$ and $\mathrm{Sponge}_\pi(m')$ since m and m' are at most ℓ-blocks long

Analysis Tool 1: Permutation Graph

Permutation Graph: G_π

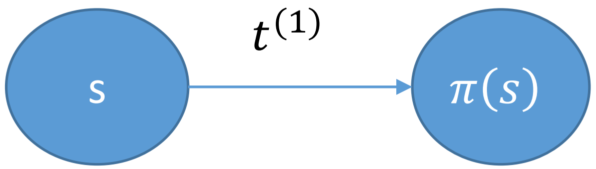
$$s \xrightarrow{t^{(1)}} \pi(s)$$

- **Nodes:** $V = \{0,1\}^{c+r}$
- **Directed Edges:** $(s, t = \pi(s))$
  - Each node has indegree 1 and outdegree 1
  - Label Edges with first r bits of output $t^{(1)} \| t^{(2)} = \pi(s)$
- **Special Start node:** $s_0 = 0^r \| IV$
- A sponge-input $m = (m_1, \ldots, m_\ell)$ ***with*** $m_i \in \{0,1\}^r$ defines a path in the above graph $s_0, s_1, \ldots, s_\ell$ with $s_i = \pi\left(\left(s_{i-1}^{(1)} \oplus m_i\right) \| s_{i-1}^{(2)}\right)$

Analysis Tool 1: Permutation Graph

Permutation Graph: G_π

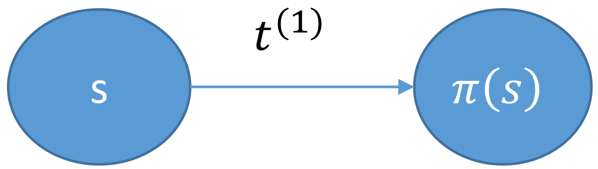
- **Nodes:** $V = \{0,1\}^{c+r}$
- **Directed Edges:** $(s, t = \pi(s))$
  - Each node has indegree 1 and outdegree 1
  - Label Edges with first r bits of output $t^{(1)} \| t^{(2)} = \pi(s)$
- Call a node s *prefixed* if A_1 fixed the value of $\pi(s)$



- Attacker is only aware of some of the edges e.g., after making q queries to π attacker is only aware at most P+q directed edges.
- Let $G_{\pi,0}$ denote initial known graph (using only edges defined by P prefixed points)
- Let $G_{\pi,i}$ denote known graph after i queries to π or π^{-1}

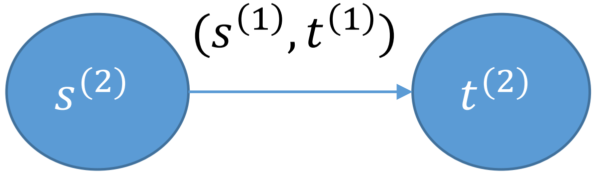
Analysis Tool 1: Permutation Graph

Permutation Graph: G_π



- **Nodes:** $V = \{0,1\}^{c+r}$
- **Directed Edges:** $(s, t = \pi(s))$
  - Each node has indegree 1 and outdegree 1
  - Label Edges with first r bits of output $t^{(1)} \| t^{(2)} = \pi(s)$
- Let $G_{\pi,0}$ denote initial known graph (using only edges defined by P prefixed points)
- Let $G_{\pi,i}$ denote known graph after i queries to π or π^{-1}
- **Special Start node:** $s^* = 0^r \| IV$
- Collision $\Leftrightarrow$ for some label $t^{(1)} \in \{0,1\}^r$ are two distinct paths from start node s^* both ending an edge labeled $t^{(1)}$ in $G_{\pi,q}$

Analysis Tool 2: Super-Node Graph



Permutation Super Graph

- **Nodes:** $V' = \{0,1\}^c$
- **Directed Edges:** $(s^{(2)}, t^{(2)}) \in E'$ iff there exists strings $s^{(1)}, t^{(1)} \in \{0,1\}^r$ such that $t^{(1)} \| t^{(2)} = \pi\left(s^{(1)} \| s^{(2)}\right)$
- Label edges with $(s^{(1)}, t^{(1)})$

- **Starting Super-node:** $IV \in \{0,1\}^c$
- Let G_0 denote the initial super-graph (defined using P fixed points)
- Let G_i denote the super-graph after i queries to π or π^{-1}
- Call a super-node $s^{(2)} \in \{0,1\}^c$ ``*pre-fixed*" if there exists $s^{(1)} \in \{0,1\}^r$ such that $s = s^{(1)} \| s^{(2)}$ was *pre-fixed*

Analysis Tool 2: Super-Node Graph

Permutation Super Graph

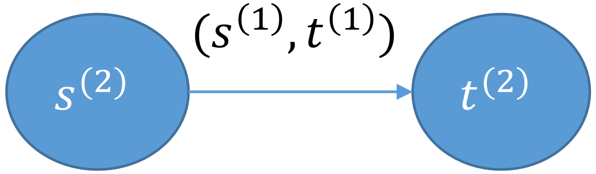
$$s^{(2)} \xrightarrow{(s^{(1)},t^{(1)})} t^{(2)}$$

- **Starting Super-node:** $IV \in \{0,1\}^c$
- Let G_0 denote the initial super-graph (defined using P fixed points)
- Let G_i denote the super-graph after i queries to π or π^{-1}
- Call a super-node $s^{(2)} \in \{0,1\}^c$ ``*pre-fixed*" if there exists $s^{(1)} \in \{0,1\}^r$ such that $s = s^{(1)} \| s^{(2)}$ was *pre-fixed*
- Let B_i denote the event that either
  - (1) G_i contains a path from starting node IV to some pre-fixed
  - (2) G_i contains two distinct paths from starting node IV to x for some supernode x $\in \{0,1\}^c$

$$\Pr[B_0] = \Pr[\text{IV prefixed}] \leq \frac{P}{2^c}$$

Analysis Tool 2: Super-Node Graph

Permutation Super Graph



- Let B_i denote the event that either
  - (1) G_i contains a path from starting node IV to some pre-fixed
  - (2) G_i contains two distinct paths from starting node IV to x for some supernode $\mathrm{x} \in \{0,1\}^c$
- If $B_{T+2\ell}$ does not occur then every supernode has one incoming edge and the value $t^{(1)} \in \{0,1\}^r$ (potential hash output) is uniform.

$$\Pr[COLLISION \mid \overline{B_{T+2\ell}}] \leq \binom{T+2\ell}{2} 2^{-r}$$

Probability of Bad Event (Forward Query)

- Let B_i denote the event that either
  - (1) G_i contains a path from starting node IV to some pre-fixed
  - (2) G_i contains two distinct paths from starting node IV to x for some supernode $\mathrm{x} \in \{0,1\}^c$

Suppose that no bad event has occurred after the first i-1 queries to π or π^{-1} *and that the ith query is of the form*

$$\pi\left(t^{(1)} \middle\| t^{(2)}\right) = y^{(1)} \| y^{(2)}$$

➔Adds edge from supernode $t^{(2)}$ to $y^{(2)}$.

➔Bad if there was already a path to $y^{(2)}$ or if $y^{(2)}$ was fixed.

➔At most $(i+P)2^r$ bad outputs for $\pi\left(t^{(1)} \middle\| t^{(2)}\right)$ out of $2^{r+c} - (i-1+P)$ possibilities

Probability of Bad Event (Forward Query)

- Let B_i denote the event that either
  - (1) G_i contains a path from starting node IV to some pre-fixed
  - (2) G_i contains two distinct paths from starting node IV to x for some supernode $\mathrm{x} \in \{0,1\}^c$

Suppose that no bad event has occurred after the first i-1 queries to π or π^{-1} *and that the ith query is of the form*

$$\pi\left(t^{(1)} \middle\| t^{(2)}\right) = y^{(1)} \middle\| y^{(2)}$$

➔Adds edge from supernode $t^{(2)}$ to $y^{(2)}$. Bad if there was already a path to $y^{(2)}$ or if $y^{(2)}$ was fixed.

$$\Pr\left[B_i \mid \overline{B_1} \cap \ldots \cap \overline{B_{i-1}}\right] \leq \frac{(i+P)2^r}{2^{c+r} - (i-1+P)} \leq \frac{i+P}{2^{c-1}}$$

WLOG assume $P + T + 2\ell \leq 2^{c+r-1}$

Probability of Bad Event (Inverse Query)

- Let B_i denote the event that either
  - (1) G_i contains a path from starting node IV to some pre-fixed
  - (2) G_i contains two distinct paths from starting node IV to x for some supernode $\mathrm{x} \in \{0,1\}^c$

Suppose that no bad event has occurred after the first i-1 queries to π or π^{-1} *and that the ith query is of the form*

$$\pi^{-1}\left(y^{(1)} \middle\| y^{(2)}\right) = t^{(1)} \middle\| t^{(2)}$$

➔Adds edge from supernode $t^{(2)}$ to $y^{(2)}$.

➔Potentially bad if there was already a path from IV to $y^{(2)}$.

➔At most $i2^r$ bad outputs form $\pi^{-1}\left(y^{(1)} \middle\| y^{(2)}\right)$ out of $2^{r+c} - (i - 1 + P)$ possibilities

Probability of Bad Event (Inverse Query)

- Let B_i denote the event that either
  - (1) G_i contains a path from starting node IV to some pre-fixed
  - (2) G_i contains two distinct paths from starting node IV to x for some supernode x ∈ $\{0,1\}^c$

Suppose that no bad event has occurred after the first i-1 queries to π or π^{-1} *and that the ith query is of the form*

$$\pi^{-1}\left(y^{(1)} \middle\| y^{(2)}\right) = t^{(1)} \middle\| t^{(2)}$$

$$\Pr\left[B_i \mid \overline{B_1} \cap \ldots \cap \overline{B_{i-1}}\right] \le \frac{i2^r}{2^{c+r} - (i-1+P)} \le \frac{i}{2^{c-1}} \le \frac{i+P}{2^{c-1}}$$

WLOG assume $P + T + 2\ell \le 2^{c+r-1}$

Probability of Bad Event (Total)

- Let B_i denote the event that either
  - (1) G_i contains a path from starting node IV to some pre-fixed
  - (2) G_i contains two distinct paths from starting node IV to x for some supernode $\mathrm{x} \in \{0,1\}^c$
- ***After all*** $T + 2\ell$ ***queries we have***

$$\Pr\left[B_{T+2\ell}\right] \leq \Pr\left[B_0\ \right] + \sum_{i=1}^{T+2\ell} \Pr\left[B_i\ \middle|\ \overline{B_1} \cap \ldots \cap \overline{B_{i-1}}\right]$$

$$\leq \frac{P}{2^c} + \sum_{i=1}^{T+2\ell} \frac{i+P}{2^{c-1}} \leq \frac{(T+\ell)^2 + TP + 2\ell P + P}{2^{c-1}}$$

Probability of Collision (Bit-Fixing)

$$\Pr[COLLISION \mid \overline{B_{T+2\ell}}] \leq \binom{T+2\ell}{2} 2^{-r}$$

$$\Pr\left[B_{T+2\ell}\right] \leq \frac{(T+\ell)^2 + TP + 2\ell P + P}{2^{c-1}}$$

$$\Pr[COLLISION] \leq \binom{T+2\ell}{2} 2^{-r} + \frac{(T+\ell)^2 + TP + 2\ell P + P}{2^{c-1}}$$

Probability of Collision: Auxilliary-Input

- Bit-Fixing(P): $\Pr[COLLISION] \leq \binom{T+2\ell}{2} 2^{-r} + \frac{(T+\ell)^2 + TP + 2\ell P + P}{2^{c-1}}$

- Set $P = 6(S + c + r)(T + 2\ell)$ to apply main theorem

Thm (Informal): Salted-Sponge is $((S, T, \ell), \varepsilon)$-secure in the auxiliary-input model with

$$\varepsilon \leq O\left(\frac{(T+\ell)^2}{2^{r-1}} + \frac{(T+\ell)^2(S+c+r)}{2^{c-1}}\right)$$

	AI Security	SM Security	Best Attack
OWP	$\frac{ST}{N}$	$\frac{T}{N}$	$\frac{ST}{N}$ [33]
EM	$\left(\frac{ST^2}{N}\right)^{1/2} + \frac{T^2}{N}$	$\frac{T^2}{N}$	$\left(\frac{S}{N}\right)^{1/2}$ [17]
BC-IC	$\left(\frac{ST}{K}\right)^{1/2} + \frac{T}{K}$	$\frac{T}{K}$	$\left(\frac{S}{K}\right)^{1/2}$ [17]
PRF-DM	$\left(\frac{ST}{N}\right)^{1/2} + \frac{T}{N}$	$\frac{T}{N}$	$\left(\frac{S}{N}\right)^{1/2}$ [17]
CRHF-DM	$\frac{(ST)^2}{N}$	$\frac{T^2}{N}$	not known
CRHF-S	$\frac{ST^2}{2^c} + \frac{T^2}{2^r}$	$\frac{T^2}{2^c} + \frac{T^2}{2^r}$	$\frac{ST^2}{N}$ [15]
PRF-S	$\left(\frac{ST^2}{2^c}\right)^{1/2}$	$\frac{T^2}{2^c}$	$\left(\frac{S}{N}\right)^{1/2}$ [17]
MAC-S	$\frac{ST^2}{2^c} + \frac{T}{2^r}$	$\frac{T^2}{2^c} + \frac{T}{2^r}$	$\min\left\{\frac{ST}{N}, \left(\frac{S^2T}{N^2}\right)^{1/3}\right\} + \frac{T}{N}$ [33]
CRHF-MD	$\frac{ST^2}{N}$	$\frac{T^2}{N}$	$\frac{ST^2}{N}$ [15]
PRF-MD-N	$\left(\frac{ST^3}{N}\right)^{1/2} + \frac{T^3}{N}$	$\frac{T^3}{N}$	$\left(\frac{S}{N}\right)^{1/2}$ [17]
NMAC/HMAC	$\frac{ST^3}{N}$	$\frac{T^3}{N}$	$\min\left\{\frac{ST}{N}, \left(\frac{S^2T}{N^2}\right)^{1/3}\right\} + \frac{T}{N}$ [33]

	AI-GGM Security	GGM Security	Best Attack
DL/CDH	$\frac{ST^2}{N} + \frac{T^2}{N}$	$\frac{T^2}{N}$	$\frac{ST^2}{N}$ [16, 38, 5]
t-fold MDL	$\left(\frac{S(T+t)^2}{tN} + \frac{(T+t)^2}{tN}\right)^t$	$\left(\frac{(T+t)^2}{tN}\right)^t$	see caption [16]
DDH	$\left(\frac{ST^2}{N}\right)^{1/2} + \frac{T^2}{N}$	$\frac{T^2}{N}$	$\frac{ST^2}{N}$ [16, 38, 5]
sqDDH	$\left(\frac{ST^2}{N}\right)^{1/2} + \frac{T^2}{N}$	$\frac{T^2}{N}$	$\left(\frac{ST^2}{N}\right)^{1/2}$ [16]
OM-DL	$\left(\frac{S(T+t)^2}{N}\right) + \frac{(T+t)^2}{N}$	$\frac{T^2}{N}$	$\frac{ST^2}{N}$ [16, 38, 5]
KEA	$\frac{ST^2}{N}$	$\frac{T^2}{N}$	not known

Table 2: *Asymptotic upper and lower bounds on the security of applications in the generic-group model against (S,T)-attackers in the AI-ROM; new bounds are in a bold-face font. The value t for the one-more DL problem stands for the number of challenges requested by the attacker. The attack against MDL succeeds with constant probability and requires that $ST^2/t + T^2 = \Theta(tN)$.*

Course Project Ideas: Analyze different construction vs pre-processing attackers (easier) or tighten existing bounds (likely harder).

Reminder: Link Between BF-RO and AI-RO

Theorem 5. *For any $P \in \mathbb{N}$ and every $\gamma > 0$, if an application G is $((S,T,p),\varepsilon')$-secure in the* BF-RO(P)*-model, then it is $((S,T,p),\varepsilon)$-secure in the* AI-RO*-model, for*

$$\varepsilon \;\leq\; \varepsilon' + \frac{2(S + \log \gamma^{-1}) \cdot T_G^{\mathsf{comb}}}{P} + 2\gamma \;,$$

where T_G^{comb} is the combined query complexity corresponding to G.

So far we have used this result (or similar results for Ideal-Ciphers, Permutations etc…) as a black-box.

How is this result proved?

Leaky vs Dense Sources

Definition 1. *An (N, M)-source is a random variable X with range $[M]^N$. A source is called*

- $(1-\delta)$-dense *if for every subset $I \subseteq [N]$,*

$$H_\infty(X_I) \;\geq\; (1-\delta)\cdot|I|\cdot\log M \;=\; (1-\delta)\cdot\log M^{|I|}.$$

- $(P, 1-\delta)$-dense *if it is fixed on at most P coordinates and is $(1-\delta)$-dense on the rest,*

- P-bit-fixing *if it is fixed on at most P coordinates and uniform on the rest.*

- **Idea 1:** Leaky Source (auxiliary-input) can be replaced by convex combination of $(P, 1-\delta)$-dense sources.
- **Idea 2:** Hard to distinguish between $(P, 1-\delta)$-dense source and P-bit-fixing source after T queries

Lemma 1. *Let X be distributed uniformly over $[M]^N$ and $Z := f(X)$, where $f : [M]^N \to \{0,1\}^S$ is an arbitrary function. For any $\gamma > 0$ and $P \in \mathbb{N}$, there exists a family $\{Y_z\}_{z \in \{0,1\}^S}$ of convex combinations Y_z of P-bit-fixing (N, M)-sources such that for any distinguisher D taking an S-bit input and querying at most $T < P$ coordinates of its oracle,*

$$\left|\mathsf{P}\left[\mathcal{D}^X(f(X)) = 1\right] - \mathsf{P}\left[\mathcal{D}^{Y_{f(X)}}(f(X)) = 1\right]\right| \leq \frac{(S + \log 1/\gamma) \cdot T}{P} + \gamma$$

and

$$\mathsf{P}\left[\mathcal{D}^X(f(X)) = 1\right] \leq 2^{(S+2\log 1/\gamma)T/P} \cdot \mathsf{P}\left[\mathcal{D}^{Y_{f(X)}}(f(X)) = 1\right] + 2\gamma\,.$$

Claim 2. *For every $\delta > 0$, X_z is γ-close to a convex combination of finitely many $(P', 1-\delta)$-dense sources for*

$$P' = \frac{S_z + \log 1/\gamma}{\delta \cdot \log M}\,.$$

Claim 3. *For any $(P', 1-\delta)$-dense source X' and its corresponding P'-bit-fixing source Y', it holds that for any (adaptive) distinguisher $\mathcal{D}$ that queries at most T coordinates of its oracle,*

$$\left|\mathsf{P}\left[\mathcal{D}^{X'} = 1\right] - \mathsf{P}\left[\mathcal{D}^{Y'} = 1\right]\right| \;\leq\; T\delta \cdot \log M,$$

and

$$\mathsf{P}\left[\mathcal{D}^{X'} = 1\right] \;\leq\; M^{T\delta} \cdot \mathsf{P}\left[\mathcal{D}^{Y'} = 1\right].$$

Proof. Assume without loss of generality that $\mathcal{D}$ is deterministic and does not query any of the fixed positions. Let $T_{X'}$ and $T_{Y'}$ be the random variables corresponding to the transcripts containing the query/answer pairs resulting from $\mathcal{D}$'s interaction with X' and Y', respectively. For a fixed transcript τ, denote by $\mathsf{p}_{X'}(\tau)$ and $\mathsf{p}_{Y'}(\tau)$ the probabilities that X' and Y', respectively, produce the answers in τ if the queries in τ are asked. Observe that these probabilities depend only on X' resp. Y' and are independent of $\mathcal{D}$.

Observe that for every transcript τ,

$$\mathsf{p}_{X'}(\tau) \;\leq\; M^{-(1-\delta)T} \qquad \text{and} \qquad \mathsf{p}_{Y'}(\tau) \;=\; M^{-T} \tag{1}$$

as X' is $(1-\delta)$-dense and Y' is uniformly distributed.

Since $\mathcal{D}$ is deterministic, $\mathsf{P}[T_{X'} = \tau] \in \{0, \mathsf{p}_{X'}(\tau)\}$, and similarly, $\mathsf{P}[T_{Y'} = \tau] \in \{0, \mathsf{p}_{Y'}(\tau)\}$. Denote by $\mathcal{T}_X$ the set of all transcripts τ for which $\mathsf{P}[T_{X'} = \tau] > 0$. For such τ, $\mathsf{P}[T_{X'} = \tau] = \mathsf{p}_{X'}(\tau)$

Claim 3. *For any $(P', 1-\delta)$-dense source X' and its corresponding P'-bit-fixing source Y', it holds that for any (adaptive) distinguisher $\mathcal{D}$ that queries at most T coordinates of its oracle,*

$$\left|\mathsf{P}\left[\mathcal{D}^{X'}=1\right]-\mathsf{P}\left[\mathcal{D}^{Y'}=1\right]\right| \;\leq\; T\delta\cdot\log M,$$

and

and also $\mathsf{P}[T_{Y'}=\tau]=\mathsf{p}_{Y'}(\tau)$. Towards proving the first part of the lemma, observe that

and a

$$\begin{aligned}
\left|\mathsf{P}\left[\mathcal{D}^{X'}=1\right]-\mathsf{P}\left[\mathcal{D}^{Y'}=1\right]\right| \;&\leq\; \mathsf{SD}(T_{X'},T_{Y'}) \\
&= \sum_{\tau} \max\left\{0, \mathsf{P}[T_{X'}=\tau]-\mathsf{P}[T_{Y'}=\tau]\right\} \\
&= \sum_{\tau\in\mathcal{T}_X} \max\left\{0, \mathsf{p}_{X'}(\tau)-\mathsf{p}_{Y'}(\tau)\right\} \\
&= \sum_{\tau\in\mathcal{T}_X} \mathsf{p}_{X'}(\tau)\cdot\max\left\{0, 1-\frac{\mathsf{p}_{Y'}(\tau)}{\mathsf{p}_{X'}(\tau)}\right\} \\
&\leq\; 1-M^{-T\delta} \;\leq\; T\delta\cdot\log M,
\end{aligned}$$

where the first sum is over all possible transcripts and where the last inequality uses $2^{-x}\geq 1-x$ for $x\geq 0$.

$\tau\in\mathcal{T}_X$

$$\leq\; 1-M^{-T\delta} \;\leq\; T\delta\cdot\log M,$$

where the first sum is over all possible transcripts and where the last inequality uses $2^{-x}\geq 1-x$ for $x\geq 0$.