Lecture 19: Preprocessing Attacks

Henry Corrigan-Gibbs
CS 355 - Spring 2019
June 5, 2019

Logistics

* HW5 due Friday, June 5 @ 5pm via Gradescope
  ↳ NO LATE DAYS ALLOWED!
  Please turn it in on time, else we won't grade it.

* Keep some crypto in your life!
  ↳ Stanford sec seminar
  ↳ Stanford sec lunch
  } Sign up for mailing lists!

* Teaching evaluations.
  ↳ These matter to us!
  We take them very seriously (maybe too seriously).
  Please, please, please fill them out.
  ↳ Best way to thank us .... or to get revenge. ☺
  ↳ If you really liked the course.

Plan

* Recap
* Preprocessing Attacks
* Breaking OWP w/ preprocessing
* Hellman Tables: Breaking OWF w/ preprocessing
* Wrap up & What's next?

Recap: Indistinguishability Obfuscation (iO)

- Ideally, we'd have a strong "virtual black box" (VBB) notion of obfuscation

"Anything you can learn from $\mathcal{O}(C)$ you can learn from queries to C."

$\forall$ Adv $\exists$ Sim s.t. $\forall$ ckts C

$$\{ Adv(\mathcal{O}(C)) \} \approx_c \{ Sim^C(1^{|C|}) \}.$$

Can consider security against advs that see many obfuscated ckts.

VBB Obfuscation is too powerful a notion:

Thm (BGIRSVY '01): Informally, VBB obfuscation cannot exist. Very slick argument.

Intuition: Always more powerful to have code... can feed it to other code as input.

<u>Pf Idea:</u> Let $C_{\alpha,\beta}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ 0 & \text{o.w.} \end{cases}$

Let $D_{\alpha,\beta}(C) = \begin{cases} 1 & \text{if } C(\alpha) = \beta \\ 0 & \text{o.w} \end{cases}$ Outputs 1 on $C_{\alpha,\beta}$ and 0 o.w.

$$\underbrace{\{ Adv(\mathcal{O}(C_{\alpha,\beta}), \mathcal{O}(D_{\alpha,\beta})) \}}_{\text{Always 1}} \overset{?}{\approx} \{ Sim^{C_{\alpha,\beta}, D_{\alpha,\beta}} \} \approx_c \{ Sim^{Zero, D_{\alpha,\beta}} \} \overset{?}{\approx} \underbrace{\{ Adv(\mathcal{O}(Zero), \mathcal{O}(D_{\alpha,\beta})) \}}_{\text{Always Zero}}$$

Close

$\Rightarrow$ VBB cannot exist... many nice! extensions... see paper

Recap So instead of VBB, we settle for iO.

Intuitively: Obfuscations of two programs of equal size computing the same fn are comp indist.

→ Surprisingly powerful, when combined w/ OWFs.

Weird Fact: If P=NP, then iO exists unconditionally.

$$\mathcal{O}(C) = \left\{ \begin{array}{l} \text{Smallest ckt computing same fn} \\ \text{that } C \text{ computes.} \end{array} \right\}$$

If P=NP, $\mathcal{O}(\cdot)$ runs in poly time.

Preprocessing Attacks

AES is perhaps the most widely used crypto primitive...

- TLS
- SSH
- GPG
- ⋮

The security of these applications relies on the following problem being hard.

Chosen-plaintext attack on AES-128

Given: $ct_0 = AES(k, \text{"0000...0"})$

$ct_1 = AES(k, \text{"000.... 1"})$ for $k \xleftarrow{R} \{0,1\}^{128}$

Find: Key k. [Under reasonable assumptions about AES, this key k will be unique.]

What is the best attack?

Brute force: 2^{128} time

Clever attack: $2^{126.1}$ time (2^{88} ct blocks)

{"Biclique" attack
Bogdanov
Biryukov
Rechberger (Asiacrypt 2011)

These attacks assume that the adversary knows nothing about AES when the attack begins.

Q: What if the adversary can precompute a data structure ahead of time that it can later use to mount a key-recovery attack on AES?

A: Can get a much better attack!

Preprocessing: $\approx 2^{86}$ time w/ structure of size 2^{86}

Not practical for two reasons
1) 2^{86} is a lot of space
2) 2^{128} preprocessing time.

Still, very impressive speedup over $2^{126.1}$-time attack!

⇨ Preprocessing attacks are profitable when everyone uses the same few crypto primitives (AES, SHA, etc.)

→ Attacker can amortize cost of building data structure over many subsequent attacks.

We will show a preprocessing attack for <u>inverting functions</u>.

$$f : [N] \to [N]$$

Think of $N \approx 2^{128}$

Notation:

$[N] = \{1, \ldots, N\}$

<u>Examples</u>

$$f_{AES}(x) := AES(x, \text{"}000\cdots0\text{"}) \,\|\, AES(x, \text{"}000\cdots1\text{"})$$

↳ Inverting ≡ Chosen-plaintext attack on AES

$$f_{SHA}(x) := SHA256(x)$$

↳ Inverting ≡ breaking one-wayness of SHA.

↳ Used in "password cracking"

$$f_{PRG}(x) := PRG(x)$$

↳ Inverting ≡ Recovering PRG given PRG output

⇒ distinguishing from random.

If you can invert fns, you can break many crypto primitives!

Preprocessing Attack

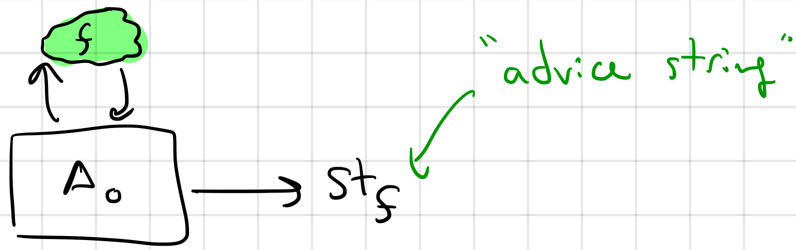
Function $f: [N] \to [N]$.

Attack alg is a pair (A_0, A_1)

↳ We'll focus on algs that use f as a "black box".

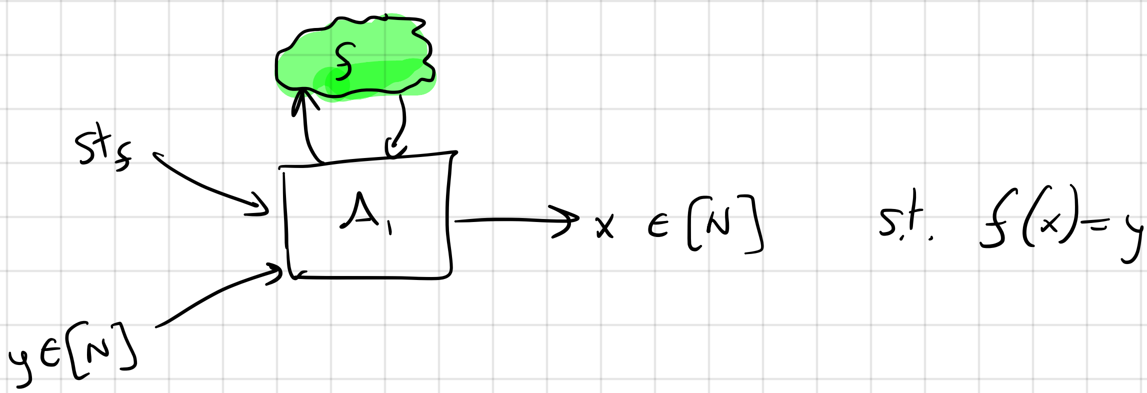
① Preprocessing phase

↳ Adv can look at entire fn f, compute as much as it wants, then outputs an S-bit string st_f.



② Online phase

↳ Adv takes as input its preprocessed advice st_f and a challenge y. Adv makes at most T queries to f and then must output an inverse of y under f.



Intuition: - A_0 does ~~preprocessing relative to f_{AES}~~.
- A_1 breaks your TLS session in real time by inverting f_{AES}.

We measure the complexity of a preproc alg by

$S = |st_f|$ ("space")

T = # of online queries ("time")

Two Simple Preproc Algs for fn inversion

1. Brute-force search. ($S=0$, $T=N$)

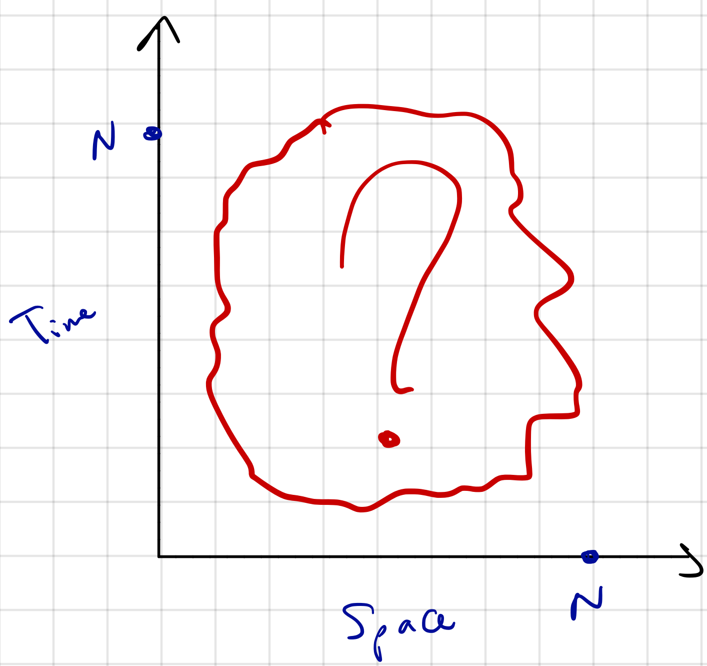
   ↳ A_0 outputs nothing.
   On input y, A_1 computes $f(1), f(2), f(3), \ldots$ until finding an x s.t. $f(x)=y$.

2. Look-up table ($S = N\log N$, $T = \tilde{O}(1)$)

   - A_0 stores table mapping $\langle y \to \text{inverse of } y \text{ under } f \rangle$
   - A_1 looks up inverse in table.

Q: For a given choice of S, what is the best T achievable?

e.g. $S=T=O(N^{1/4})$ possible?



Some history

- In 1975, the US govt (Nat'l Bureau of Standards) published the DES block cipher.
  - ↳ First standard public cipher in U.S.
  - ↳ Used a 56-bit key
- Diffie (Stanford PhD student) and Hellman (Stanford prof) complained that 56 bits were too few (1975–77)
  - ↳ advocated 128-bit keys for "future-proof" security
- Today, can crack 56-bit DES key for \$30 (https://crack.sh)
  - ↳ Today, we use 128-bit keys
- In 1980, Hellman showed that 56-bit keys were dangerous even back then
  - ↳ Introduced preprocessing attack and showed that w/ preproc could break DES in time & space $\underbrace{\approx 2^{40}}_{\text{Practical even then}}$.

More generally...

Thm (Hellman) There exists a preproc alg (A_0, A_1) that inverts a constant fraction of fns $f: [N] \to [N]$ using $S = \tilde{O}(N^{2/3})$ and $T = \tilde{O}(N^{2/3})$... under mild heuristic assumption.

Fiat and Naor (1991) Remove the need for the assumption.

In fact, attack works for any choice of S, T s.t.

$$S^2 T = N^2 \ldots \text{ up to log factors.}$$

Hellman Tables

↳ preproc attack that proves the thm... used for password cracking!

(You may have heard of "Rainbow Tables"... use a very similar idea and achieve essentially the same tradeoff.)

Idea: In preproc phase build a table

$$x_1 \to f(x_1) \to f(f(x_1)) \to \cdots \to f^{N^{1/3}}(x_1) = z_1$$

$$x_2 \to f(x_2) \to f(f(x_2)) \to \cdots \to f^{N^{1/3}}(x_2) = z_2$$

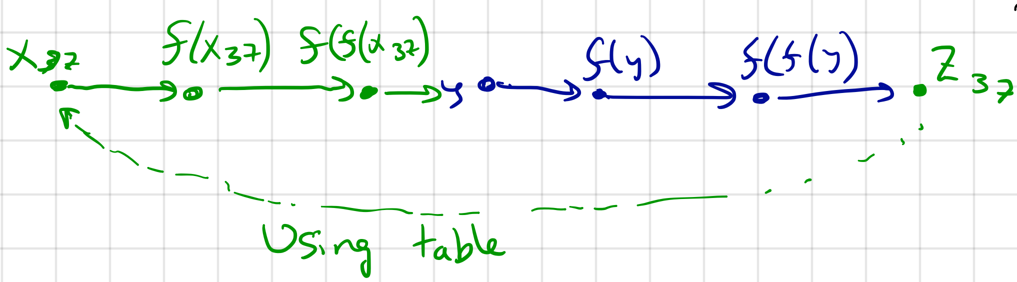
$$\vdots$$

$$x_{N^{1/3}} \to f(x_{N^{1/3}}) \to f(f(x_{N^{1/3}})) \to \cdots \to f^{N^{1/3}}(x_{N^{1/3}}) = z_{N^{1/3}}$$

Store beginning & end of each chain using

$$S = 2 \cdot \log(N) \cdot N^{1/3} \text{ bits.}$$

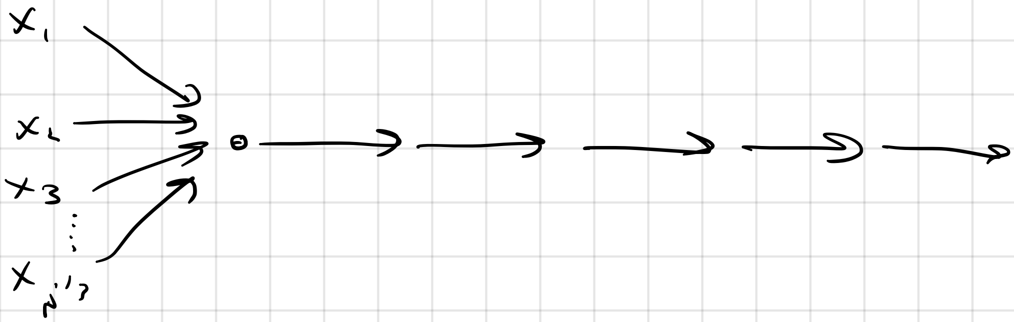
Now, in online phase, say that we are given a point $y \in [N]$ to invert that appears somewhere in our table. What can we do



Boom! We find the inverse after $N^{1/3}$ steps!

So now if challenge y is in table, we're in business. How likely is that?

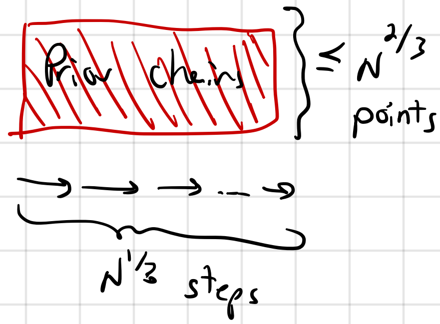
Bad outcome: Table contains only $O(N^{1/3})$ points.



The "at least" cousin of big-O.

Claim: Table contains $\Omega(N^{2/3})$ points in expectation.

Pf. Consider the ith chain.



$$\Pr\left[\begin{array}{l}\text{ith chain collides}\\ \text{with some prior}\\ \text{chain}\end{array}\right] \leq \left(1 - \frac{N^{2/3}}{N}\right)^{N^{1/3}}$$

$$\leq \left(e^{-\frac{N^{2/3}}{N}}\right)^{N^{1/3}}$$

$$\leq e^{-1}$$

$$\leq \text{constant.}$$

Important life fact

$(1+x) \leq e^x$

Then the # of points in table will be $\geq$

$$(\text{constant})\underbrace{(\#\text{chains})}_{N^{1/3}}\underbrace{(\text{length of chain})}_{N^{1/3}} \geq \Omega(N^{2/3})$$

So with this table trick we can invert

$$\varepsilon = \frac{\Omega(N^{2/3})}{N} = \Omega(N^{-1/3}) \text{ fraction of points.}$$

BUT we want to invert all points.

Idea: Rerandomize fn f. Build $N^{1/3}$ tables, each of which inverts $1/N^{1/3}$ fraction of the points.

↳ Then every point will be inverted by <u>some</u> table.

$$S = (N^{1/3} \text{ tables})(\tilde{O}(N^{1/3}) \text{ bits/table}) = \tilde{O}(N^{2/3}) \text{ bits}$$

$$T = (N^{1/3} \text{ tables})(O(N^{1/3}) \text{ time/table}) = \tilde{O}(N^{2/3}) \text{ time}$$

$\Rightarrow$ This completes the attack.

Last task: Show how to construct the rerandomized tables.

Rerandomizing f.

Say that we have random permutations

$$g_1, \ldots, g_{N^{1/3}} : [N] \to [N].$$

Define "flavors" of f:

$$f_1 = g_1 \circ f, \quad f_2 = g \circ f_2, \quad \ldots, \quad f_{N^{1/3}} = g_{N^{1/3}} \circ f.$$

Preproc Phase

- Build $N^{1/3}$, one to invert each of $f_1, \ldots, f_{N^{1/3}}$

Space is $\tilde{O}(N^{1/3}) \cdot \underbrace{N^{1/3}}_{\#\text{tables}} = \tilde{O}(N^{2/3})$.

Online Phase

Given point y to invert, for each table $i = 1, \ldots, N^{1/3}$

* Compute $\hat{y} = g_i(y)$, try to invert using i-th table.
* If find inverse $\hat{x}$ s.t. $f_i(\hat{x}) = \hat{y} = g_i(y)$

then $g_i(f(\hat{x})) = g_i(y)$

$$f(\hat{x}) = y$$

$\Rightarrow \hat{x}$ is inverse of y under f.

Time is $\tilde{O}(N^{1/3}) \cdot \underbrace{N^{1/3}}_{\#\text{tables}} = \tilde{O}(N^{2/3})$.

Analysis

If we think of f_i's as indep random fns (they're not) then analysis is immediate.

With more nuanced analysis, can show that this attack really works even though f_i are not indep.

↳ With even more work, can remove need for indep random g_i's.

The Catch: Time to build tables is

$$\tilde{\Omega}(N^{2/3}) \cdot N^{1/3} = \tilde{\Omega}(N).$$

This is inherent... think about why.

E.g. AES128, $N = 2^{128}$

Preproc time $\approx 2^{128}$

Space $\approx 2^{128 \cdot \frac{2}{3}} = 2^{86}$ bits

Time $\approx 2^{128 \cdot \frac{2}{3}} = 2^{86}$ time.

So Hellman gives us an $S = T = \tilde{O}(N^{2/3})$ attack.
Can we do better?

Thm (Yao) Any preproc attack that makes "black-box" use of f and inverts w/ constant prob satisfies $S \cdot T \geq \tilde{\Omega}(N)$.

$\Rightarrow$ Best we can hope for (black box) is
$$S = T = \tilde{O}(N^{1/2})$$

Still, there's a big gap b/w Hellman's $N^{2/3}$ attack and the impossibility result of Yao $\geq N^{1/2}$.

[2^{86} attack on AES-128 vs 2^{64} attack]

Open Q: Better-than-Hellman attack?

See our recent paper for ideas...

If you come up w/ such an attack, please let us know. ☺