Classical Cryptanalysis

Logistics: - Final project report due at beginning of class next week [No late days!]
- Submit hard copy in class and an electronic copy (projects will be posted on course website at the end of the course)

[Rajeev Motwani Lecture - Piotr Indyk talk, 4:15 PM
Beckte Conference Center

The course so far:

Part I: Assumptions and primitives (symmetric-key cryptography, public-key cryptography, pairings)

Part II: Cryptographic protocols (zero-knowledge, multiparty computation, secret-sharing)

This week: examining the algebraic assumptions that underlie our current cryptographic constructions (factoring and discrete log)

Next week: looking forward - new assumptions based on lattices [conjectured post-quantum resistance]

Quick review of secret-sharing:

- t-out-of-n secret sharing: - take a value x and choose random polynomial p of degree $t-1$ over finite field $\mathbb{F}$ where $p(0) = x$
  - shares are values $(1, p(1)), \ldots, (n, p(n))$
- Reconstruction is polynomial interpolation [linear function in the shares] → often done in the exponent for threshold cryptography
- Security is information-theoretic (all messages equally likely given fewer than t shares)
- Secret sharing is basis of BGW multi-party computation protocol (information-theoretic security)
- Can be extended to idea of secret-sharing functions [BGI15]
  ↳ Special case is distributed point functions — very useful for private information retrieval (PIR) and enabling private writes to a database

This lecture: return to the beginning and see if our fundamental assumptions actually stand (focus on two classical assumptions: discrete log and factoring)

"20th century assumptions"

Discrete Log Algorithms

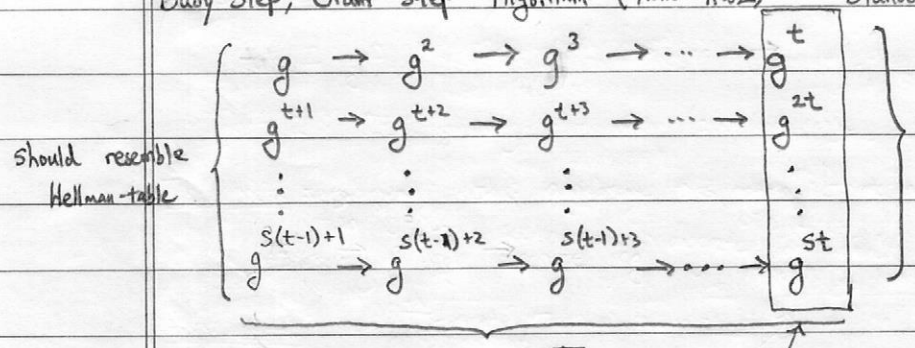
Recall: discrete log problem in prime-order group $\mathbb{G}$:

given generator g and challenge $g^x \in \mathbb{G}$ for random x,

compute x

[shanks] → meet-in-the-middle

Baby-Step, Giant-Step Algorithm (from HW2) – Standard time-space tradeoff (see lecture 2)



↳ note: should directly compute $g^t, g^{2t}, \ldots, g^{st}$ in $O(s)$ time

Store just the last column (with the exponents)

↳ can solve discrete log in $O(t)$ steps by traversing the chain and $O(s)$ space (for the table)

↳ typically: choose $s, t = O(\sqrt{p})$

Main drawback of Baby-Step, Giant-Step: space needed is $O(\sqrt{p})$

- Can we do better? Recall collision-finding algorithms from lecture 2

We want to solve $h = g^x$. Suppose we know $\alpha, \beta, \alpha', \beta'$ such that

$$g^\alpha h^\beta = g^{\alpha'} h^{\beta'} \quad \text{and} \quad (\alpha, \beta) \neq (\alpha', \beta')$$

Then, we can solve for x by first writing

$$\alpha + \beta x = \alpha' + \beta' x$$

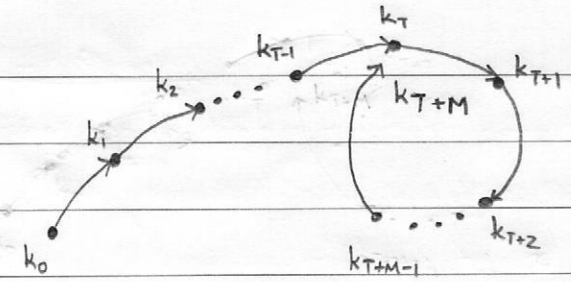
$$\therefore \; x = (\alpha - \alpha')(\beta' - \beta)^{-1} \pmod p$$

Choose a random starting point (α, β) and set $k = g^\alpha h^\beta$. Consider sequence

$$k, f(k), f^2(k), \ldots$$

Suppose that $f(k)$ looks like a random function $\Rightarrow$ after $\sqrt{p}$ operations, there will be a collision: $f^{(i)}(k) = f^{(j)}(k)$

↳ use a p-method to find efficiently



T is length of the tail

M is length of the cycle

(α_i, β_i) $(\alpha_{2i}, \beta_{2i})$

- Use Floyd's cycle-finding algorithm to obtain index i where $k_i = k_{2i}$
- How to choose f? Pollard's suggestion:

$$f(k) = \begin{cases} g \cdot k & \text{if } 0 \le k \le p/3 \\ k^2 & \text{if } p/3 \le k \le 2p/3 \\ h \cdot k & \text{if } 2p/3 \le k < p \end{cases} \qquad \begin{cases} (\alpha, \beta) \mapsto (\alpha+1, \beta) \\ (\alpha, \beta) \mapsto (2\alpha, 2\beta) \\ (\alpha, \beta) \mapsto (\alpha, \beta+1) \end{cases}$$

don't know how to prove that this sequence is random but works well in practice (empirically)

↳ can use any function that induces random-looking behavior

Algorithm runs in time $O(\sqrt{p})$ and space $\tilde{O}(1)$

Observe: basic algorithms above does not exploit any specific properties of the group (just requires ability to evaluate group operation and check equality) - e.g., makes "black-box" use of the group

- Both algorithms require time $O(\sqrt{p})$ - can we do better?
- Shoup 01: discrete log algorithm that makes black-box use of the group must perform at least $\Omega(\sqrt{p})$ group operations - cannot do much better unless we exploit concrete property of the group

Generic group model: captures the notion of a black box group (e.g., only operation that can be done is invoke the group operation and see if two group elements are equal)

- Similar to other ideal models such as random oracle or random permutation model

Generic group: group elements represented by random strings in $\{0,1\}^n$ where $n = \text{poly}(\lambda)$

$g^0 \mapsto \sigma_0 \in \{0,1\}^n$

$g^1 \mapsto \sigma_1 \in \{0,1\}^n$

$\vdots$

$g^x \mapsto \sigma_x \in \{0,1\}^n$

encodings of group elements are independent of group element

$\sigma_1, \ldots, \sigma_x \xleftarrow{R} \{0,1\}^n$ (n large enough so no collisions with overwhelming prob)

group operation handled by an oracle:

$O(\sigma_x, \sigma_y)$:

if there is mapping

$g^x \mapsto \sigma_x$

$g^y \mapsto \sigma_y$

$\Rightarrow$ output σ

Generic Group Model

In generic group model, adversary only gets handles to group elements and oracle access to group operation

- Captures "black-box" access to group

Observe: baby-step, giant-step is a generic algorithm

Given σ_1 (encoding of g) and σ_x (encoding of $h = g^x$):

- adversary can compute σ_t (encoding of g^t) in $O(\log t)$ queries to oracle
- adversary computes table $\sigma_t, \sigma_{2t}, \ldots, \sigma_{st}$ in $O(s)$ queries to oracle } check for collision
- adversary computes $\sigma_x, \sigma_{x+1}, \ldots, \sigma_{x+t}$ in $O(t)$ queries to oracle

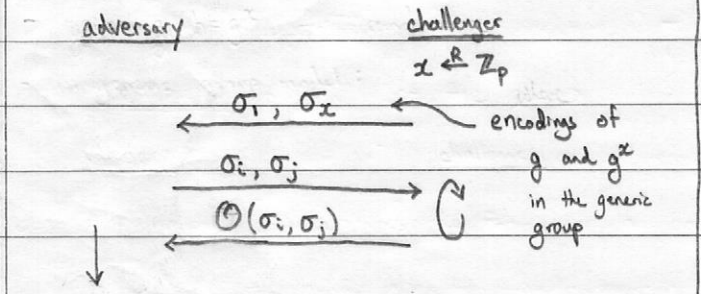
↳ total run-time: $O(\max\{s,t\})$ where $st = p$

$\Rightarrow O(\sqrt{p})$ run-time

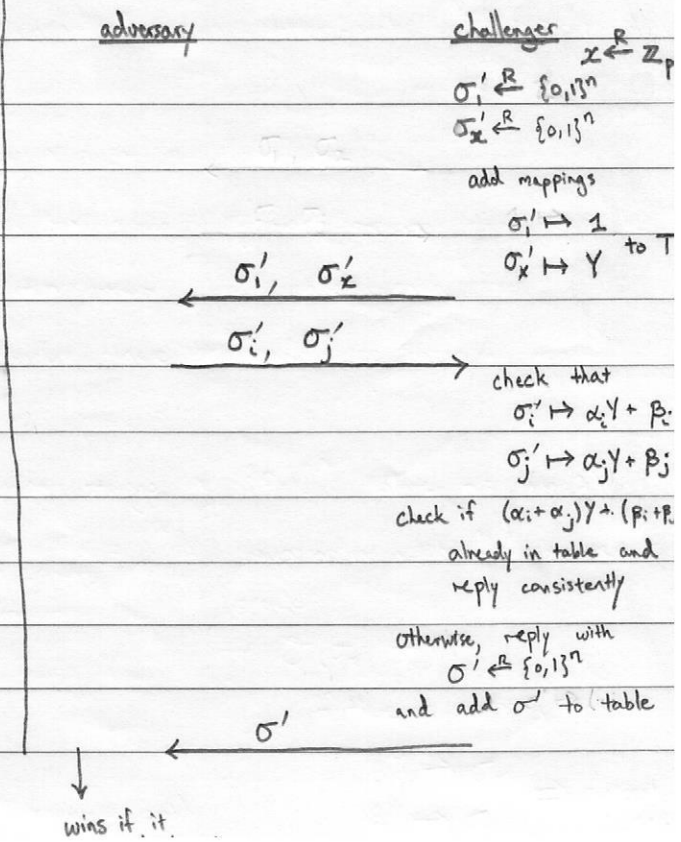
Similarly, Pollard-ρ is generic algorithm

Theorem [Shoup01]: In the generic group model, an adversary that makes m queries to generic group oracle has advantage $O(m^2/p)$ in solving discrete log.

Proof. Consider the real game:



Define the following game:



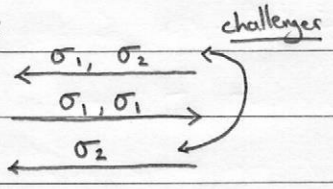
Generic Group Lower Bound

But in new game, adversary cannot win! Its view is <u>independent</u> of x. So in new game, adversary's advantage is $\frac{1}{p}$ (probability that random x is equal to adversary's guess)

What is the difference between the two games?

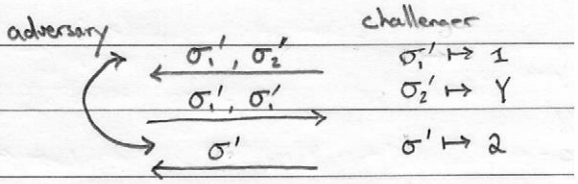
Consider an example: suppose $x = 2$

In real game:



encodings are consistent

In new game:



encodings can be inconsistent (independent of x)

→ different polynomials, equal only after instantiation (at x)

Observation: two games differ only if there are two values σ_i, σ_j that are <u>equal</u> in real game but are <u>unequal</u> in the new game

$$\text{new game} \begin{cases} \sigma_i \mapsto \alpha_i Y + \beta_i \\ \sigma_j \mapsto \alpha_j Y + \beta_j \end{cases} \Rightarrow \text{must be the case that } (\alpha_i - \alpha_j) Y + (\beta_i - \beta_j) \neq 0, \quad \text{(unequal in ideal ga}$$

but $(\alpha_i - \alpha_j) x + (\beta_i - \beta_j) = 0$ (equal when Y instantiated with x)

Now, $(\alpha_i - \alpha_j) Y + (\beta_i - \beta_j)$ is a linear function in Y and is not identically 0:

Schwartz-Zippel lemma

$$\Pr_{x \xleftarrow{\$} \mathbb{Z}_p} \left[(\alpha_i - \alpha_j) x + (\beta_i - \beta_j) = 0 \right] = \frac{1}{p}$$

$\therefore$ with probability $\frac{1}{p}$ over the choice of x (sample x at the end and check if queries simulated correctly) does the challenger produce an inconsistent answer in the new game for any pair of queries

$\therefore$ if adversary makes m queries total, then by union bound, with prob. m^2/p, there are no inconsistent pairs and challenger's behavior in new game coincides with the honest behavior

Adversary's advantage is thus bounded by

$$m^2/p + 1/p = O(m^2/p)$$

m^2/p: difference between two games; $1/p$: advantage in new game

Generic Group Lower Bound

Implications: more efficient algorithms for discrete log must rely on non-generic property (e.g., additional structure)

- For discrete log over integers (e.g., $\mathbb{Z}_p^*$), this is the case
- For discrete log over elliptic curve group, best attack in many cases is generic attack (e.g., 256-bit groups sufficient for 128-bits of security)

Generic group model has emerged as sanity check for other assumptions (e.g., CDH, DDH all hold in generic group model)

- Security in generic group model does not equate to security in standard model, but is often a good first step to understanding security

Generic group model has also been extended to generic bilinear model, generic multilinear model for more advanced cryptographic primitives - very useful tool for exploring viability of new assumptions