Time/Space Tradeoffs for Symmetric Cryptanalysis

DES block cipher - developed by IBM (initially key was 64 bits)

$\Rightarrow$ NSA tries to convince IBM to reduce key size to 48 bits to enable brute force

$\Rightarrow$ Eventually compromised on 56-bit design

Question: What is the cost of breaking DES? Let $N$ denote number of keys.

Exhaustive search: given several message-plaintext pairs, try all of the keys

- time complexity: $O(N)$
- space complexity: $O(1)$

$\Rightarrow$ e.g. CPA attack

Table lookup: suppose we have known message-plaintext pair and precomputed table of all key

- time complexity: $O(\log N)$
- space complexity: $O(N)$

More general problem: inverting a one-way function (permutation)

$\Rightarrow$ Hellman introduced notion of time-memory tradeoff with Hellman tables (Ca255)

\[
\begin{align*}
 & x_1 \rightarrow f(x_1) \rightarrow f(f(x_1)) \rightarrow \cdots \rightarrow f^{(t)}(x_1) \\
 & x_2 \rightarrow f(x_2) \rightarrow f(f(x_2)) \rightarrow \cdots \rightarrow f^{(t)}(x_2) \\
 & \vdots \\
 & x_m \rightarrow f(x_m) \rightarrow f(f(x_m)) \rightarrow \cdots \rightarrow f^{(t)}(x_m)
\end{align*}
\]

$\Rightarrow$ values in Hellman table

Key observation: suppose all elements in table are distinct

$\Rightarrow$ success probability of inverting OWF on random input is $\frac{m^t}{N}$

$\Rightarrow$ compare with exhaustive search: $\frac{t}{N}$ and table lookup: $\frac{m}{N}$

$\Rightarrow$ constant fraction overlaps $\Rightarrow$ success prob. reduced by same constant factor

How much overlap should we expect in the table entries? Suppose $f$ is modeled as a random function. How much of the domain can we expect to cover?

Let $X_{ij}$ denote the $(i,j)^{th}$ entry in the table. Let $A$ be the set of values in the table. Then:

$$|A| = \sum_{i=1}^{m} \sum_{j=1}^{t} I\{X_{ij} \text{ is new}\}$$

$$\Pr[x \in A] = \frac{|A|}{N} \approx \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} e^{-t^2/N}$$
Now, \( E[|A|] = \sum_{i=1}^{m} \sum_{j=1}^{t} P_r(X_{ij} \text{ is new}) \) (linearity of expectation, expectation of indicator)

Then,
\[
\Pr(X_{ij} \text{ is new}) \geq \Pr (X_{i1}, \ldots, X_{ij-1} \text{ are all new})
\]
\[
= \Pr (X_{i1} \text{ is new}) \Pr (X_{i2} \text{ is new} \mid X_{i1} \text{ is new}) \ldots \Pr (X_{ij} \text{ is new} \mid X_{i1}, \ldots, X_{ij-1} \text{ is new})
\]
\[
= \frac{N-|A_i|}{N} \times \frac{N-|A_i|-1}{N} \times \ldots \times \frac{N-|A_i|-j+1}{N} \quad \text{where } A_i \text{ is new elements in first } i \text{ rows}
\]
\[
\geq \left( \frac{N - it}{N} \right)^j
\]
\[
\therefore \Pr (x \in A) \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} \left( \frac{N - it}{N} \right)^j = \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} \left( 1 - \frac{it}{N} \right)^j \approx \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} e^{-it/N}
\]

Not much gained when \( mt^2 \gg N \) (consider very small) \( \Rightarrow \) setting \( m = t = \sqrt{N} \) not sufficient (too many columns)

When \( mt^2 \ll N \Rightarrow \Pr(x \in A) \approx \frac{mt}{N}
\]
\[
\Rightarrow \text{set } m = t = N^{1/3} \Rightarrow \text{ succeed w.p. } \approx N^{-1/3}
\]
\[
\Rightarrow \text{time } N^{2/3}, \text{ space } N^{7/3} - 54 \text{-bit DES } \Rightarrow \text{38-bit effective security ($\$10$ cost in 1978)}
\]

What if we could use \( O(N) \) space during precomputation?

\[
\Rightarrow \text{construct table of size } O(N) \text{ to allow online insertion in time } O(N)
\]

What if \( f \) is a permutation (with a single cycle)? \( \Rightarrow \) discrete log

\[
\Rightarrow \text{again admits a } \sqrt{N}, \sqrt{N} \text{ time-memory trade-off}
\]

Double DES and Meet-in-the-Middle Attacks.

Double DES: \( \text{DES}_2((k_1, k_2), x) : \text{DES}(k_2, \text{DES}(k_1, x)) \) looks like 112-bit keys

- time-memory tradeoff (with known plaintext \( m \))
- \([\text{pre}]-\text{compute table } \text{DES}(k_i, m) \text{ for all keys } k_i (size } 2^{56} \)
- given ciphertext \( c \), evaluate \( \text{DES}^{-1}(k_2, c) \) for all \( k_2 \)

\[
c = \text{DES}(k_2, \text{DES}(k_1, x))
\]
\[
\text{DES}^{-1}(k_2, c) = \text{DES}(k_1, x)
\]
\[
\Rightarrow \text{2}^{57} \text{-cost attack}
\]
What if we have only $w < 2^{56}$ memory

$\Rightarrow$ then partition the space into blocks of size $w$ and repeat attack for each block

$\Rightarrow$ requires $\left( \frac{N}{w} \right) (w + N) = N + \frac{N^2}{w}$ time and $w$ space

Another approach: reduce meet-in-the-middle to a collision search problem (reduce space requirements)

Meet in the middle attack: find $(k_1, k_2)$ such that

$$\text{DES}^{-1}(k_2, c) = \text{DES}(k_1, m)$$

$\Rightarrow f_2(k_2) = f_1(k_1)$

Define a function $f(k, c) = f_c(k)$ observe that $f(k_2, a) = f(k_1, l)$ which is a collision for $f$

$\Rightarrow$ goal: build collision finding algorithm
(also independently important)

Abstract goal: suppose we have a function $f : S \rightarrow S$ and we want to find a collision

- Naive strategy: compute $f(x_1), \ldots, f(x_n)$ for random $x_1, \ldots, x_n$ until collision is found

$\Rightarrow$ birthday bound: time $O(\sqrt{N})$ and space $O(\sqrt{N})$

- Using a cycle finding algorithm: "rho method"

  start with random $x$ and compute $x, f(x), f(f(x)), \ldots, f^n(x)$

  if $f$ looks like a random function, then after $\sqrt{N}$ applications, will have a collision (e.g., cycle)

- Cycle detection via fast pointer/slow pointer [Floyd]

  choose random $x_0 = x_0'$ and compute
  
  \[ x_i \leftarrow f(x_{i-1}) \]
  
  \[ x'_i \leftarrow f(f(x'_{i-1})) \]

  $\Rightarrow$ How to go from distinguished point to collision:

  1. Compute length of cycle $O(\sqrt{N})$ time

  2. Advance further pointer until it is equidistant from the distinguished point

  3. Advance both pointers in sync -- will collide at some position
Rho algorithm: $O(\sqrt{N})$ time, $O(1)$ space, for finding collisions

- Naive parallel extension to the algorithm does not provide compelling speed-up
- Suppose we have $m$ processors each running independent execution of the algorithm
  - After each processor has evaluated $f$ a total of $k$ times, probability there is no collision
    \[
    \left(1 - \frac{1}{N}\right)\left(1 - \frac{1}{N}\right) \cdots \left(1 - \frac{k - 1}{N}\right) \approx e^{-\frac{k^2}{N}} \leq \left(1 - \frac{k}{N}\right)^m \approx e^{-\frac{k^2m}{N}}
    \]
  - So expression is for single processor finding collision over domain of size $N/m$
  - Collision after $k = \sqrt{\frac{N}{m}}$ steps $\implies$ only $\sqrt{m}$ speed-up despite $m$ processors

Parallel collision search: getting linear speed-up from multiple processors [van Oorschot and Wiener]

- Each processor chooses a random point and evaluates $f$ until hitting a "distinguished" point

```
  processor 1

  processor 2

  processor 3
```

- Observation: after $O(\sqrt{N})$ total points, there will be a collision

  $\implies$ after each processor has taken $O(\sqrt{N}/m)$ steps

- Choose distinguished points so trails expected to be long $\implies$ will not require too much space
- Gives collision-finding algorithm with small space!