Rainbow Tables

Store: username, H(passwd)
Still vulnerable to brute force dictionary attack

Ex: 10-char strings of upper/lower case letters

⇒ 52^{10} strings in dictionary
Storing (explicitly) hash of each dictionary item
⇒ ~2^{61} bytes (2 exabytes = 2 billion GB)
looking up a hash?
⇒ at most 61 steps (binary search)

Hash Chain:

```
<table>
<thead>
<tr>
<th>dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>30,13^{128} = output of H</td>
</tr>
</tbody>
</table>
```

R: reduction function
⇒ not an inverse of H
⇒ not a hash function itself

Ex: Dictionary = all 10-digit #s
R: interpret y ∈ 30,13^{128} as integer, reduce mod 10^{10}

Mike is out of town next week ⇒ his office hours canceled
Mon: Jesse Walker of Intel (WEP cracker! )
project outlines due
Wed: Zhangxiang
Fri: no class ⁄ homework due!
Chain: instead of storing \[ \begin{array}{c|c} x_1 & H(x_1) \\ x_2 & H(x_2) \\ \vdots & \vdots \\ \end{array} \]
we do:
\[ x_1 \xrightarrow{H} y_1 \xrightarrow{R} x_2 \xrightarrow{H} y_2 \xrightarrow{R} \ldots \xrightarrow{H} y_n \]

Store only \( x_1 \) & \( y_n \)

Given \( y^* \), how can we check whether it appeared in this hash chain?

\( \rightarrow \) reconstruct chain from \( x_1 \xrightarrow{H} y_1 \xrightarrow{R} \ldots \)

\( \rightarrow \) do \( y^* \xrightarrow{R} x_1 \xrightarrow{H} y_n \)

see if you reach \( y_n \) eventually

Suppose \( y^* \xrightarrow{R} \ldots \xrightarrow{H} \boxed{y_n} \)

Can I find \( x^* \) s.t. \( H(x^*) = y^* \) ?

\( \rightarrow \) start from \( x_1 \xrightarrow{H} \), traverse until getting \( y^* \)

Am I guaranteed to encounter \( y^* \)?

No:
\( y^* \xrightarrow{R} \ldots \xrightarrow{R} x_3 \xrightarrow{R} \ldots \xrightarrow{R} x_n \xrightarrow{R} y_n \)

2 chains merged from collision! \( \Rightarrow \) false positive
Time-Space Tradeoff:

- Hash chain requires $\frac{1}{n}$ as much storage.
- Looking up value in chain costs $n$ times more.

Rainbow Tables:

- Use $n$ different reduction functions $R_1 \ldots R_n$.

$$x_1 \overset{H}{\rightarrow} y_1 \overset{R_1}{\rightarrow} x_2 \overset{H}{\rightarrow} y_2 \overset{R_2}{\rightarrow} \ldots \overset{R_n}{\rightarrow} y_n$$

Checking whether $y^*$ exists in chain?

- $y^* \overset{R_n}{\rightarrow} \overset{H}{\rightarrow} y_n$ ??
- $y^* \overset{R_{n-1}}{\rightarrow} \overset{H}{\rightarrow} R_n \overset{H}{\rightarrow} y_n$ ??
- $\vdots$

Collisions must happen with same $R$ function to merge chains.

- Doesn't merge chains $R_3$
- Does merge! $R_2$

Overview:

- Generate lots of rainbow hash chains, random starting points.
- Can compute how many chains needed to cover 99.9% of dictionary.
Counter measure: **SALT**

instead of storing $H(\text{passwd})$

choose random $s$, store $(s, H(s\|\text{passwd}))$

“Salt”

Idea: for each $s$, $H(s\|\text{passwd})$ is a unique hash function to crack, I need a different rainbow table for each salt I care about