Cracking Passwords with Time-memory Trade-offs

INGI2347: COMPUTER SYSTEM SECURITY (Spring 2014)

Marco Canini

Guest lecturer: Xavier Carpent

SUMMARY

MOTIVATIONS

One-way Function

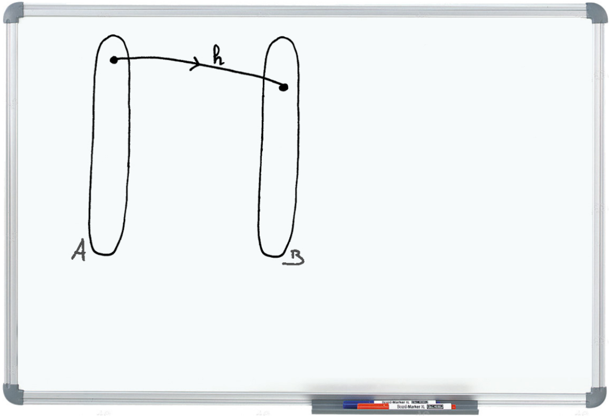
Function $h : A \rightarrow B$ that is easy to compute on every input, but hard to invert given the image of an arbitrary input.

One-way Function

Function $h : A \rightarrow B$ that is easy to compute on every input, but hard to invert given the image of an arbitrary input.

One-way Function

Function $h : A \to B$ that is easy to compute on every input, but hard to invert given the image of an arbitrary input.



Example: Password-based Authentication

User (username, pwd) Computer

$\xrightarrow{\text{username, pwd}}$

Compute $h(\text{pwd})$

username$_1$	$h(\text{pwd}_1)$
username$_2$	$h(\text{pwd}_2)$
username$_3$	$h(\text{pwd}_3)$
$\vdots$	$\vdots$
username$_N$	$h(\text{pwd}_N)$

Exhaustive Search

- Online exhaustive search:
  - Computation: $N := |A|$
  - Storage: 0
  - Precalculation: 0

- Precalculated exhaustive search:
  - Computation: 0
  - Storage: N
  - Precalculation: N

HELLMAN TABLES

Precalculation Phase

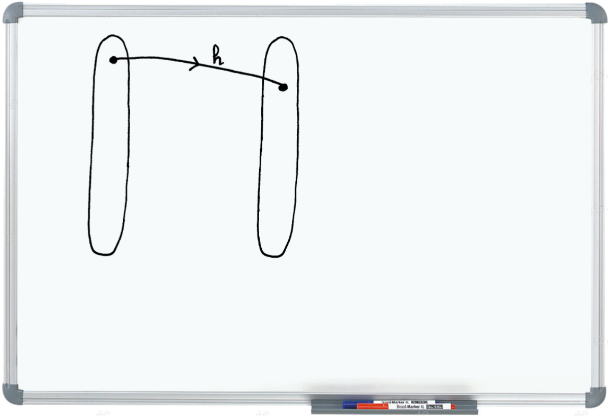
- Martin Hellman's cryptanalytic time-memory trade-off (1980).
- Precalculation phase to speed up the online attack: $T \propto \frac{N^2}{M^2}$

Precalculation Phase

- Martin Hellman's cryptanalytic time-memory trade-off (1980).
- Precalculation phase to speed up the online attack: $T \propto \frac{N^2}{M^2}$

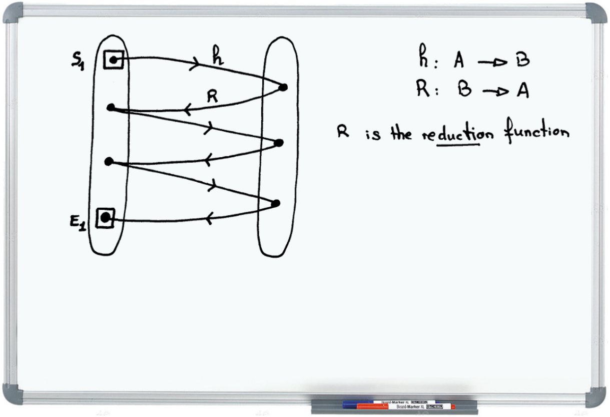
Precalculation Phase

- Martin Hellman's cryptanalytic time-memory trade-off (1980).
- Precalculation phase to speed up the online attack: $T \propto \frac{N^2}{M^2}$



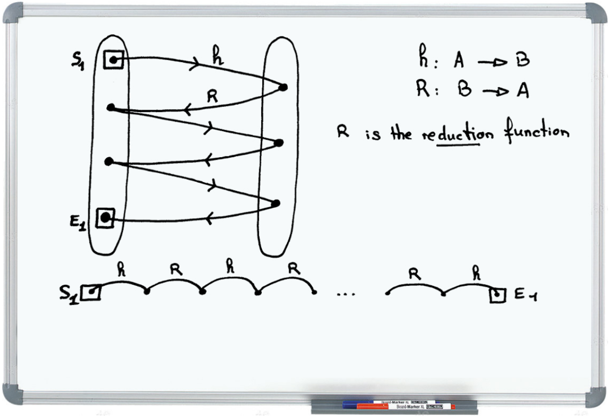
Precalculation Phase

- Martin Hellman's cryptanalytic time-memory trade-off (1980).
- Precalculation phase to speed up the online attack: $T \propto \frac{N^2}{M^2}$



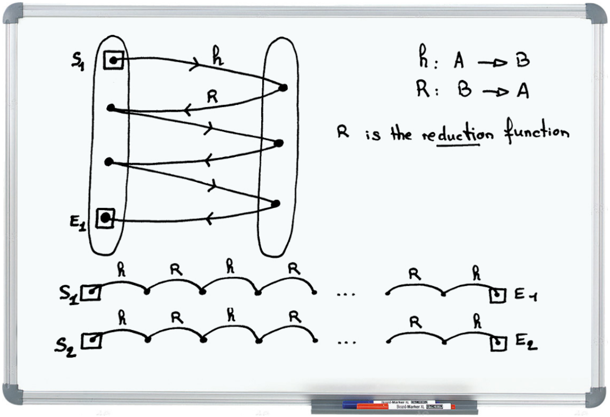
Precalculation Phase

- Martin Hellman's cryptanalytic time-memory trade-off (1980).
- Precalculation phase to speed up the online attack: $T \propto \frac{N^2}{M^2}$



Precalculation Phase

- Martin Hellman's cryptanalytic time-memory trade-off (1980).
- Precalculation phase to speed up the online attack: $T \propto \frac{N^2}{M^2}$



Reduction Functions

- $R : B \to A$ is used to map a point from B to A arbitrarily
- It should be fast to compute (w.r.t. h)
- R should be surjective.
- R should be deterministic.
- $\forall a \in A, \;\; |R^{-1}(a)| \approx \frac{|B|}{|A|}$
- Typically, $R : b \mapsto b \bmod N$.

Precalculation Phase (recap)

- Invert $h : A \to B$.
- Define $R : B \to A$ an arbitrary (reduction) function.
- Define $f : A \to A$ such that $f = R \circ h$.
- Chains are generated from arbitrary values in A.

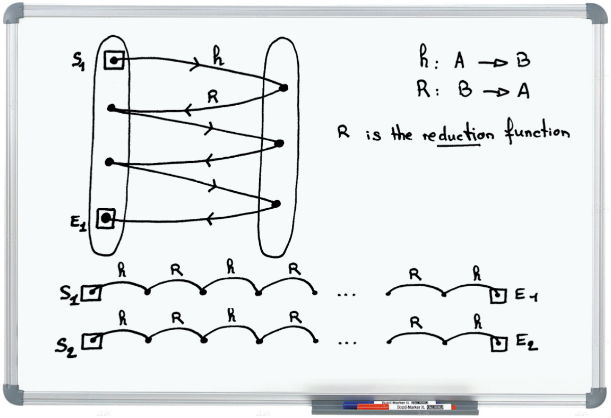
$$\begin{array}{ccccccccccccc}
S_1 & = & X_{1,1} & \xrightarrow{f} & X_{1,2} & \xrightarrow{f} & X_{1,3} & \xrightarrow{f} & \dots & \xrightarrow{f} & X_{1,t} & = & E_1 \\
S_2 & = & X_{2,1} & \xrightarrow{f} & X_{2,2} & \xrightarrow{f} & X_{2,3} & \xrightarrow{f} & \dots & \xrightarrow{f} & X_{2,t} & = & E_2 \\
 & \vdots & & & & & & & & & & \vdots & \\
S_m & = & X_{m,1} & \xrightarrow{f} & X_{m,2} & \xrightarrow{f} & X_{m,3} & \xrightarrow{f} & \dots & \xrightarrow{f} & X_{m,t} & = & E_m
\end{array}$$

- The generated values should cover the set A (probabilistic).
- Only the first and the last element of each chain is stored.

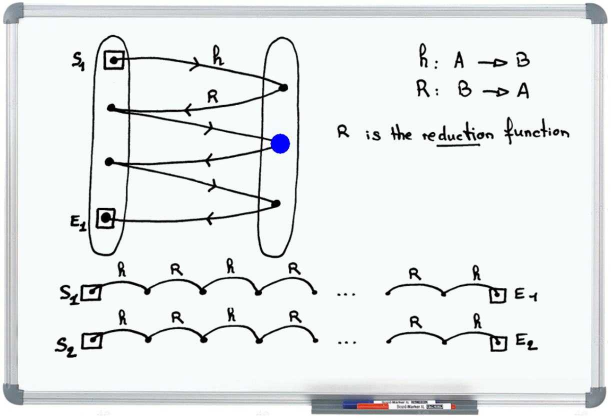
Online Attack

Online Attack

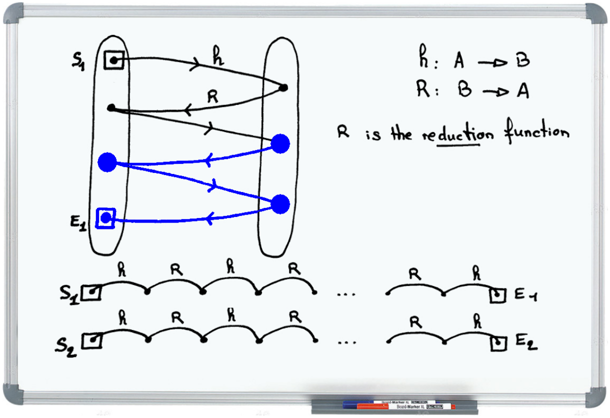
Online Attack



Online Attack

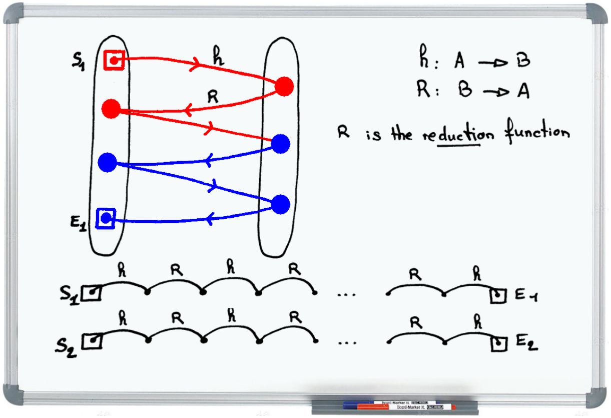


Online Attack



$h: A \to B$
$R: B \to A$
R is the reduction function
S_1
E_1
h
R
S_1 h R h R ... R h E_1
S_2 h R h R ... R h E_2

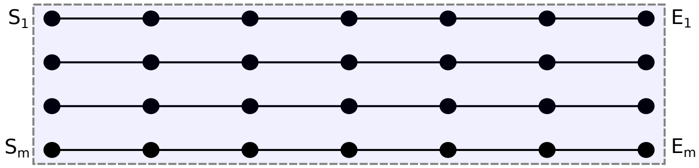
Online Attack



S_1
h
R
E_1
$h: A \to B$
$R: B \to A$
R is the reduction function
S_1 h R h R … R h E_1
S_2 h R h R … R h E_2

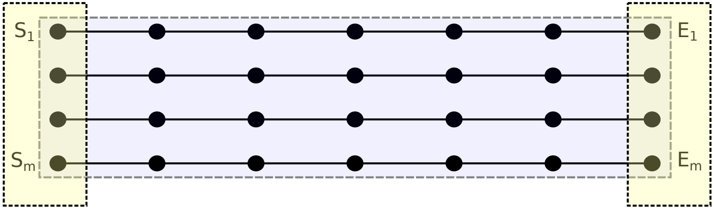
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \ldots y_s$



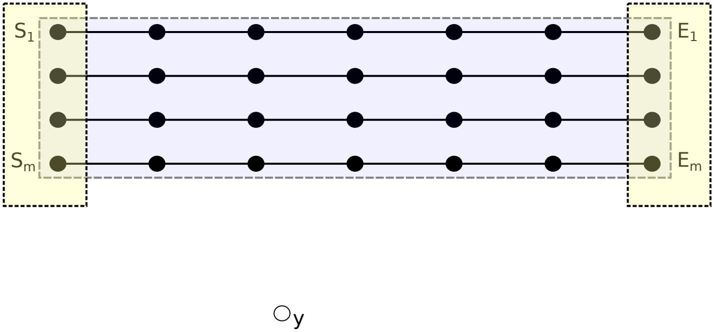
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \dots y_s$



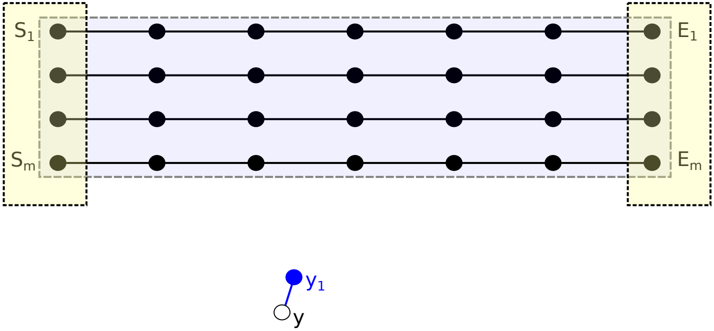
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \ldots y_s$



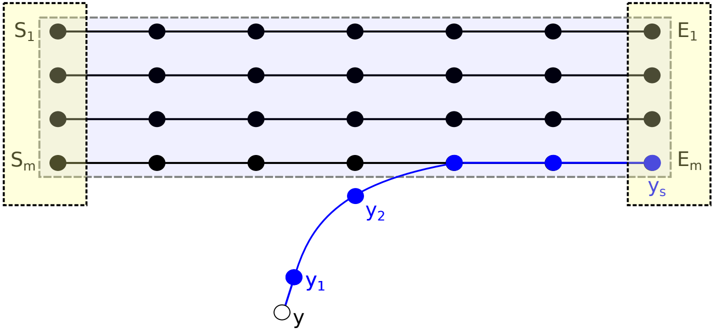
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \dots y_s$



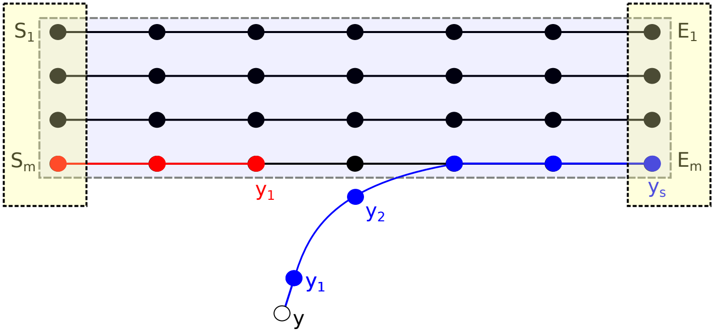
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \ldots y_s$



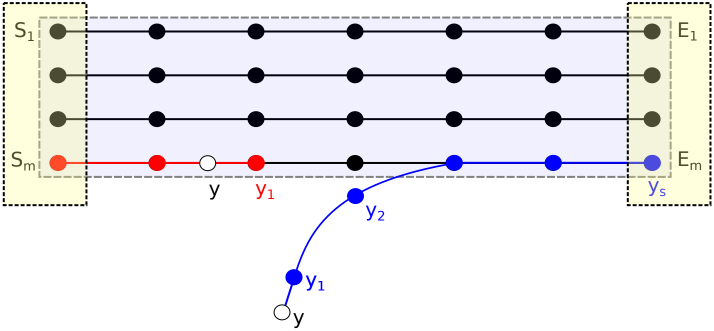
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \ldots y_s$



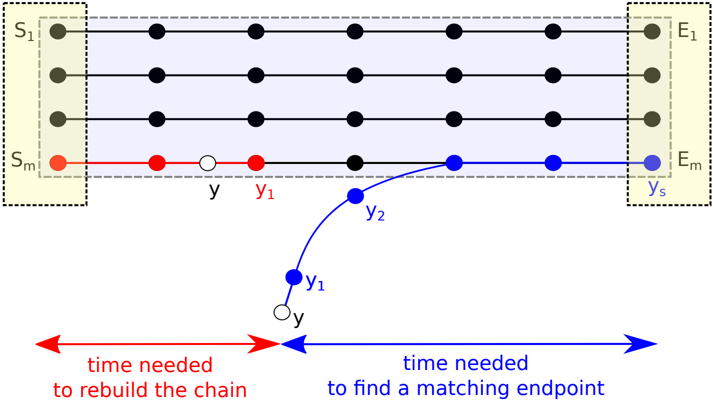
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \dots y_s$



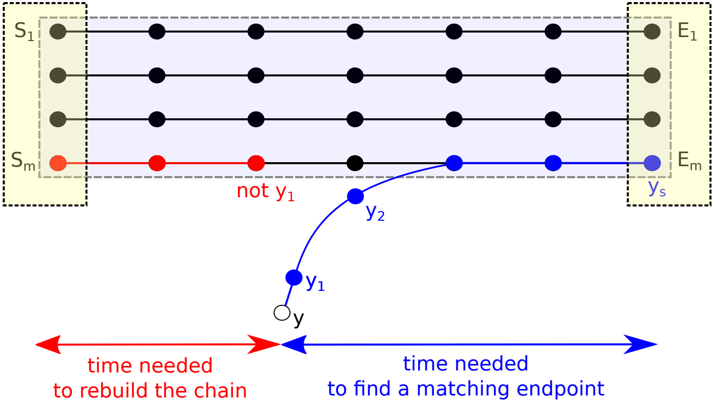
Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \dots y_s$



Online Attack (Recap)

- Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1: $y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \dots y_s$



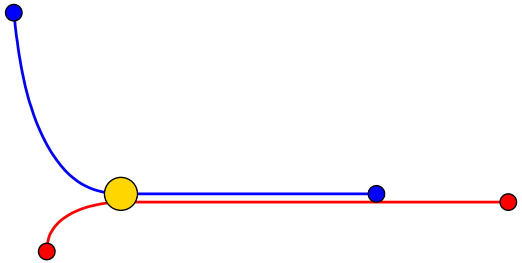
S_1
S_m
E_1
E_m
not y_1
y_s
y_2
y_1
y
time needed to rebuild the chain
time needed to find a matching endpoint

Coverage and Collisions

- Collisions occur during the precalculation phase.

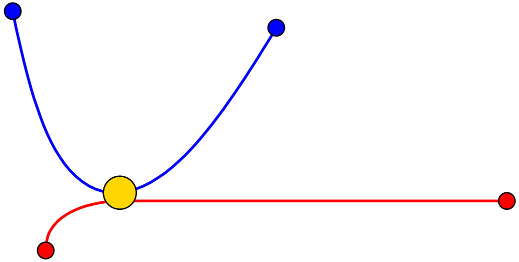
Coverage and Collisions

- Collisions occur during the precalculation phase.



Coverage and Collisions

- Collisions occur during the precalculation phase.
- Several tables with different reduction functions.



OECHSLIN TABLES

Using Several Reduction Functions (Oechslin, 2003)

- Use a different reduction function per column: rainbow tables.
- Invert $h : A \to B$.
- Define $R_i : B \to A$ arbitrary (reduction) functions.
- Define $f_i : A \to A$ such that $f_i = R_i \circ h$.

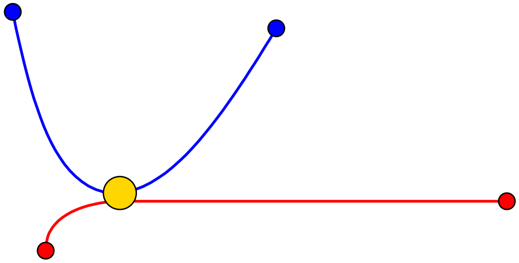
$$\begin{array}{ccccccccccccc}
S_1 & = & X_{1,1} & \xrightarrow{f_1} & X_{1,2} & \xrightarrow{f_2} & X_{1,3} & \xrightarrow{f_3} & \dots & \xrightarrow{f_t} & X_{1,t} & = & E_1 \\
S_2 & = & X_{2,1} & \xrightarrow{f_1} & X_{2,2} & \xrightarrow{f_2} & X_{2,3} & \xrightarrow{f_3} & \dots & \xrightarrow{f_t} & X_{2,t} & = & E_2 \\
 & \vdots & & & & & & & & & & \vdots & \\
S_m & = & X_{m,1} & \xrightarrow{f_1} & X_{m,2} & \xrightarrow{f_2} & X_{m,3} & \xrightarrow{f_3} & \dots & \xrightarrow{f_t} & X_{m,t} & = & E_m
\end{array}$$

Discarding the Merges

- If 2 chains collide in different columns, they don't merge.

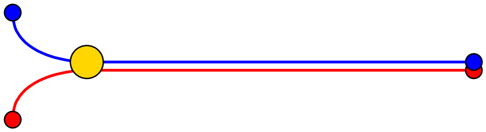
Discarding the Merges

- If 2 chains collide in different columns, they don't merge.



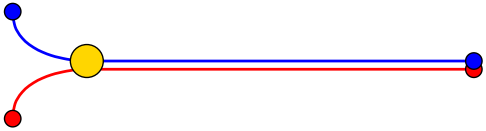
Discarding the Merges

- If 2 chains collide in different columns, they don't merge.



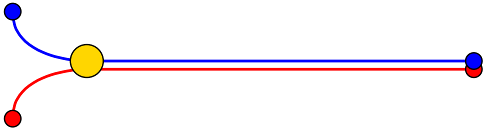
Discarding the Merges

- If 2 chains collide in different columns, they don't merge.
- If 2 chains collide in same column, merge can be detected.



Discarding the Merges

- If 2 chains collide in different columns, they don't merge.
- If 2 chains collide in same column, merge can be detected.

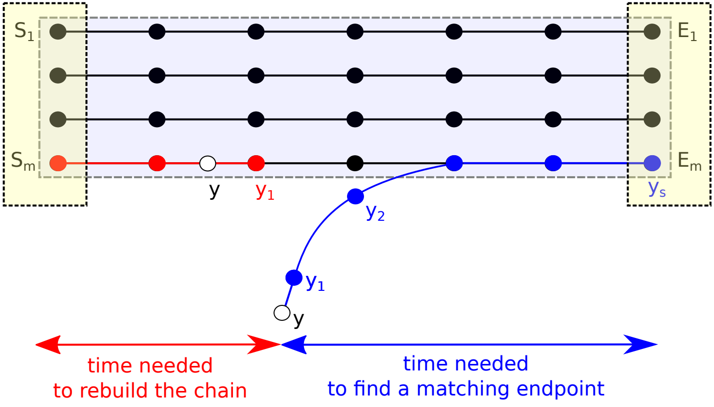


A table without merges is said perfect (*clean*).

Online Procedure is More Complex

Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1:

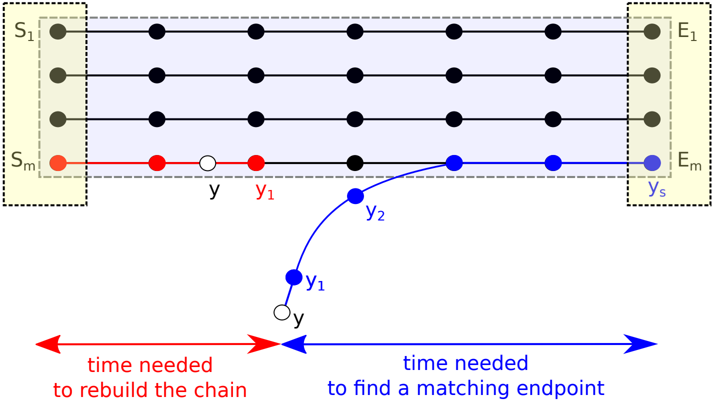
$$y_1 \xrightarrow{f} y_2 \xrightarrow{f} y_3 \xrightarrow{f} \dots y_s$$



Online Procedure is More Complex

Given one output $y \in B$, we compute $y_1 := R(y)$ and generate a chain starting at y_1:

$$y_1 \xrightarrow{f_{t-s}} y_2 \xrightarrow{f_{t-s+1}} y_3 \xrightarrow{f_{t-s+2}} \dots y_s$$



Success Probability of a Table is Bounded

Theorem

Given t and a sufficiently large N, the expected maximum number of chains per perfect rainbow table without merge is:

$$m_{\max}(t) \approx \frac{2N}{t+1}.$$

Theorem

Given t, for any problem of size N, the expected maximum probability of success of a single perfect rainbow table is:

$$P_{\max}(t) \approx 1 - \left(1 - \frac{2}{t+1}\right)^t$$

which tends toward $1 - e^{-2} \approx 86\%$ *when t is large.*

Average Cryptanalysis Time

Theorem

Given N, m, ℓ, and t, the average cryptanalysis time is:

$$T = \sum_{\substack{k=1 \\ c=t-\lfloor \frac{k-1}{\ell} \rfloor}}^{k=\ell t} p_k\left(\frac{(t-c)(t-c+1)}{2} + \sum_{i=c}^{i=t} q_i i\right)\ell +$$

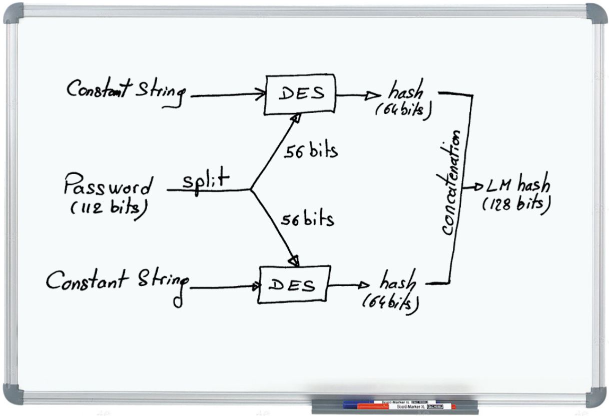
$$\left(1 - \frac{m}{N}\right)^{\ell t}\left(\frac{t(t-1)}{2} + \sum_{i=1}^{i=t} q_i i\right)\ell$$

where

$$q_i = 1 - \frac{m}{N} - \frac{i(i-1)}{t(t+1)}.$$

REAL LIFE EXAMPLES

Windows LM Passwords (Algorithm)



- Win98/ME/2k/XP uses the Lan Manager Hash (LM hash).
- The password is cut in two blocks of 7 characters.
- Lowercase letters are converted to uppercase. Not salted.

Windows LM Hash (Results)

Cracking an alphanumerical password (LM Hash) on a PC. Size of the problem: $N = 8.06 \times 10^{10} = 2^{36.23}$.

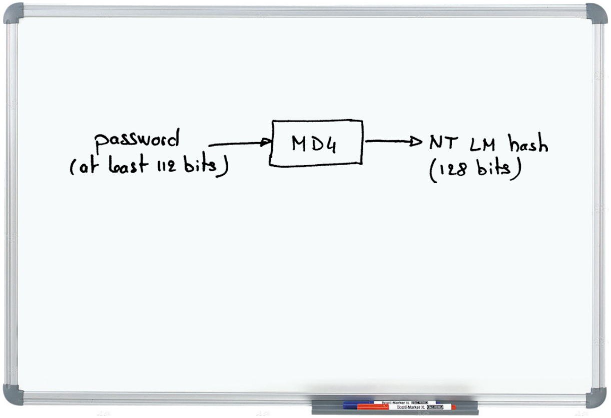
	Brute Force	TMTO
Online Attack (op)	4.03×10^{10}	1.13×10^{6}
Time	2 h 15	0.226 sec
Precalculation (op)	0	1.42×10^{13}
Time	0	33 days
Storage	0	2 GB

Statistics from 10,000 Leaked Hotmail Passwords

Password Type	%
numeric	19%
lower case alpha	42%
mixed case alpha	3%
mixed numeric alpha	30%
other charac	6%

Password Length	%
≤ 7	37%
≤ 8	58%
≤ 9	70%

Windows NT LM Passwords



- Win NT/2000/XP/Vista/Seven uses the NT LM Hash.
- The password is no longer cut in two blocks.
- Lowercase letters are not converted to uppercase. Not salted.

Windows NT LM Hash (Results)

Cracking a 7-char (max) alphanumerical password (NT LM Hash) on a PC. Size of the problem: $N = 2^{41.7}$.

	Brute Force	TMTO
Online Attack (op)	1.78×10^{12}	4.48×10^{7}
Time	99 hrs	9.0 sec
Precalculation (op)	0	6.29×10^{14}
Time	0	1458 days
Storage	0	16 GB

CONCLUSION

- Motivations
- Hellman Tables
- Oechslin Tables
- Real Life Examples
- Conclusion

Limits of Cryptanalytic Time-memory Trade-offs

- A TMTO is never better than a brute force.
- TMTO makes sense in several scenarios.
  - Attack repeated several times.
  - Lunchtime attack.
  - Attacker is not powerful but can download tables.
- Two conditions to perform a TMTO.
  - Reasonably-sized problem.
  - One-way function (or chosen plaintext attack on a ciphertext).