ENEE 459-C
Computer Security

Rainbow tables

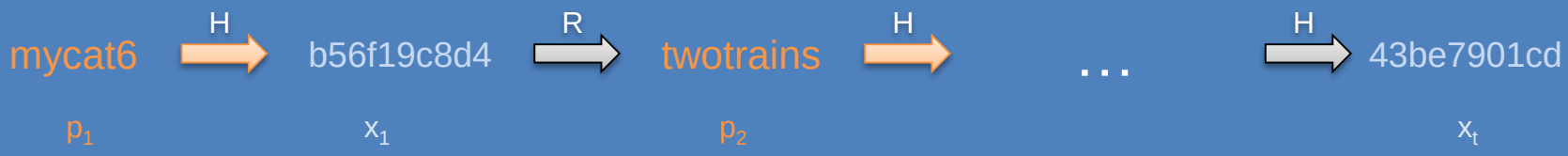
UNIVERSITY OF
MARYLAND

Reduction Function

- A reduction function maps a hash value to a password from a given password space
- Example reduction function p = R(x)
  - Consider 256-bit hash values and 8-character passwords from an alphabet of 64 symbols a_1, a_2, …, a_{64}
  - Split x into 48-bit blocks x_1, x_2, …, x_5 and one 16-bit block x_6
  - Compute $y = x_1 \oplus x_2 \ldots \oplus x_5$
  - Split y into 6-bit blocks y_1, y_2, …, y_8
  - Let $p = a_{y_1}, a_{y_2}, \ldots, a_{y_8}$
- Above method can be generalized to arbitrary password spaces

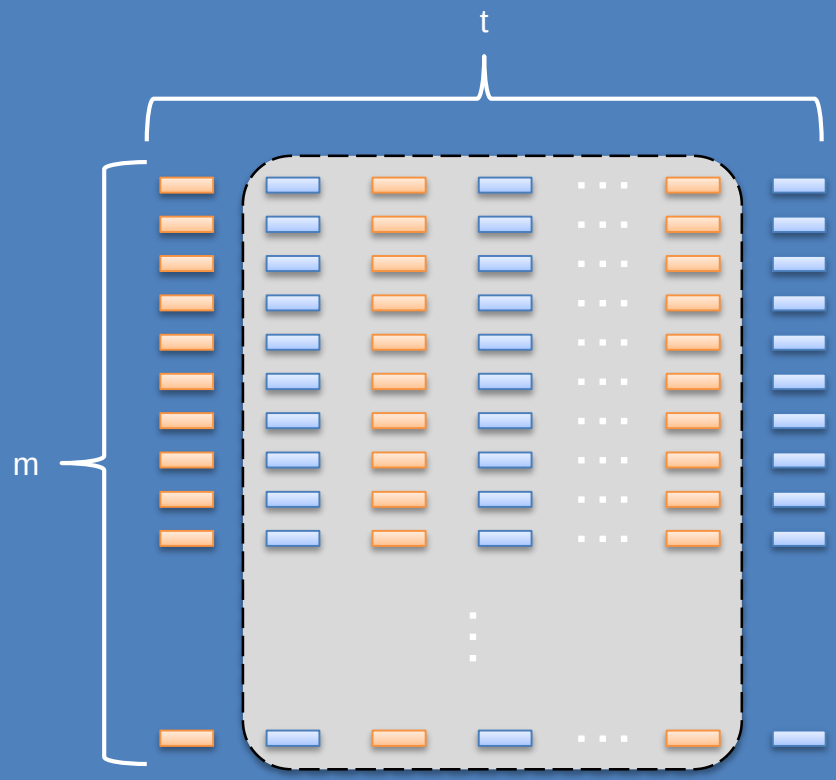
Password Chain

- Sequence of alternating passwords and hash values
  - Start with a random password p_1
  - Use cryptographic hash function H and reduction function R
  - $x_i = H(p_i)$
  - $p_{i+1} = R(x_i)$
  - End with a hash value x_t



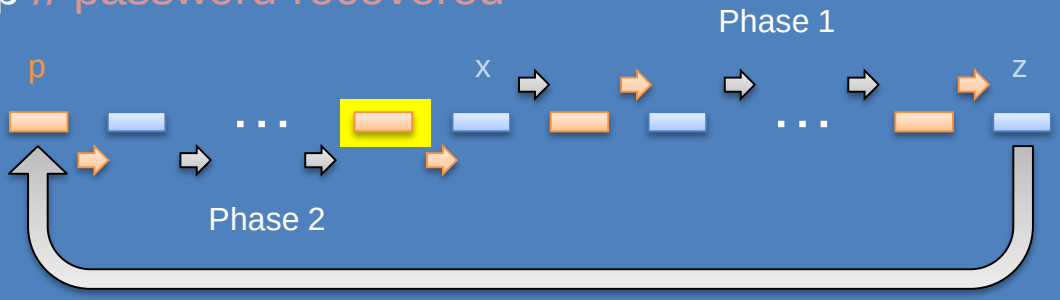
Hellman's Classic Method

- Starting from m random passwords, build m password chains, each of length t
- Because of collisions in the reduction function, the expected number of distinct passwords in a table is less than the theoretical maximum, mt
- Compressed storage:
  - For each chain, keep only first password, p, and last hash value, z
  - Store pairs (z, p) in a dictionary D indexed by hash value z

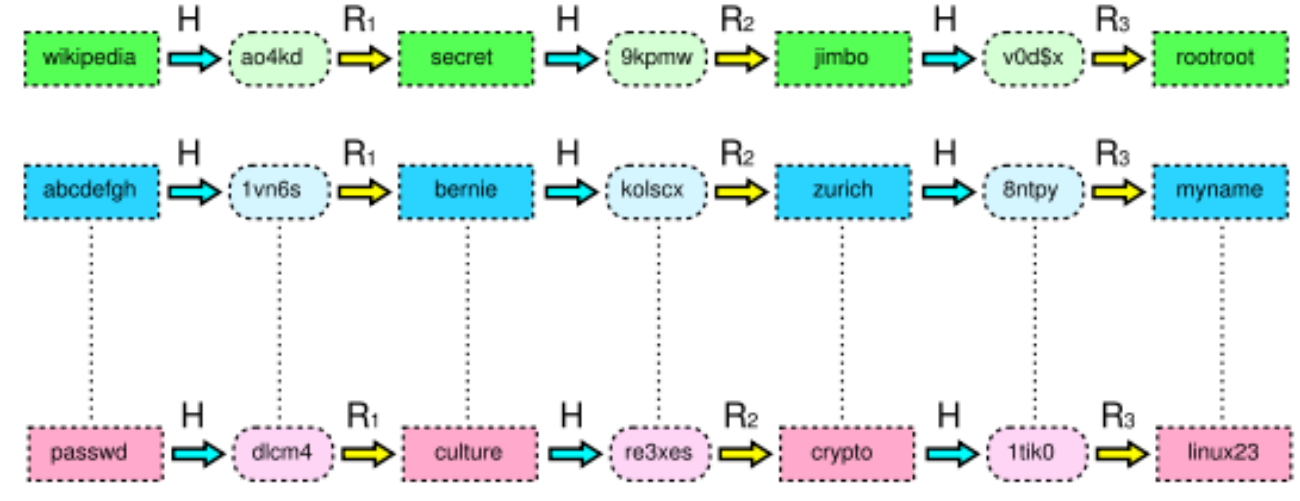


Classic Password Recovery

- Recovery of password with hash value x
- Step 1: traverse the suffix of the chain starting at x
  - y = x
  - while p = D.get(y) is null
    - y = H(R(y)) // advance
    - if i++ > t return “failure” // x is not in the table
- Step 2: traverse the prefix of the chain ending at x
  - while y = H(p) ≠ x
    - p = R(y) // advance
    - if j++ > t return “failure” // x is not in the table
  - return p // password recovered

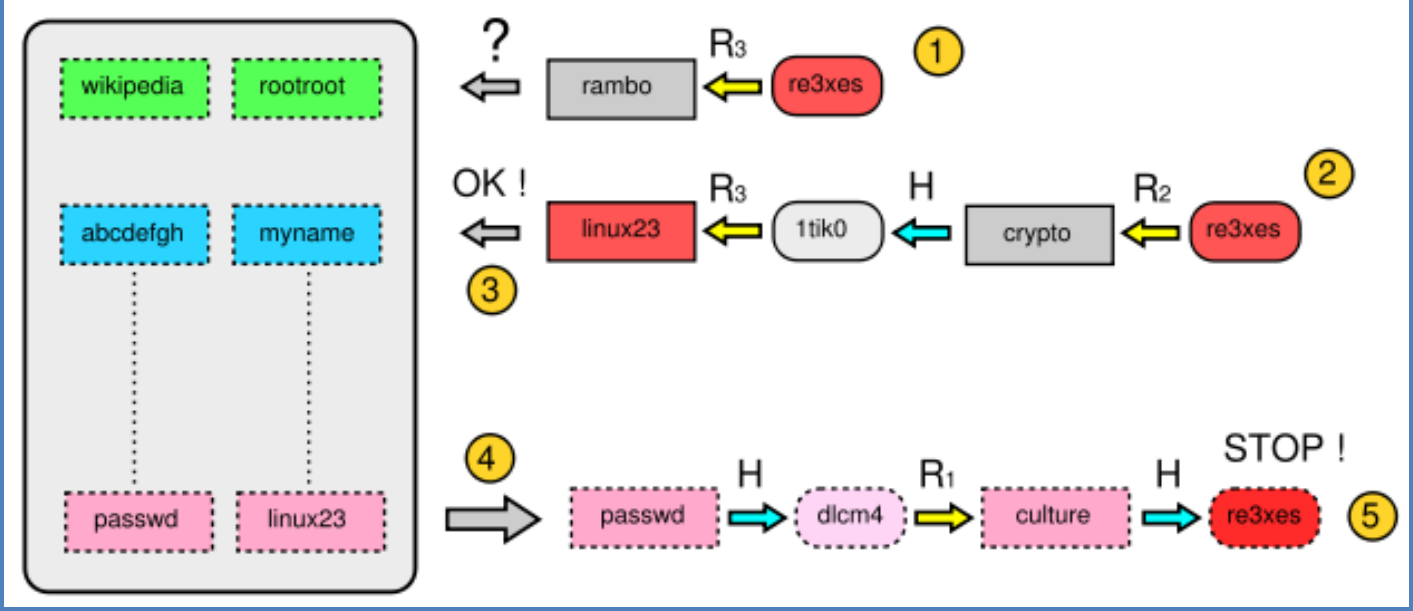


Setting up the table



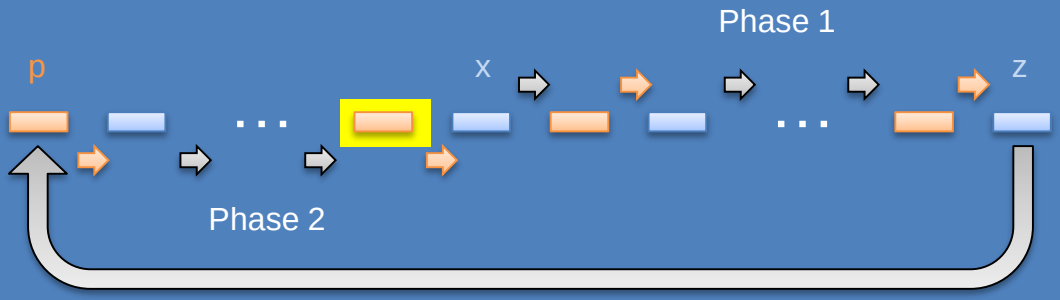
Recovering

- Recovering a password



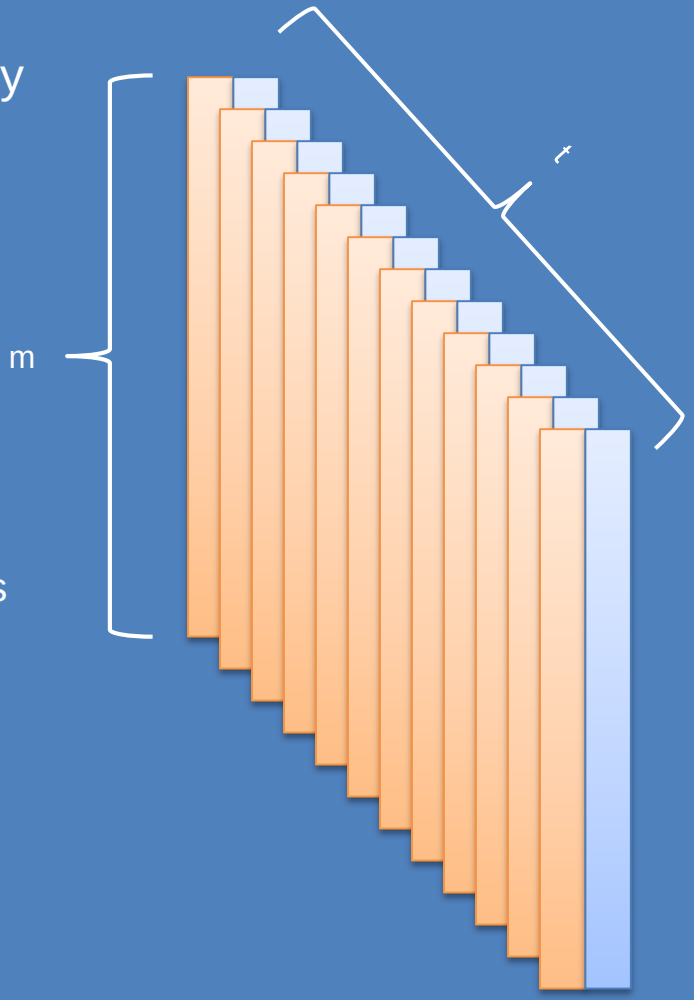
Does it always work?

- First you might never find a matching hash
- What if you find a matching hash?



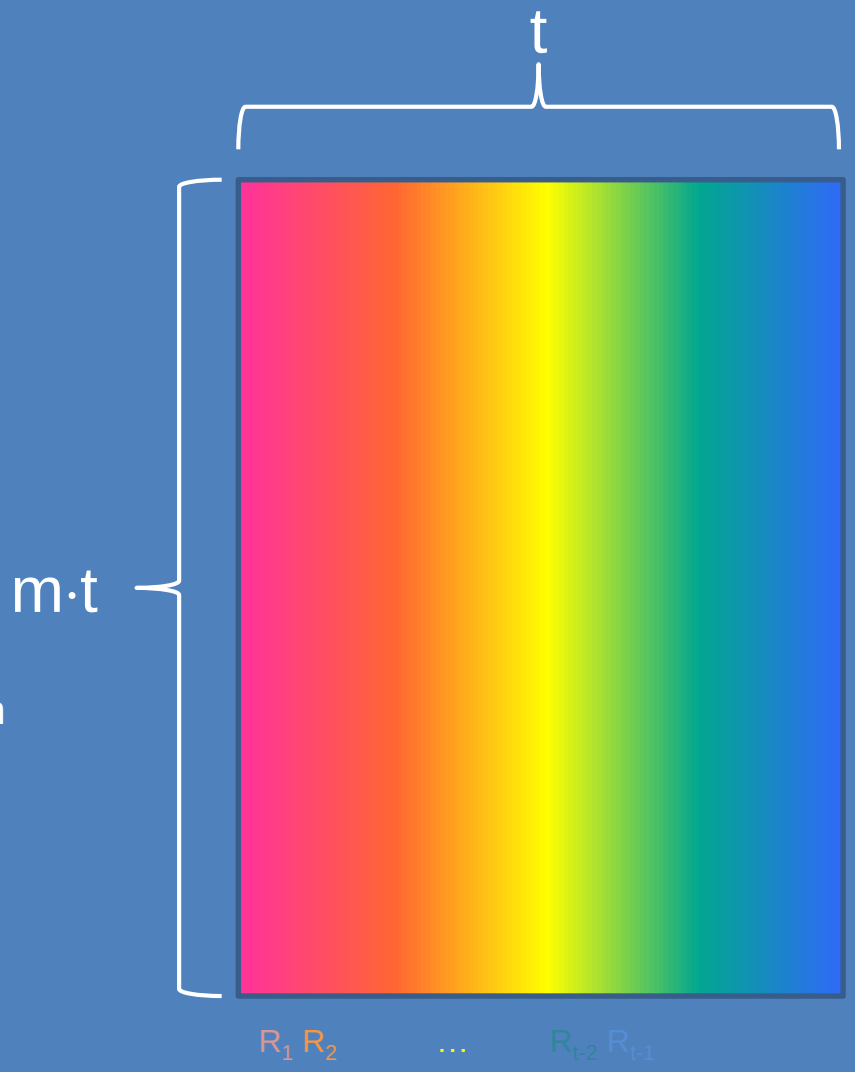
High-Probability Recovery

- For a password space, of size n, the probability P of recovery grows with m and t
- Hellman proved that if $mt^2 = n$, $P \approx 0.8\ \ mt / n$
- E.g., for $m = t = n^{1/3}$, $P \approx 0.8 / n^{1/3}$
- To achieve high success probability, use t tables, each with a different reduction function
- Performance
  - Storage cost: mt cryptographic hash values
  - Recovery cost: t^2 cryptographic hash computations and t^2 dictionary lookups
- Example
  - $m = t = n^{1/3}$
  - $n = 1{,}000{,}000{,}000$
  - $mt = t^2 = 1{,}000{,}000$



Rainbow Table

- Instead of t different tables, use a single table with
  - mt chains of length t
  - A different reduction function at each step
- Visualizing the reduction functions with a gradient of colors yields a rainbow
- Performance
  - Storage cost: mt cryptographic hash values, similar to previous method
  - Recovery cost: $t^2/2$ cryptographic hash computations and t dictionary lookups, lower than previous method



Rainbow Password Recovery

Recovery of password with hash value x

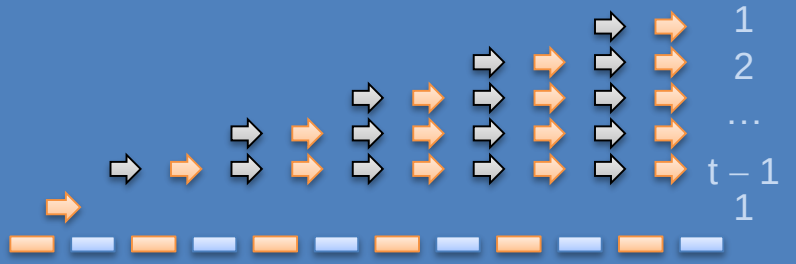
```
for i = t, (t – 1), … , 1
    // Traverse from i to t
    y = x
    for j = i, …, t - 1
        y = H(Rj(y)) // advance
    if p = D.get(y) is not null
        // i is the candidate position
        for j = 1 … i - 1 // Traverse chain from 1 to i
            p = Rj(H(p)) // advance
        if H(p) = x   return p // password recovered
        else return “failure” // x is not in the table
return “failure” // x is not in the table
```

Final loop:
traverse from 1 to i

Inner loop:
traverse from i to t

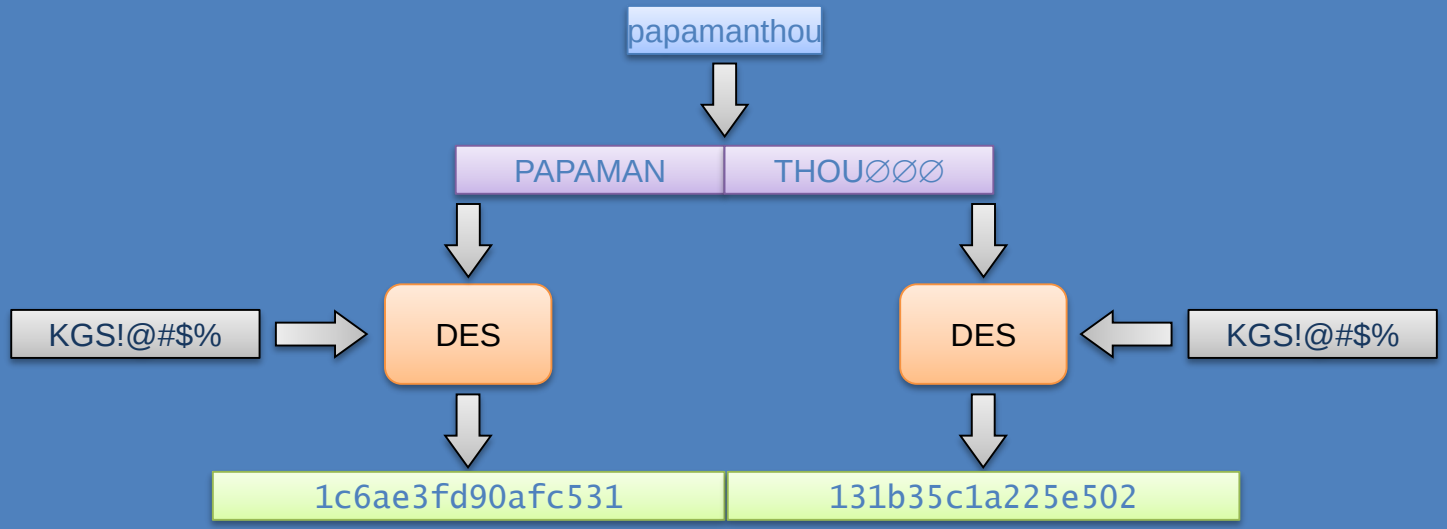
Worst-case number of hash computations

$1 + 2 + \ldots + (t - 1) + 1 \approx t^2/2$



Legacy Windows Passwords

- LAN Manager Hash
  - Convert password to uppercase, null-padded or truncated to length 14
  - Split into two 7-charcter halves
  - Derive a DES key (56 bits) from each half
  - DES-encrypt plaintext KGS!@#$% with each key, resulting in two 8-byte ciphertexts whose concatenation is the 16-byte LM hash
- Supported by all versions of Windows for backward compatibility



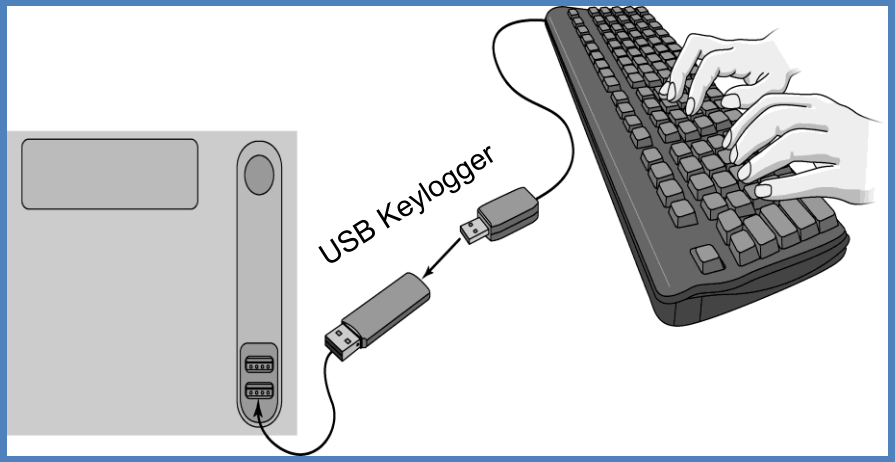
LH Hash Weaknesses

- Password stored unsalted
- Small password space
  - Equivalent to two passwords of 7 characters each
  - Each password from a space of size

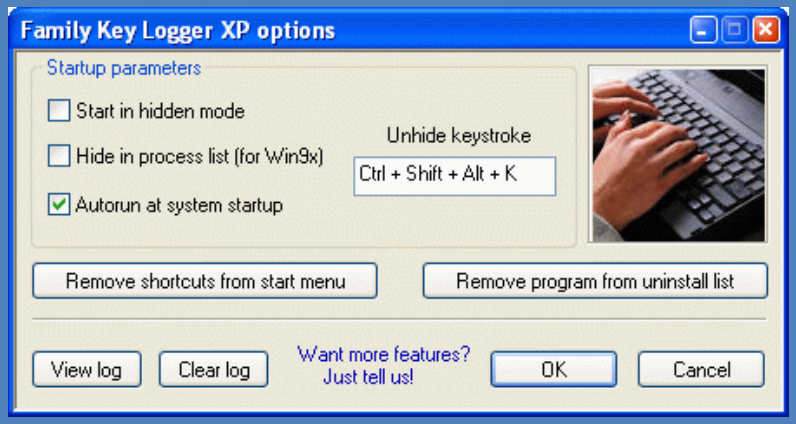
    $68^7 \approx 6.7 \cdot 10^{12}$ (6.7 trillion possible passwords)
- Separate attack on each password performed with rainbow tables
- Experiment on cracking alphanumerical Windows XP passwords
  - 5 rainbow tables
  - each table has 35,000,000 rows and 4,666 columns
  - only rightmost hashes, reduction functions, and generators of random passwords are stored
  - tables use overall 1.4GB space
  - 99.9% success rate
  - 14 seconds recovery time

If Cracking does not Work

Keyloggers



Hardware



Software

Two-factor authentication

- Requires the presentation of two or more of the three authentication factors
- When you use a bank cash card at an ATM you must provide the machine with two factors:
  - your physical card (something you have)
  - with your PIN (something you know)

"Hackers destroyed my entire digital life in the span of an hour"
Mat Honan Wired senior writer

References

- Martin Hellman, A Cryptanalytic Time-Memory Tradeoff, IEEE Trans. Information Theory, 1980
- Philippe Oechslin, Making a Faster Cryptanalytic Time-Memory Trade-Off, CRYPTO, 2003
- Avoine Gildas, Pascal Junod, Philippe Oechslin: Characterization and Improvement of Time-Memory Trade-Off Based on Perfect Tables. ACM Trans. Inf. Syst. Secur. 11(4): (2008)
- Top 10 Password Crackers, http://sectools.org/crackers.html
- Cain & Abel, http://www.oxid.it/cain.html
- PWDump, http://www.foofus.net/fizzgig/pwdump/
- Ophcrack, http://lasecwww.epfl.ch/~oechslin/projects/ophcrack/
- Winrtgen, http://www.oxid.it/projects.html
- Mac OS X password hashes, http://www.macshadows.com/kb/index.php?title=Mac_OS_X_password_hashes
- Cardinale, Giacchetti, Giovannetti, Hacking dei Sistemi: Password