An Oblivious Password Cracking Server

Aureliano Calvo - *aureliano.calvo@coresecurity.com*
Ariel Futoransky - *ariel.futoransky@coresecurity.com*
Carlos Sarraute - *carlos.sarraute@coresecurity.com*

Corelabs
Core Security Technologies

2012-08-31

Agenda

CORE SECURITY

Intro

- Reversing cryptographically-strong hashes is really resource intensive.
- What if a resource constrained hacker wants to crack a password?
- Hash reversing tables to the rescue!

CORE SECURITY

Intro

- Reversing cryptographically-strong hashes is really resource intensive.
- What if a resource constrained hacker wants to crack a password?
- Hash reversing tables to the rescue!

Intro

- Reversing cryptographically-strong hashes is really resource intensive.
- What if a resource constrained hacker wants to crack a password?
- Hash reversing tables to the rescue!

Space-Time trade-off

- Rainbow tables allow to use $O(N^{2/3})$ space and $O(N^{2/3})$ time to invert a hash where N is the size of the pre-image.
  - For instance a rainbow table used to break *LM* takes 34GBytes.
- What if the password cracker cannot store the tables?

Space-Time trade-off

- Rainbow tables allow to use $O(N^{2/3})$ space and $O(N^{2/3})$ time to invert a hash where N is the size of the pre-image.
  - For instance a rainbow table used to break *LM* takes 34GBytes.
- What if the password cracker cannot store the tables?

Space-Time trade-off

- Rainbow tables allow to use $O(N^{2/3})$ space and $O(N^{2/3})$ time to invert a hash where N is the size of the pre-image.
  - For instance a rainbow table used to break *LM* takes 34GBytes.
- What if the password cracker cannot store the tables?

Privacy implications

- A server serves the tables.
- But......
- Now the server can know the passwords being cracked :(.

Privacy implications

- A server serves the tables.
- But......
- Now the server can know the passwords being cracked :(.

Privacy implications

- A server serves the tables.
- But......
- Now the server can know the passwords being cracked :(.

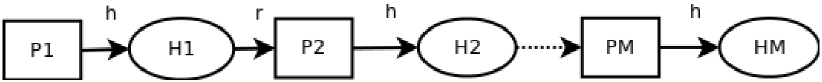
Agenda

CORE SECURITY

Agenda

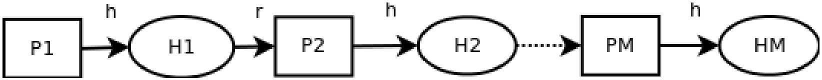
CORE SECURITY

Hellman Tables



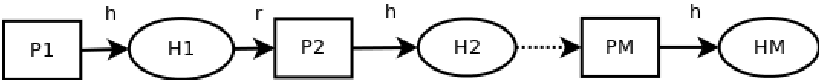
- Store initial password and ending hash for each chain (length *M*).
- Each table stores *M* chains.
- *M* tables. Each one uses its own reduction function.

Hellman Tables



- Store initial password and ending hash for each chain (length *M*).
- Each table stores *M* chains.
- *M* tables. Each one uses its own reduction function.

Hellman Tables


$$\begin{tikzcd}\boxed{P1} \arrow[r, "h"] & H1 \arrow[r, "r"] & \boxed{P2} \arrow[r, "h"] & H2 \arrow[r, dotted] & \boxed{PM} \arrow[r, "h"] & HM\end{tikzcd}$$

- Store initial password and ending hash for each chain (length *M*).
- Each table stores *M* chains.
- *M* tables. Each one uses its own reduction function.

Hellman Tables with Distinguished End-Points

Differences with Hellman Tables

- End-points have a property that distinguishes them
  - Such as number of leading 0s
- The expected length of chains is M.
- Minimizes number of queries made to the tables (just M).

Hellman Tables with Distinguished End-Points

Differences with Hellman Tables

- End-points have a property that distinguishes them
  - Such as number of leading 0s
- The expected length of chains is M.
- Minimizes number of queries made to the tables (just M).

Hellman Tables with Distinguished End-Points

Differences with Hellman Tables

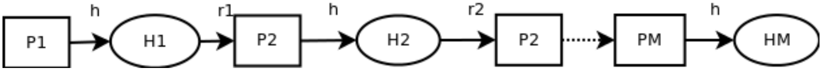
- End-points have a property that distinguishes them
  - Such as number of leading 0s
- The expected length of chains is M.
- Minimizes number of queries made to the tables (just M).

Hellman Tables with Distinguished End-Points

Differences with Hellman Tables

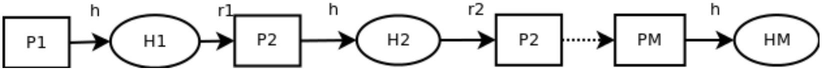
- End-points have a property that distinguishes them
  - Such as number of leading 0s
- The expected length of chains is M.
- Minimizes number of queries made to the tables (just M).

Rainbow Tables



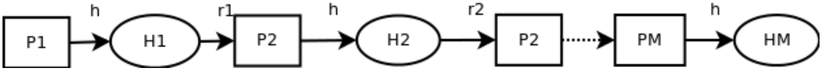
- Just one table with M^2 entries.
- Collisions are avoided by using a different reduction function for each step.
- This is the most popular kind of hash reversing tables.

Rainbow Tables



- Just one table with M^2 entries.
- Collisions are avoided by using a different reduction function for each step.
- This is the most popular kind of hash reversing tables.

Rainbow Tables



- Just one table with M^2 entries.
- Collisions are avoided by using a different reduction function for each step.
- This is the most popular kind of hash reversing tables.

Table sizes

- Number of chains = M^2
  - M chains for each of the M tables for Hellman Tables and Hellman Tables With Distinguished Endpoints.
- Chain length = M
  - Average chain length for Hellman Tables With Distinguished Endpoints.
- $M = -\sqrt[3]{\ln(1-\alpha)\cdot N}$ where:
  - N is the size of the preimage
  - α is the probability that a preimage can be found using the table.

Table sizes

- Number of chains = M^2
    - M chains for each of the M tables for Hellman Tables and Hellman Tables With Distinguished Endpoints.
- Chain length = M
    - Average chain length for Hellman Tables With Distinguished Endpoints.
- $M = -\sqrt[3]{\ln(1-\alpha)\cdot N}$ where:
    - N is the size of the preimage
    - α is the probability that a preimage can be found using the table.

CORE SECURITY

Table sizes

- Number of chains = M^2
  - M chains for each of the M tables for Hellman Tables and Hellman Tables With Distinguished Endpoints.
- Chain length = M
  - Average chain length for Hellman Tables With Distinguished Endpoints.
- $M = -\sqrt[3]{\ln(1-\alpha)\cdot N}$ where:
  - N is the size of the preimage
  - α is the probability that a preimage can be found using the table.

Table sizes

- Number of chains = M^2
  - M chains for each of the M tables for Hellman Tables and Hellman Tables With Distinguished Endpoints.
- Chain length = M
  - Average chain length for Hellman Tables With Distinguished Endpoints.
- $M = -\sqrt[3]{\ln(1-\alpha) \cdot N}$ where:
  - N is the size of the preimage
  - α is the probability that a preimage can be found using the table.

Table sizes

- Number of chains = M^2
  - M chains for each of the M tables for Hellman Tables and Hellman Tables With Distinguished Endpoints.
- Chain length = M
  - Average chain length for Hellman Tables With Distinguished Endpoints.
- $M = -\sqrt[3]{\ln(1-\alpha) \cdot N}$ where:
  - N is the size of the preimage
  - α is the probability that a preimage can be found using the table.

Table sizes

- Number of chains = M^2
  - M chains for each of the M tables for Hellman Tables and Hellman Tables With Distinguished Endpoints.
- Chain length = M
  - Average chain length for Hellman Tables With Distinguished Endpoints.
- $M = -\sqrt[3]{\ln(1-\alpha) \cdot N}$ where:
  - N is the size of the preimage
  - α is the probability that a preimage can be found using the table.

Table sizes

- Number of chains = M^2
  - M chains for each of the M tables for Hellman Tables and Hellman Tables With Distinguished Endpoints.
- Chain length = M
  - Average chain length for Hellman Tables With Distinguished Endpoints.
- $M = -\sqrt[3]{\ln(1-\alpha) \cdot N}$ where:
  - N is the size of the preimage
  - α is the probability that a preimage can be found using the table.

Agenda

Problem to be solved

- Ask for a bit in a database stored by a server without revealing the requested bit to the server.
- Without sending the entire database (!).

Problem to be solved

- Ask for a bit in a database stored by a server without revealing the requested bit to the server.
- Without sending the entire database (!).

Problem to be solved

"Something being imposible does not imply that it has never been done" (Fernando Russ)

Classic PIR

Eyal Kushilevitz and Rafail Ostrovsky
Replication is not needed: Single database, computationally-private information retrieval.
Proceedings of the 38th Annu. IEEE Symp. on Foudations of Computer Science, pages:364–373, 1997.

CORE SECURITY

Classic PIR - Complexities

- Server processing complexity: $O(n)$.
- Client processing complexity: $O(\sqrt{n})$.
- Transfer complexity: $O(\sqrt{n})$.
- Where n is the number of bits in the database.

Classic PIR - Complexities

- Server processing complexity: $O(n)$.
- Client processing complexity: $O(\sqrt{n})$.
- Transfer complexity: $O(\sqrt{n})$.
- Where n is the number of bits in the database.

Classic PIR - Complexities

- Server processing complexity: $O(n)$.
- Client processing complexity: $O(\sqrt{n})$.
- Transfer complexity: $O(\sqrt{n})$.
- Where n is the number of bits in the database.

Classic PIR - Complexities

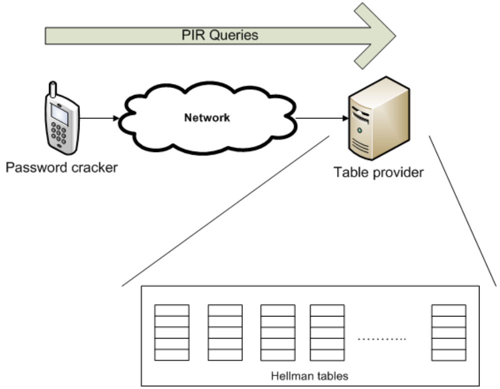
- Server processing complexity: $O(n)$.
- Client processing complexity: $O(\sqrt{n})$.
- Transfer complexity: $O(\sqrt{n})$.
- Where n is the number of bits in the database.

Agenda

CORE SECURITY

Our Work

Hellman Tables With Distinguished Endpoints as Hash-Reversing Tables queried using Classic PIR.



Calculating the tables

- (*begin*, *end*) pairs are stored in a closed hash table.
- The size of each table is βM, where $\beta > 1$.
- Each entry has an index, representing the initial plain-text, and the end-of-chain hash.
- The key of the entry is the end-of-chain hash.
- Hash-Table Collisions are discarded when the tables are generated.

Calculating the tables

- ($begin$, end) pairs are stored in a closed hash table.
- The size of each table is βM, where $\beta > 1$.
- Each entry has an index, representing the initial plain-text, and the end-of-chain hash.
- The key of the entry is the end-of-chain hash.
- Hash-Table Collisions are discarded when the tables are generated.

Calculating the tables

- $(begin, end)$ pairs are stored in a closed hash table.
- The size of each table is βM, where $\beta > 1$.
- Each entry has an index, representing the initial plain-text, and the end-of-chain hash.
- The key of the entry is the end-of-chain hash.
- Hash-Table Collisions are discarded when the tables are generated.

Calculating the tables

- ($begin$, end) pairs are stored in a closed hash table.
- The size of each table is βM, where $\beta > 1$.
- Each entry has an index, representing the initial plain-text, and the end-of-chain hash.
- The key of the entry is the end-of-chain hash.
- Hash-Table Collisions are discarded when the tables are generated.

Calculating the tables

- (*begin*, *end*) pairs are stored in a closed hash table.
- The size of each table is βM, where $\beta > 1$.
- Each entry has an index, representing the initial plain-text, and the end-of-chain hash.
- The key of the entry is the end-of-chain hash.
- Hash-Table Collisions are discarded when the tables are generated.

Reversing a Hash

- The client calculates the *M* ends-of-chain for the hash being reversed. One for each table.
- It requests the corresponding server buckets, via *M* PIR requests.
- For each response, it checks if the end-of-chain is the one needed. If so, it navigates the chain looking for the preimage of the hash being reversed.

Reversing a Hash

- The client calculates the M ends-of-chain for the hash being reversed. One for each table.
- It requests the corresponding server buckets, via M PIR requests.
- For each response, it checks if the end-of-chain is the one needed. If so, it navigates the chain looking for the preimage of the hash being reversed.

Reversing a Hash

- The client calculates the M ends-of-chain for the hash being reversed. One for each table.
- It requests the corresponding server buckets, via M PIR requests.
- For each response, it checks if the end-of-chain is the one needed. If so, it navigates the chain looking for the preimage of the hash being reversed.

Complexity

- Single hash-reversing client processing complexity: $O(M^2)$.
- Single hash-reversing server processing complexity: $O(M^2)$.
- Transfer complexity: $O(M^{3/2})$.
- Server storage complexity: $O(M^2)$.
- Table calculation complexity: $O(M^3)$.

Complexity

- Single hash-reversing client processing complexity: $O(M^2)$.
- Single hash-reversing server processing complexity: $O(M^2)$.
- Transfer complexity: $O(M^{3/2})$.
- Server storage complexity: $O(M^2)$.
- Table calculation complexity: $O(M^3)$.

Complexity

- Single hash-reversing client processing complexity: $O(M^2)$.
- Single hash-reversing server processing complexity: $O(M^2)$.
- Transfer complexity: $O(M^{3/2})$.
- Server storage complexity: $O(M^2)$.
- Table calculation complexity: $O(M^3)$.

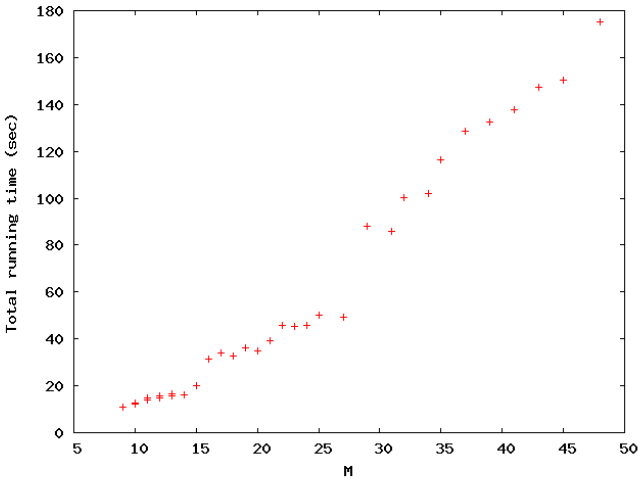
Complexity

- Single hash-reversing client processing complexity: $O(M^2)$.
- Single hash-reversing server processing complexity: $O(M^2)$.
- Transfer complexity: $O(M^{3/2})$.
- Server storage complexity: $O(M^2)$.
- Table calculation complexity: $O(M^3)$.

Complexity

- Single hash-reversing client processing complexity: $O(M^2)$.
- Single hash-reversing server processing complexity: $O(M^2)$.
- Transfer complexity: $O(M^{3/2})$.
- Server storage complexity: $O(M^2)$.
- Table calculation complexity: $O(M^3)$.

Experimental Results



Agenda

CORE SECURITY

Future Work

- Optimize using other PIR schemes
- Paralelize cracking
- Devise a way to use the current rainbow tables with a PIR scheme
- Use PIR to reverse hashes in the client while the server brute forces it (no tables stored).

Future Work

- Optimize using other PIR schemes
- Paralelize cracking
- Devise a way to use the current rainbow tables with a PIR scheme
- Use PIR to reverse hashes in the client while the server brute forces it (no tables stored).

CORE SECURITY

Future Work

- Optimize using other PIR schemes
- Paralelize cracking
- Devise a way to use the current rainbow tables with a PIR scheme
- Use PIR to reverse hashes in the client while the server brute forces it (no tables stored).

Future Work

- Optimize using other PIR schemes
- Paralelize cracking
- Devise a way to use the current rainbow tables with a PIR scheme
- Use PIR to reverse hashes in the client while the server brute forces it (no tables stored).

Any Question?

Thank you!