Outline

- Primitives based on cryptographic hash functions
  - One-way hash chain
  - Merkle hash tree
  - Client puzzles
- Bloom filters
  - Based on a different type of hash functions

One-Way Hash Chain

- Used for many network security applications
  - Example: S/Key
- Good for authentication of the hash values

\[ K_0, K_1, ..., K_n = R \]

\[ K_i = F(K_{i-1}) \]

\[ F \text{ hash function} \]
Merkle Hash Tree

- A binary tree over data values
  - For authentication purpose
- The root is the commitment of the Merkle tree
  - Known to the verifier.
- Example
  - To authenticate $k_2$, send $(k_2, m_3, m_{01}, m_{47})$
  - Verify $m_{07} = h(h(m_{01}) || h(f(k_2) || m_3) || m_{47})$

Bloom Filters

- It’s used to verify that some data is not in the database (mismatch)
  - List of bad credit card numbers
  - Useful when the data consumes a very small portion of search space
- A bloom filter is a bit string
- $k$ hash functions that map the data into $n$ bits in the bloom filter

A Simple Example

- Use a bloom filter of 16 bits
  - $H1(key) = key \mod 16$
  - $H2(key) = key \mod 14 + 2$
- Insert numbers 27, 18, 29 and 28

```
1   1   1   1   1   1
1   2   3   4   5   6   7   8   9   10   11   12   13   14   15
```
- Check for 22:
  - $H1(22) = 6, H2(22) = 10$ (not in filter)
- Check for 51
  - $H1(51) = 3, H2(51) = 11$ (false positive)
Probability of False Positives

- Consider an $m$-bit Bloom filter with $k$ hash functions
  - After inserting $n$ elements
  - Exercise

Client Puzzles

- Juels and Brainard client puzzle construction
  - Use pre-image of crypto hash functions
  - See T02.2.x-ClientPuzzles.ppt

- Aura, Nikander, and Leiwo client puzzle construction
  - Use special image of crypto hash functions

Client Puzzles w/ Special Images of Hash Functions

- $C \rightarrow S$: Hello
- $S \rightarrow C$: $N_S$
- $C \rightarrow S$: $C, N_C, N_S, X$
- $S$: verify $h(C, N_C, N_S, X)$ has $k$ leading zero's

\[ h(C, N_C, N_S, X) = \text{leading zero's} \]

Where $h$ is a cryptographic hash function (e.g., MD5 or SHA)

- $C$ = the client's identity
- $N_C$ = the server's nonce
- $N_S$ = the server's nonce
- $X$ = the solution of the puzzle
- $k$ = the puzzle difficulty level
- 000...000 = the $k$ first bits of the hash value must be zero
- Y = the rest of the hash value, may be anything
Client Puzzles w/ Special Images of Hash Functions (Cont’d)

• Exercise
  – What’s the expected cost of finding a puzzle solution?
  – Compare with the pre-image based puzzle construction.

New Client Puzzle Outsourcing Techniques for DoS Resistance

Brent Waters, Ari Juels, J. Alex Halderman and Edward W. Felten

Motivation

• Client puzzle mechanism can become the target of DoS attacks
  – Servers have to validate solutions which require resources
• Puzzles must be solved online
  – User time is more important than CPU time
Properties of the Proposed Solution

- The creation of puzzles is outsourced to a secure entity, the bastion
  - Create puzzle with no regard to which server is going to use them
- Verifying puzzle solutions is a table lookup
- Clients can solve puzzles offline ahead of time
- A puzzle solution gives access to a virtual channel for a short time period

Puzzle Properties

- Unique puzzle solutions
  - Each puzzle has a unique solution
- Per-channel puzzle distribution
  - Puzzles are unique per each (server, channel, time period) triplet
- Per-channel puzzle solution
  - If a client has a solution for one channel, he can calculate a solution for another server with the same channel easily

\[ C: \text{A group of prime numbers with generator } g. \]
\[ \text{Pick } x_0 \in Z_q \]
\[ a_{c,t} \leftarrow q_y, \left( x_0 + 1 \right) \text{mod q} \]
\[ \text{Let } g_{a_{c,t}} = g^{a_{c,t}}, \text{puzzle } x_{c,t} = (g_{a_{c,t}}, r_{c,t}) \]
\[ \pi_{c,t} = Y, \pi_{c,t}^{\sigma_{c,t}} \text{ for all channels} \]
\[ \text{Enumerate } l \text{ values to solve } a_{c,t} \]
\[ \text{Take the easy way } \]
\[ \sigma_{c,t} = Y_{1}^{(a_{c,t} \cdot l)} \]
\[ g_{a_{c,t}} = g_{a_{c,t} \cdot l} \]
System Description

- Solutions for puzzles are only valid for the time period $t$. (Order of minutes)
  - Client:
    - During $T_c$ download puzzles for $T_{c+1}$ and solve
    - Check to see if server has a public key
    - If so append puzzle solutions to messages
  - Server:
    - During $T_c$ download and solve all puzzles for $T_{c+1}$
    - If server is under attack only accept requests that have valid tokens
    - Checking puzzle solution is a simple table lookup

Communication

- Client uses option field in TCP SYN to relay the token
- Only the first 48 bits of the solution is used
- The server determines the virtual channel
- Server limits new connection per channel