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
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 $(k_2, m_3, m_0, m_47)$
  - Verify $m_07 = h(h(m_0 || h(f(k_2)||m_3)||m_47))$

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
  
<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

  - Check for 22:
    - $H1(22) \sim 6, H2(22) \sim 10$ (not in filter)
  - Check for 51
    - $H1(51) \sim 3, H2(51) \sim 11$ (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

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

G: A group of prime numbers with generator g.

Pick $r_{c,t} \in \mathbb{Z}_q$

Let $g_{c,t} = g^{r_{c,t}}$, puzzle $x_{c,t} = (g_{c,t}, r_{c,t})$

$\pi_{c,t}$ for all channels

Enumerate l values to solve $a_{c,t}$

Solution is $\sigma_{c,t} = Y_1^{r_{c,t}}$

Take the easy way $\sigma_{c,t} = g_{c,t}^{l}$
System Description

- Solutions for puzzles are only valid for the time period \( t \). (Order of minutes)
- **Client:**
  - During \( T_i \), download puzzles for \( T_{i+1} \) and solve
  - Check to see if server has a public key
  - If so append puzzle solutions to messages
- **Server:**
  - During \( T_i \), download and solve all puzzles for \( T_{i+1} \)
  - 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
Resilience Against Attacks

- 2.1 GHz Pentium can process 1024-bit DH key in 3.7ms.
- With 5% recourse it can populate tokens for 16,000 virtual channels.
- If \( s = 2 \), every client can solve at least one puzzle and half of them can solve at least two
  - If attacker has 50 zombie machines, it can create \( 2^*50^*2 = 200 \) puzzle solutions occupying 1.25% of the channels
  - Probability of a benign user not getting a normal channel < 6.25%

Experiment

- Puzzle checking (table lookup) is implemented at kernel lvl
- After the routing and before the packet reaches higher level protocols like TCP
- Simulate conventional puzzles by replacing the lookup code with a SHA-1 hash computation
- Simulate \textit{syncookies} by allowing Linux to send an ACK packet back