Android Authentication and Billing System

Robert Sy, Steven Fierro, Wail Alkowaileet

Outline

- Problem Statement.
- Challenges.
- Existing Solutions.
- Our Solution.

Problem Statement

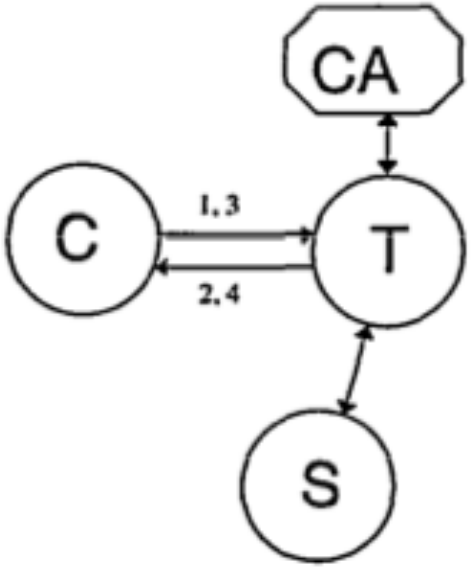
- Providing a secure authentication mechanism and a trustable billing process over networks with a focus towards ad-hoc mobile communication.

Challenges

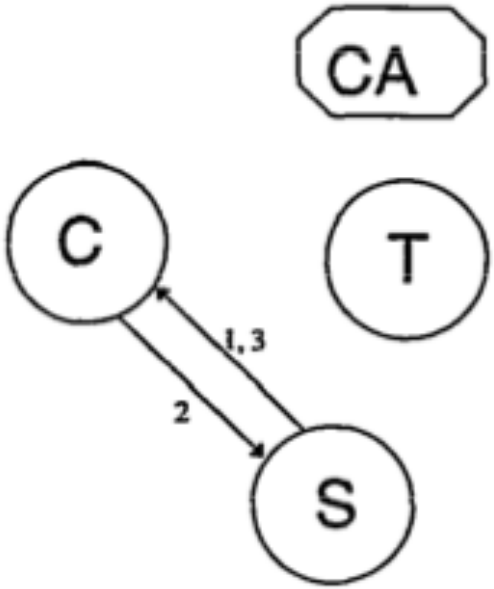
- Mobile phone limitations: Energy, processing power.
- Developing trust among network nodes.
- Creating a protocol that provides non-repudiation in terms of billing accountability.
- Realistic applicability.
- General security issues. (Replay attacks, unforgeability, MITM...etc).

Existing Solutions:

Ticket Based Service Access for the Mobile User []



Step 1: Client acquires a ticket for a particular service offered by a server from the ticket distribution server



Step 2: Client uses the ticket acquired in step 1 to access the service.

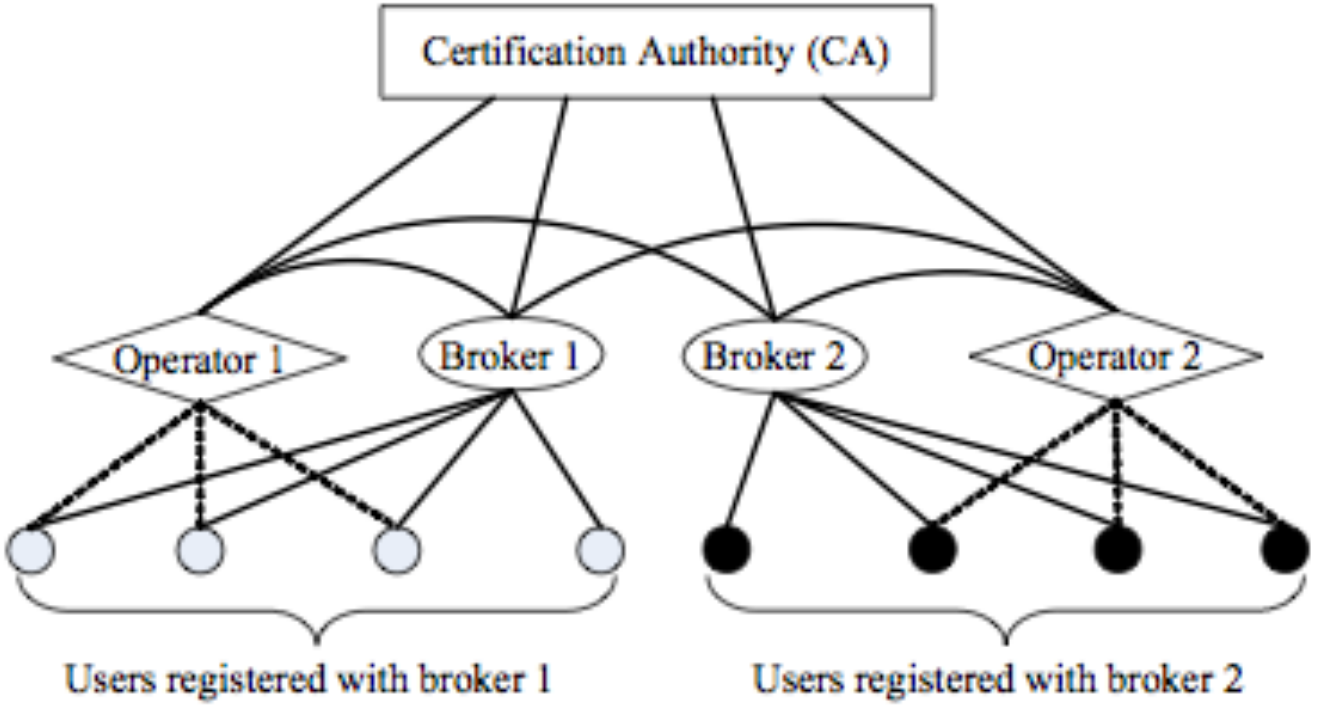
Existing Solutions:

Ticket Based Service Access for the Mobile User []

- Prepaid protocol.
- Kerberos like.
- Needs to contact the server for each time to get a new ticket.
- Single-point of failure.
- Large load for the ticketing service.
- Increased risk with increased ticket values.

Existing Solutions:

A secure authentication and billing architecture for wireless mesh networks []



Existing Solutions:

A secure authentication and billing architecture for wireless mesh networks []

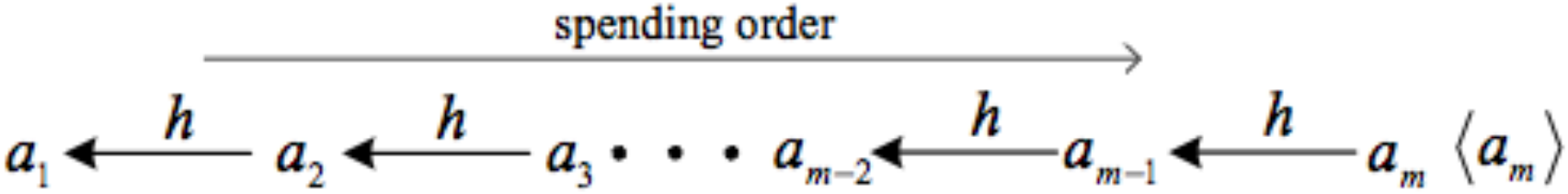
- This system is analogous to a user (client), merchant (operator) and card issuing bank (broker).
- The merchants and the user trust the banks
- The users have *universal passes* (UPASS) which are akin to credit cards.
- The users are able to use the UPASS at any operator in order to use the operator-provided data access service.

Our Solution

- Based on the UPASS hash chaining scheme.
- Each user is managed by a broker and have a shared-secret that is unique for each user-broker pair.

Our Solution

- Hash Chaining:
  - The client generates a random secret a_m and recursively hashes *m-1* times. Each hash value in the chain is a *token*.
  - $a_1 = h(h(h(...h(a_m)...)))$
  - Each token within the hash chain is worth certain amount of data transfer.



Our Solution

- Uses hash chaining along with PKI-signing to generate unforgeable tokens.
- Less load on the *Authentication Server*.
- Less risk on players since each token is worth less.
- Tradeoff between risk and communication overhead because of the value of each ticket. (Less risk --> smaller value --> more communication).