Key Agreement in Ad-hoc Networks

N. Asokan and P. Ginzboorg

Presented by

Chuk Yang Seng

Introduction

- Ad-hoc Key Agreement Scenario:
  - Small group of people at a conference in a room
  - Wireless network session during the meeting
  - **Share information securely so that no one outside the room can eavesdrop**
  - People in the room know and trust one another personally
  - Lack of infrastructure, so no means of digitally identify and authenticating one another, and public keys techniques are not relevant

Solution

- Location-based key agreement
  - Only those present in the room can read the messages
- Choosing a fresh password and share it among those in the room
- Long random password are difficult to use
- Natural language phrase are more user friendly but weak

- Use a weak shared password to derive a strong shared session key

- Password Authenticated Key Exchange

Password Authenticated Key Exchange

- Desirable properties:
  - Secrecy
  - Contributory key agreement
  - Tolerance to disruption: Intruders who can insert messages but cannot modify or delete messages sent (!!!)
- Existing method Encrypted Key Exchange (EKE) by Bellovin and Merrit
  - EKE is non-contributory

Password authenticated Diffie-Hellman key exchange

- Recall Diffie Hellman:
  - **A** and **B** agree on a prime p and a generator g where $g \in \mathbb{Z}_p^*$
  - They randomly choose secrets S_A and S_B such that $S_A, S_B \in \mathbb{Z}_p^*$
    1. **A** $\rightarrow$ **B**: g^{S_A}
    2. **B** $\rightarrow$ **A**: g^{S_B}
  - $K = g^{S_A S_B}$

- Password authenticated Diffie-Hellman

  1. **A** $\rightarrow$ **B**: **A**, $P(g^{S_A})$

  2. **B** $\rightarrow$ **A**: $P(g^{S_B}), K(C_b)$

     – $K = g^{S_A S_B}$

  3. **A** $\rightarrow$ **B**: $K(C_a, C_b)$

  4. **B** $\rightarrow$ **A**: $K(C_a)$

Multi-party version

- By Steiner et al
- Each participants, $M_1, M_2, \ldots, M_n$ share a password P
- $\forall i$: M_i generates random quantity S_i
- Goal: All who knew P will end up with a shared session key $K = g^{S_1 S_2 \ldots S_n}$

Multi-party version

1. $M_i \rightarrow M_{i+1} : g^{S_1 S_2 \ldots S_i}, i = 1, \ldots, n-2$ in sequence

2. $M_{n-1} \rightarrow ALL : \pi = g^{S_1 S_2 \ldots S_{n-1}}$, broadcast

3. $M_i \rightarrow M_n : P(c_i), i = 1, \ldots, n-1$, in parallel, where $c_i = \pi^{\widehat{S}_i / S_i}$ and $\widehat{S}_i$ is a blinding factor randomly chosen by M_i

4. $M_n \rightarrow M_i : (c_i)^{S_n}, i = 1, \ldots, n-1$, in parallel

5. $M_i \rightarrow ALL : M_i, K(M_i, H(M_1, M_2, \ldots, M_n))$, for some i, broadcast

Multi-party version

- The protocol provides perfect forward secrecy to all players
- It is contributory
- Partially resilient to disruptions:
  - M_n can disrupt the protocol completely
  - Any others can send out a random quantity at stage 1
  - If M_i sent a random quantity, then $M_1, \ldots, M_{i-1}$ will not be able to compute the session key but $M_{i+1}, \ldots, M_n$ can.

- Inefficient:
  - Stage 1 takes n-2 communication steps
  - It is not clear how (4) can be done in parallel

Fault tolerant Diffie-Hellman key exchange on a d-cube

- By Becker and Wille
- d-cube: d dimensional hypercube, which a graph in form of a cube with 2^d nodes
- Each node is connected to d other nodes
- Each node has a unique d bit address, such that the address of 2 nodes connected by an edge along the j^{th} dimension differ only in their j^{th} bit

Fault tolerant Diffie-Hellman key exchange on a d-cube

- Suppose there are $n = 2^d$ participants and each participant is assigned to a node in the hypercube.
- Each participant thus has a unique d-bit address
- Carry out 2 party version of Diffie-Hellman through d rounds
- Suppose node i has address I, then at j^{th} round, I carries out 2 party Diffie-Hellman with node whose address is $I \oplus 2^{j-1}$

- After d rounds, all players will have the same key

Fault tolerant Diffie-Hellman key exchange on a d-cube

- What if n is not a power of 2?
  - "2^d octopus"
  - $2^d < n < 2^{d+1}$
  - The remaining $n - 2^d$ (wards) are distributed among the controllers (at most 1 ward per controller)
  - Controllers carry out 2 party Diffie-Hellman with their wards

- The controllers then engage in d round hypercube protocol
- Controllers distribute the results to their wards

Dealing with faults

- If a node finds its chosen partner to be faulty, then the node should select another non-faulty partner

- Distributed algorithm for finding partners

- Node algorithm for each round

```
procedure do_round(round_number)
    mask = 00...01                              //Initialize mask
    mask = mask ≪ round_number-1                //Left shift mask
    partner = self_address ⊕ mask
    new_mask = mask ≫ 1
    two_party_exchange(partner, new_mask)
end
```

- Recursive algorithm for finding a partner and performing 2 party exchange

```
procedure two_party_exchange(candidate, mask)
    if(mask ≤ 0 )
        // Reached a leaf node
        Attempting to run two-party key exchange
        return success or failure
    endif

    //Else, reached a non-leaf node and try the left side first
    new_mask = mask ≫ 1
    result = two_party_exchange(candidate,new_mask)
    if (result = success) return success
```

```
    //left side failed, try right side
    new_candidate = candidate ⊕ mask
    return two_party_exchange(new_candidate, new_mask)
end
```

Complexity

- 2^d octopus will take $d + 2$ rounds
- But each round can have as many as $n - 1$ sub-rounds (to find a partner)

Comments (from authors)

- Synchronization:
  - Possible to allow each node to proceed independently
  - When node **A** initiates an exchange with **B**, **A** can indicate the round number i
  - If **B** has not reached i, it will reply with a "try later" status message, causing **A** to block this round
  - When **B** replies or when **A** times out and find some other partner, the blockage will be removed

Comments (from authors)

- Leader election:
  - The leader, M_n, has greater say in the final session key than other players
  - It is unclear whether this has any tangible advantage
  - If there is a lack of natural leader or ordering, then there is a need to either elect a leader or find an ordering

Comments (from authors)

- Security issues:
  - How can security be visualized so that a novice can easily configure and use them?
  - How can security policies at various levels be combined?
  - Infrastructure-less, hence difficult to provide security services

Comments (mine)

- Use of weak password P
  - P is weak hence it should not be reuse
  - May be better to use P to establish K and repeat the process again using K
  - Time for intruder to recover P should at least be as long as the conference duration
  - Why use weak password at all?

Comments (mine)

- Not flexible:
  - Everyone must be present in order to carry out the protocol else, the conference may be disrupted for key exchange whenever a late comer shows up
  - It is difficult to remove/or isolate a particular player, may need to re-establish another key and repeated use of P may compromise P

Comments (mine)

- Limited application:
  - May not work in a more complicated scenario, such as one that requires signature
- Identity
  - Since any intruder can inject messages or modify messages, how does one verify that the message is really from the key exchange partner

Comments (mine)

- Is public key techniques a better choice?

Conclusions

- Recap of Diffie-Hellman
- How we can extend Diffie-Hellman from 2 party version to multi-party version
- To increase efficiency and fault tolerance, the "d-cube" or "d-octupus" protocol is introduced