Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Security in Sensor Networks

Written by: Prof. Srdjan Capkun & Others
Presented By : Siddharth Malhotra
Mentor: Roland Flury

Mobile Ad-hoc Networks (MANET)

- Mobile

  Random and perhaps constantly changing

- Ad-hoc

  Not engineered

- Networks

  Elastic data applications which use networks to communicate

MANET Issues

- Routing (IETF's MANET group)
- IP Addressing (IETF's autoconf group)
- Transport Layer (IETF's tsvwg group)
- Power Management
- Security
- Quality of Service (QoS)
- Multicasting/ Broadcasting
- Products

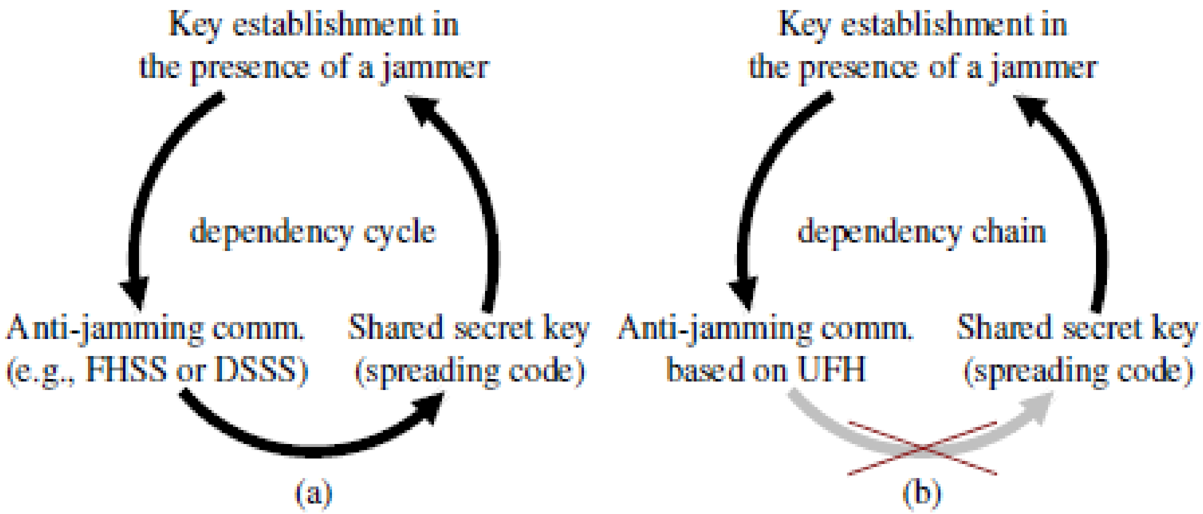
Overview

- Part 1
    - **Jamming-resistant Key Establishment using Uncoordinated Frequency Hopping**
- Part 2
    - **Secure Time Synchronization in Sensor Networks**

Jamming-resistant Key Establishment using Uncoordinated Frequency Hopping

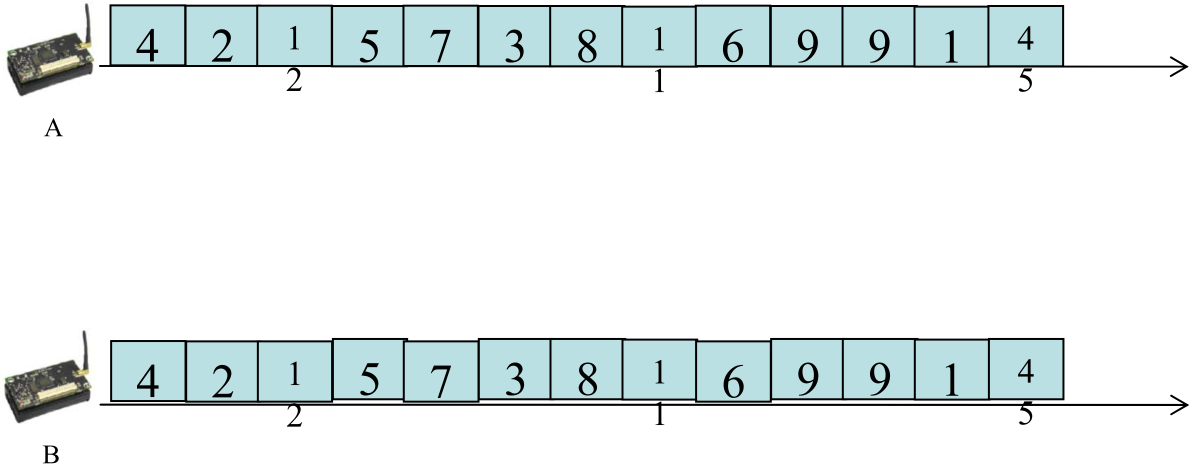
Motivation

- How can two devices that do not share any secret key for communication establish a shared secret key over a wireless radio channel in the presence of a communication jammer?
- Converting the dependency cycle to dependency chain.

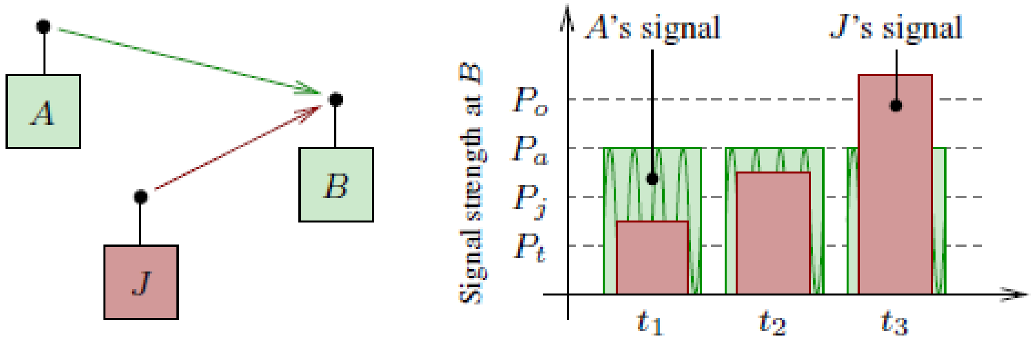


What are we destined to achieve?

Coordinated Frequency Hopping



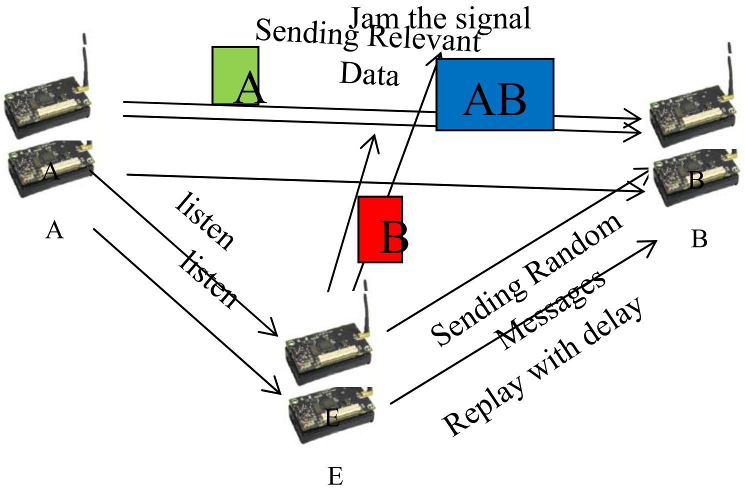
Attacker Model



A – Sender
B – Receiver
J – Attacker

Goal of the Attacker

- Prevent them from exchanging information. Increasing (possibly indefinitely) the time for the message exchange in the most efficient way.

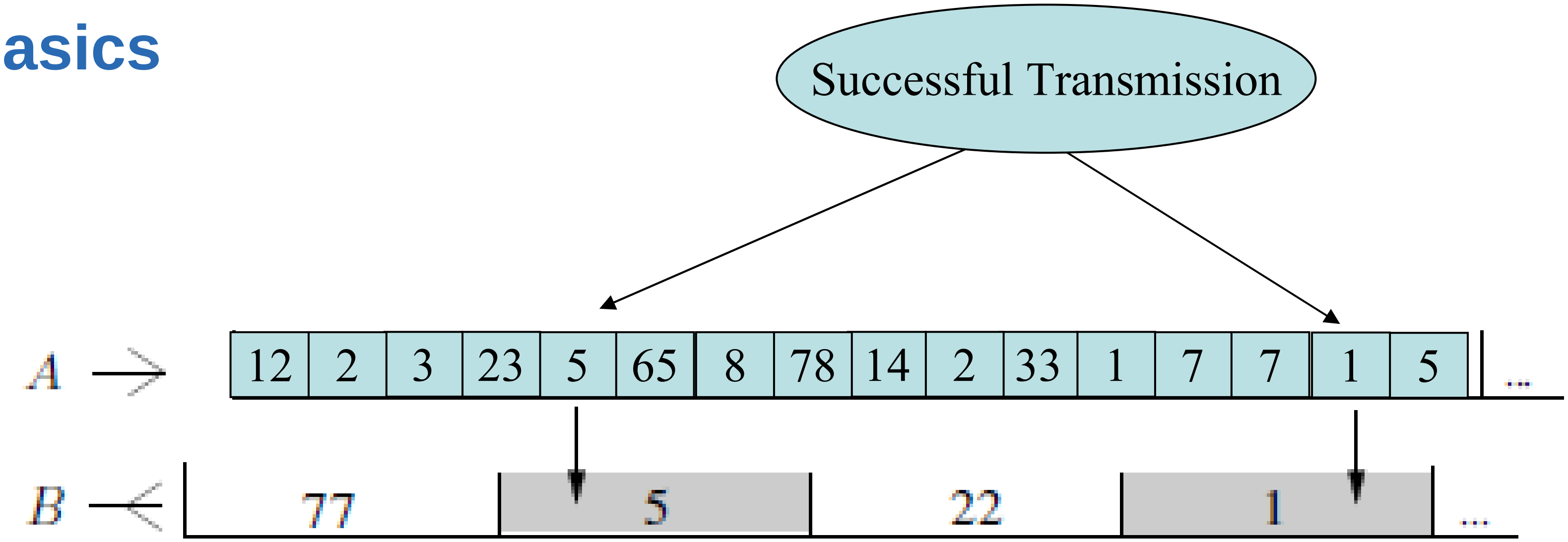


Inserting Messages: Insert messages generated using known (cryptographic) functions and keys as well as by reusing previously overheard messages.

Modifying messages: Modify messages by flipping single bits or by entirely overshadowing original messages.

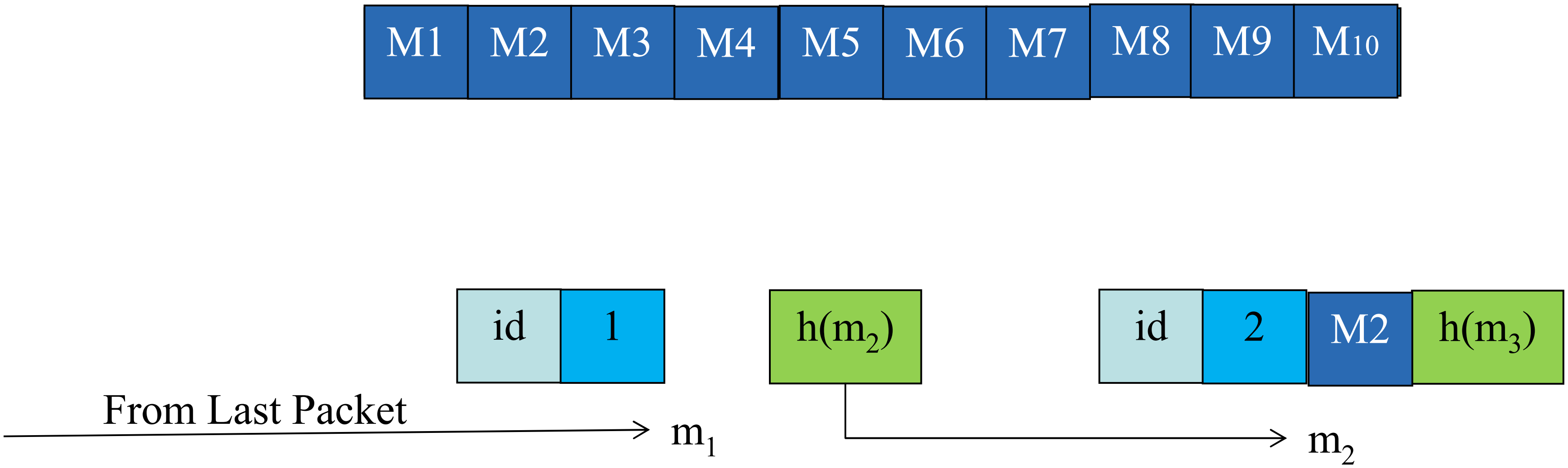
Jamming messages: Jam messages by transmitting signals that cause the original signal to become unreadable by the receiver.

Basics



Sender A is divided into small frequency channels.
Receiver B has larger frequency channels as compared to A

Uncoordinated Frequency Hopping



M1 M2 M3 M4 M5 M6 M7 M8 M9 M10

id | 1 | $h(m_2)$

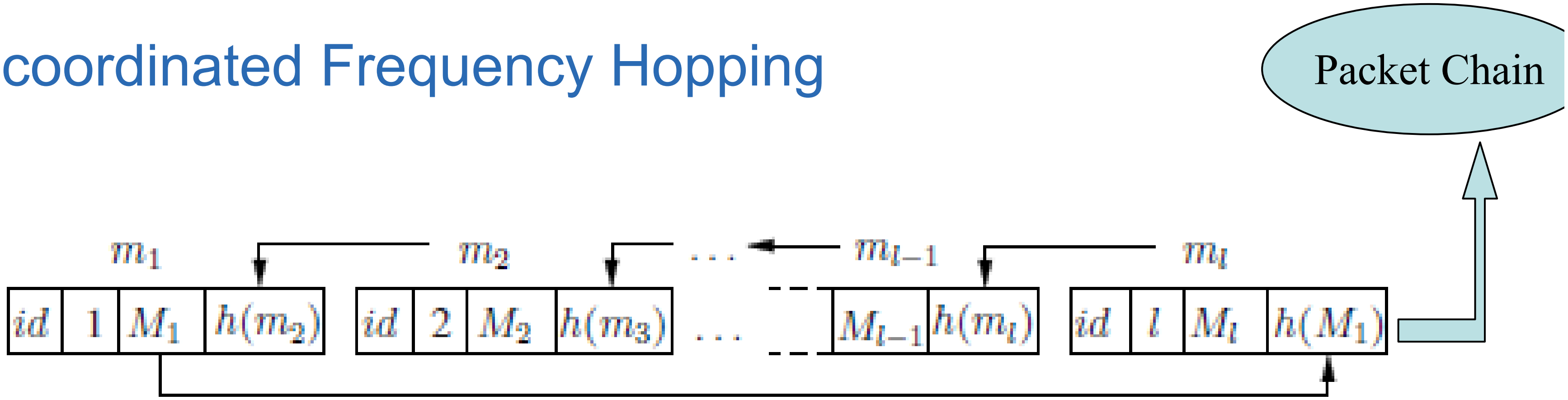
id | 2 | M2 | $h(m_3)$

From Last Packet → m_1

→ m_2

- **Each packet consists of :**
  - **Identifier (id) indicating the message the packet belongs to**
  - **Fragment number (i)**
  - **Message fragment (Mi)**
  - **Hash of the next packet ($h(m_{i+1})$).**

Uncoordinated Frequency Hopping

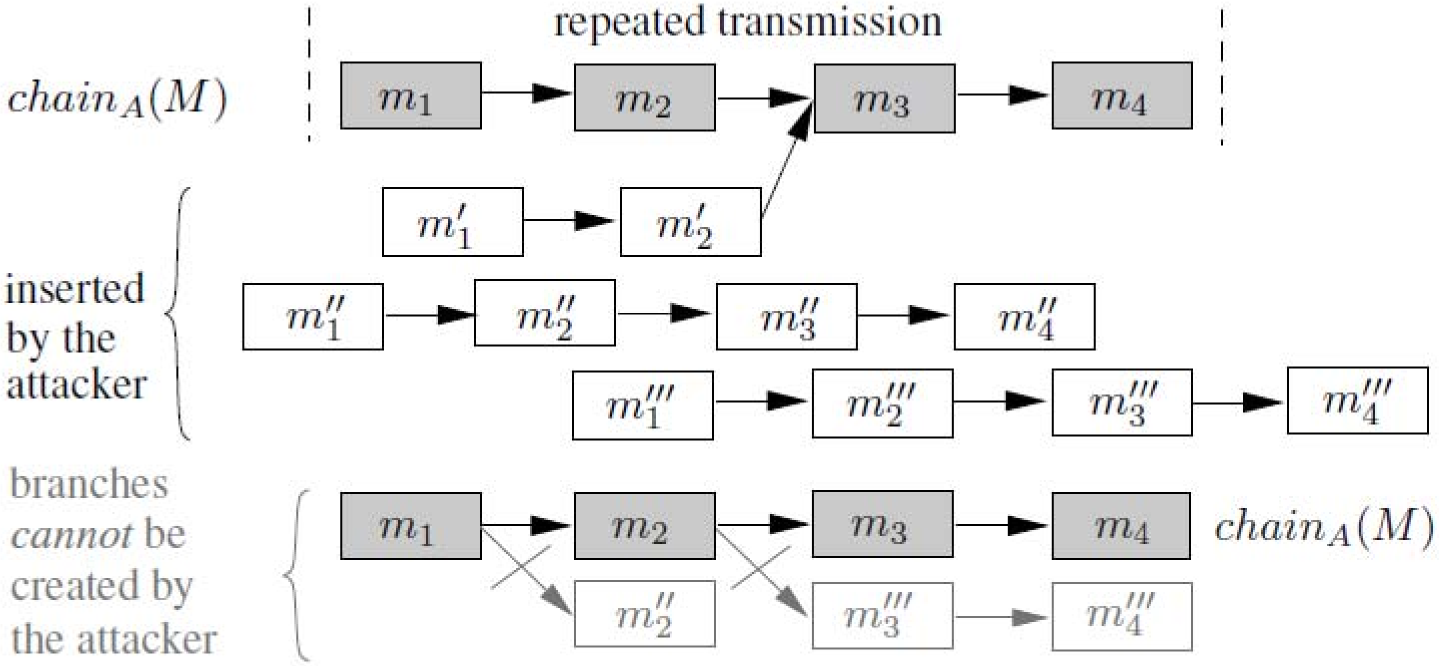


- **Each packet consists:**
  - **Identifier** (id) **indicating the message the packet belongs to**
  - **Fragment number** (i)
  - **Message fragment** (Mi)
  - **Hash of the next packet** (h(mi+1))**.**

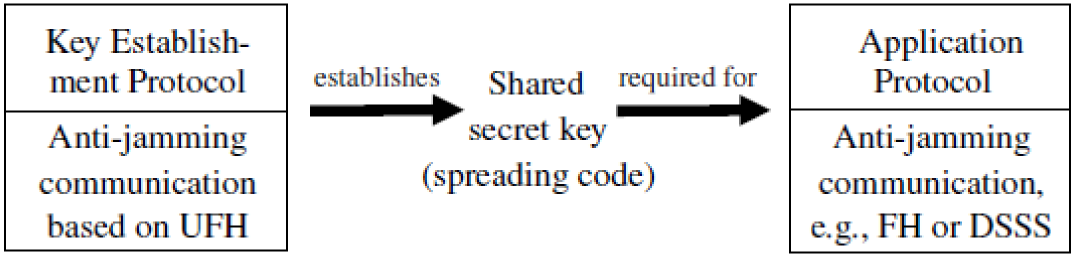
UFH Message Transfer Protocol

- The protocol enables the transfer of messages of arbitrary lengths using UFH.
  - Fragmentation
    - Fragments the message into small packets
    - Hash Function is added
  - Transmission
    - A high number of repetitions (Sends Randomly)
    - Listens the input channels to record all incoming packets
  - Reassembly
    - Packets linked according to Hash Function

Security Analysis of the UFH Message Transfer Protocol



UFH Key Establishment

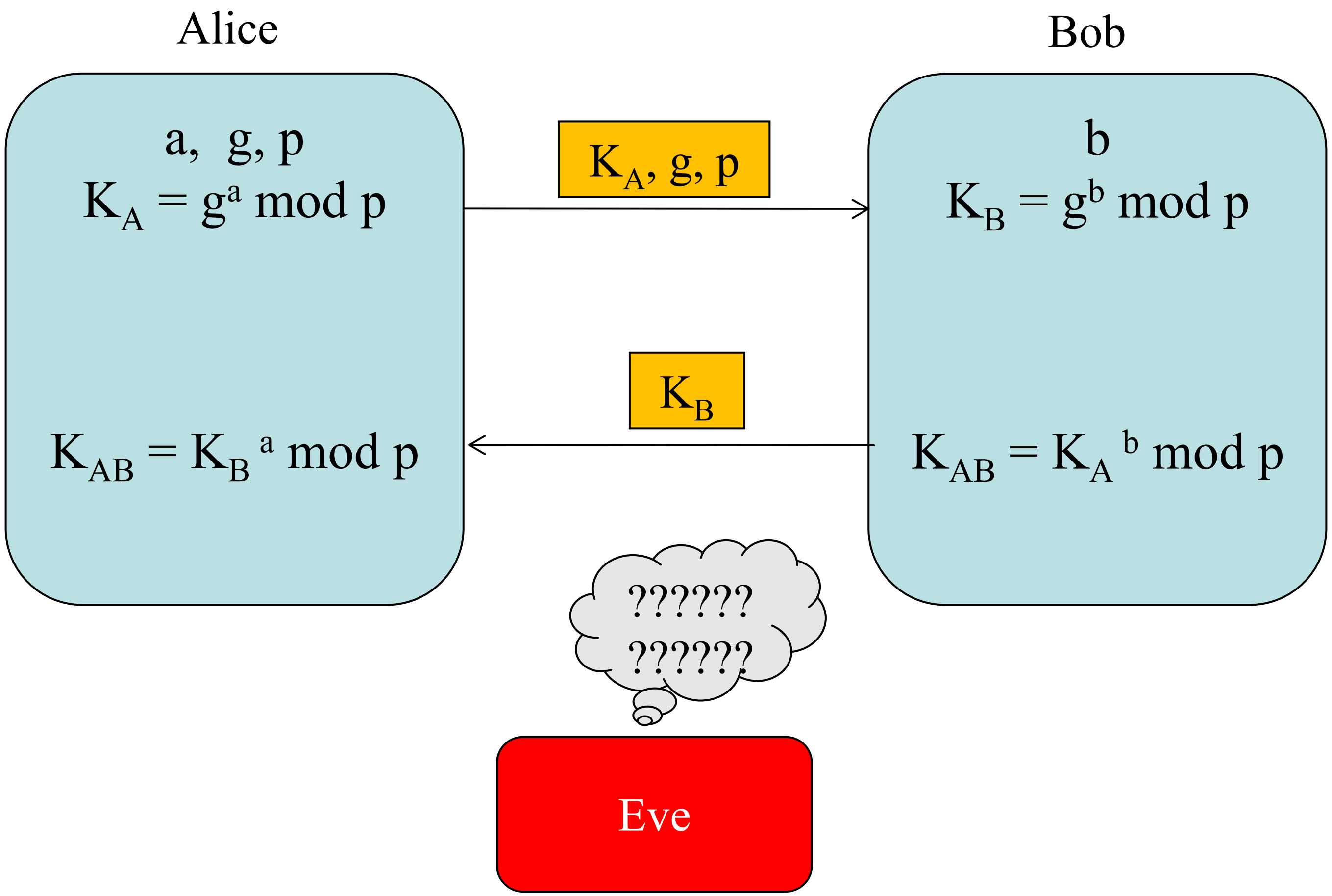


Stage 1
The nodes execute a key establishment protocol and agree on a shared secret key K using UFH.

Stage 2
Each node transforms K into a hopping sequence, subsequently, the nodes communicate using coordinated frequency hopping.

UFH key establishment using authenticated DH protocol

Diffie-Hellman Protocol for Key Exchange



Alice

a, g, p

$K_A = g^a \bmod p$

$K_{AB} = K_B{}^a \bmod p$

K_A, g, p

K_B

Bob

b

$K_B = g^b \bmod p$

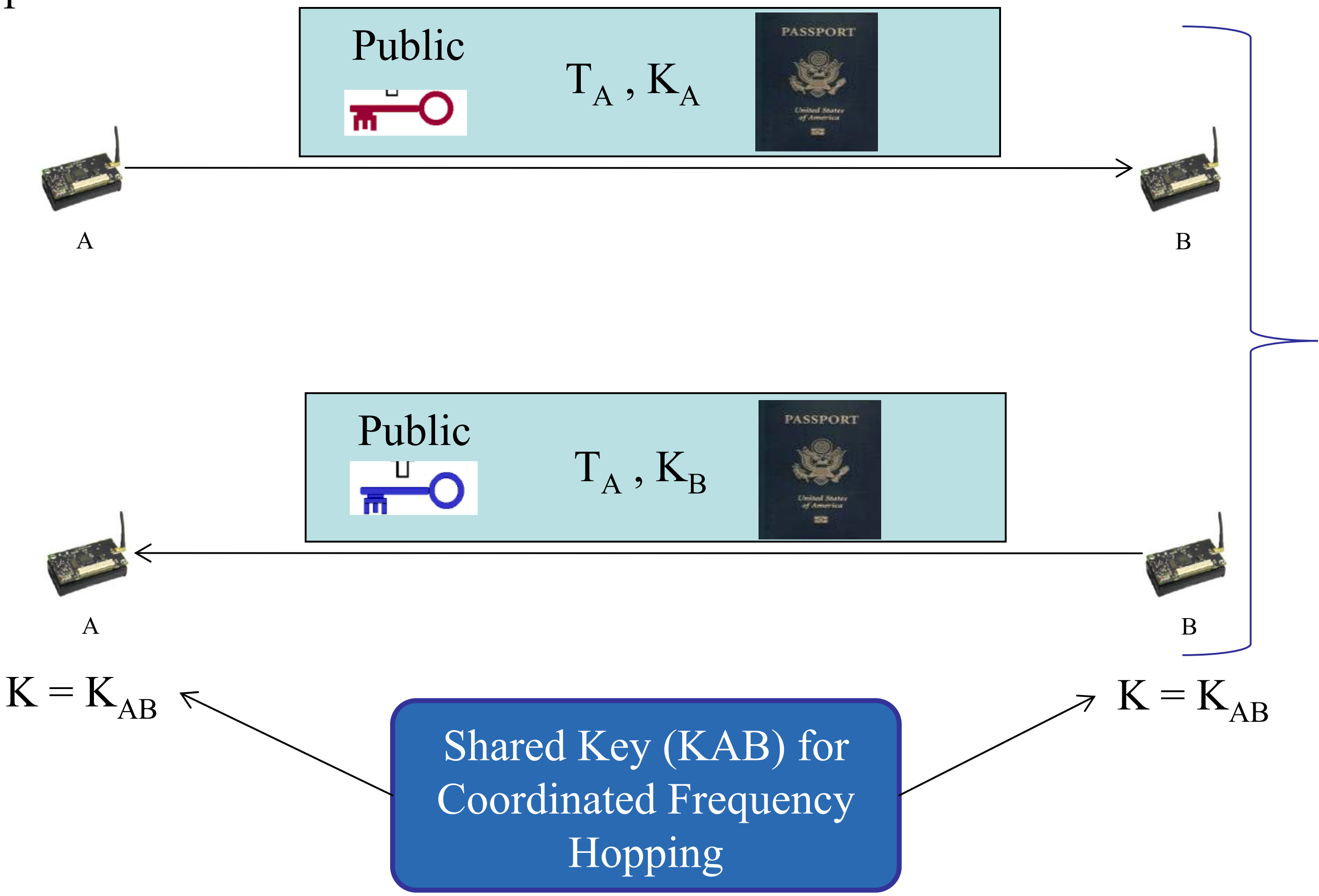
$K_{AB} = K_A{}^b \bmod p$

??????
??????

Eve

UFH key establishment using authenticated DH protocol

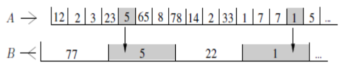
Stage 1



$K = K_{AB}$

Shared Key (KAB) for Coordinated Frequency Hopping

$K = K_{AB}$

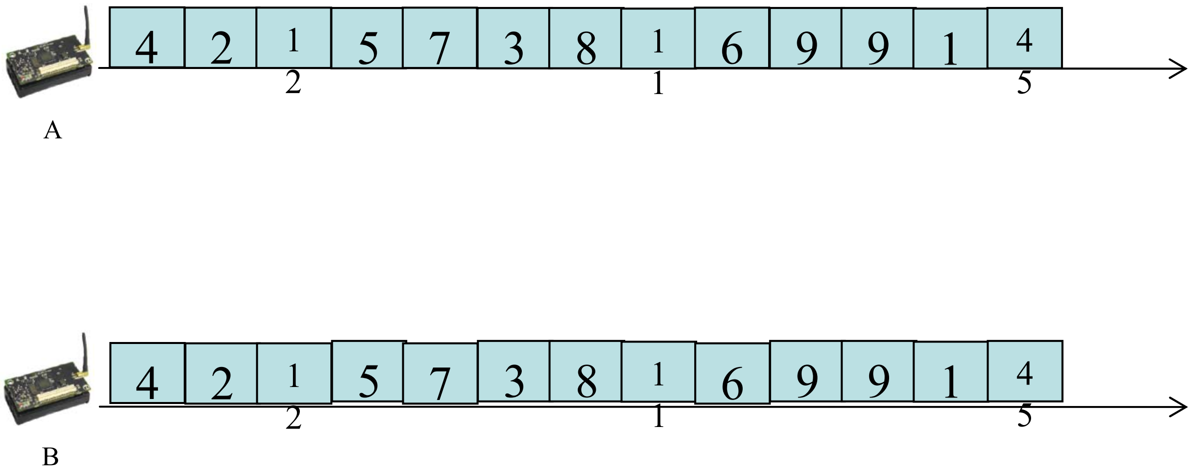


Uncoordinated Frequency Hopping

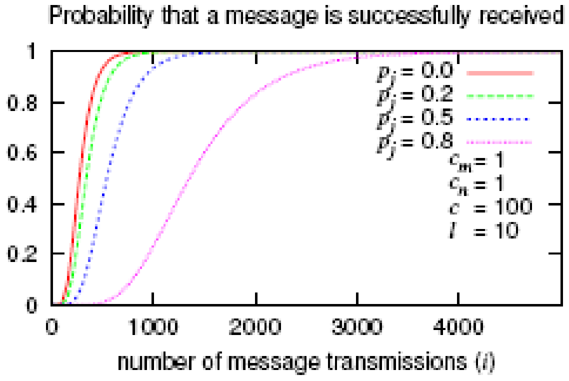
UFH key establishment using authenticated DH protocol

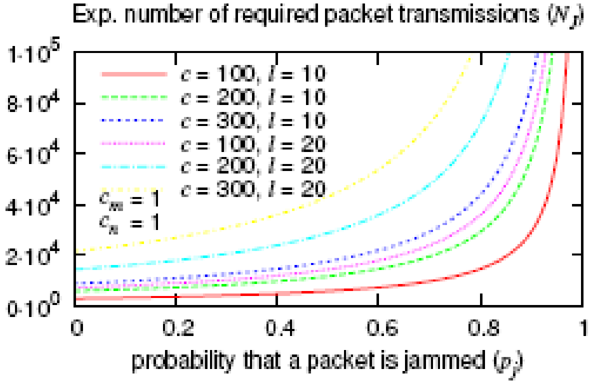
Stage 2

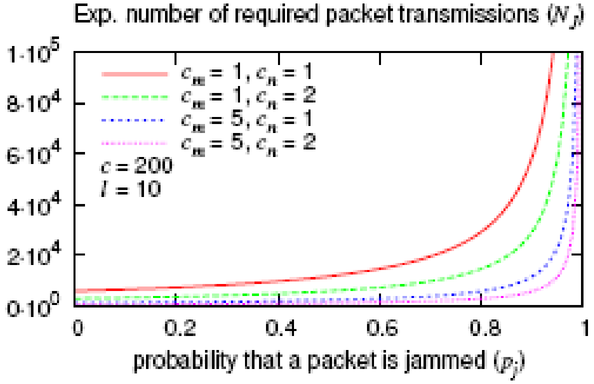
Coordinated Frequency Hopping using the K_{AB}



Results


Probability that a message is successfully received


Exp. number of required packet transmissions (N_j)


Exp. number of required packet transmissions (N_j)

P_j = Probability that a packet is Jammed
C = Total no. of Channels
l = no of packets
N_j = exp. no. of required packets transmissions
C_n = No. of channels for receiving
C_m = No. of Channels for sending

Problems

- How does the receiver know that sender is about the send some data?
- How does the sender come to know that this packet is from this specific chain (not id) like if 5 packet is received at the receiver end and 4,6 not received? How come the receiver comes to know that the packet sent is legitimate?
- Data overflow?

Conclusion

- Coordinated Frequency Hopping has been achieved in presence of a jammer without the use of pre-shared keys for frequency hopping.
- Useful in many things like **time synchronization**

Motivation

- How to provide secure time synchronization for a pair or group of nodes (Connected Directly or Indirectly)?

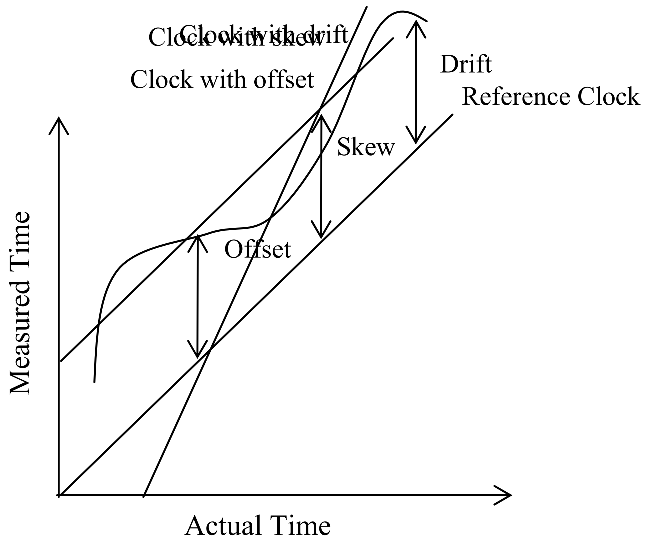
- Synchronizing time is essential for many applications
  - Security
  - Energy Efficiency

Sensor Node Clock

1. **Offset** $(\delta) = C_A(t) - C_B(t)$

2. **Skew** $(\eta) = \dfrac{\partial C_A}{\partial t} - \dfrac{\partial C_B}{\partial t}$

3. **Drift** $(\lambda) = \dfrac{\partial^2 C_A}{\partial t^2} - \dfrac{\partial^2 C_B}{\partial t^2}$



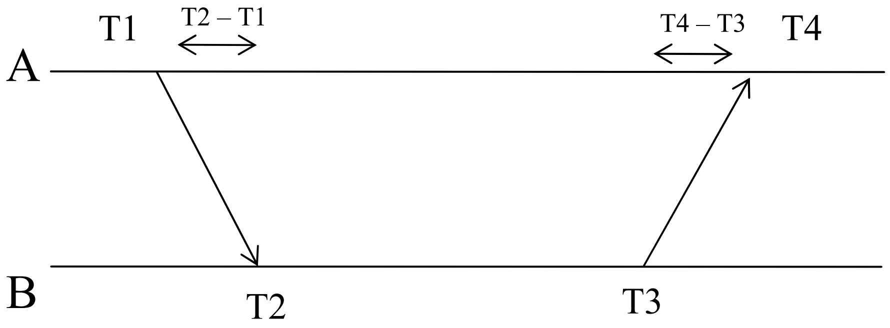
- Three reasons for the nodes to be representing different times in their respective clocks
  - The nodes might have been started at different times,
  - The quartz crystals at each of these nodes might be running at slightly different frequencies,
  - Errors due to aging or ambient conditions such as temperature

Attacker Model

- Two types of attacker models:
  - **External Attacker:** None of the nodes inside the network have been compromised
  - **Internal Attacker:** One or more nodes have been compromised, its secret key is known to the attacker

Sender-Receiver Synchronization

- A handshake protocol between a pair of nodes.



Sender synchronizes to the receiver clock

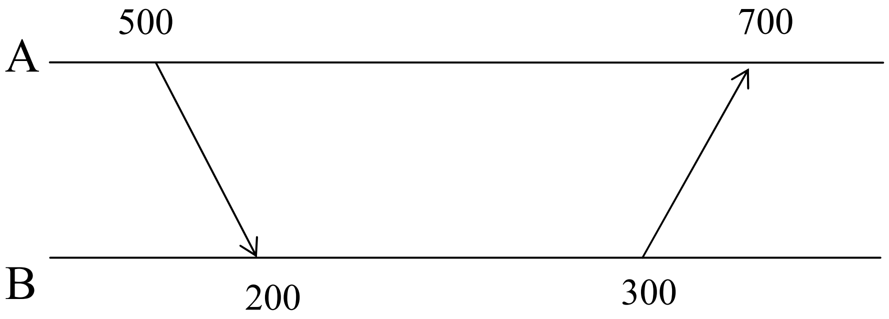
$$Step1 \rightarrow T2 = T1 + d + \delta$$

$$Step2 \rightarrow T4 = T3 - d + \delta$$

$$\underbrace{\delta = \frac{(T2 - T1) - (T4 - T3)}{2}}_{\text{Clock Offset}}; \quad \underbrace{d = \frac{(T2 - T1) + (T4 - T3)}{2}}_{\text{Delay}}$$

Sender-Receiver Synchronization

- Example



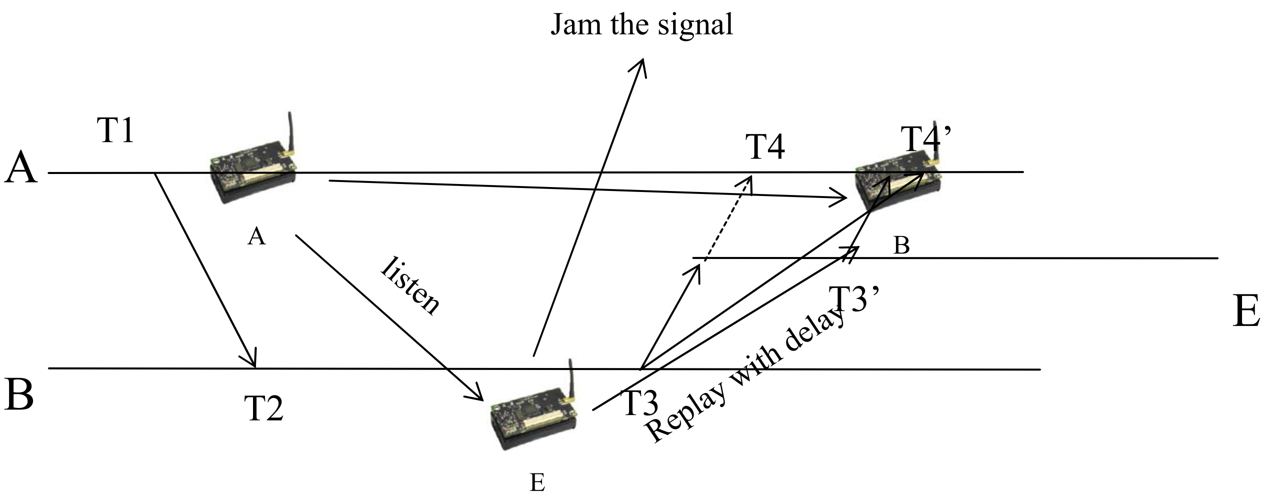
$$\delta = ((200 - 500) - (700 - 300)) / 2 = -350$$
$$d = ((200 - 500) + (700 - 300))/2 = 50$$

Sender (A) updates its clock by δ (Here -350)

External Attacker

- Three types in which attacker can harm the time synchronization:
  - Modifying the values of T2 and T3
  - Message forging and replay
  - Pulse delay Attack

Pulse Delay Attack



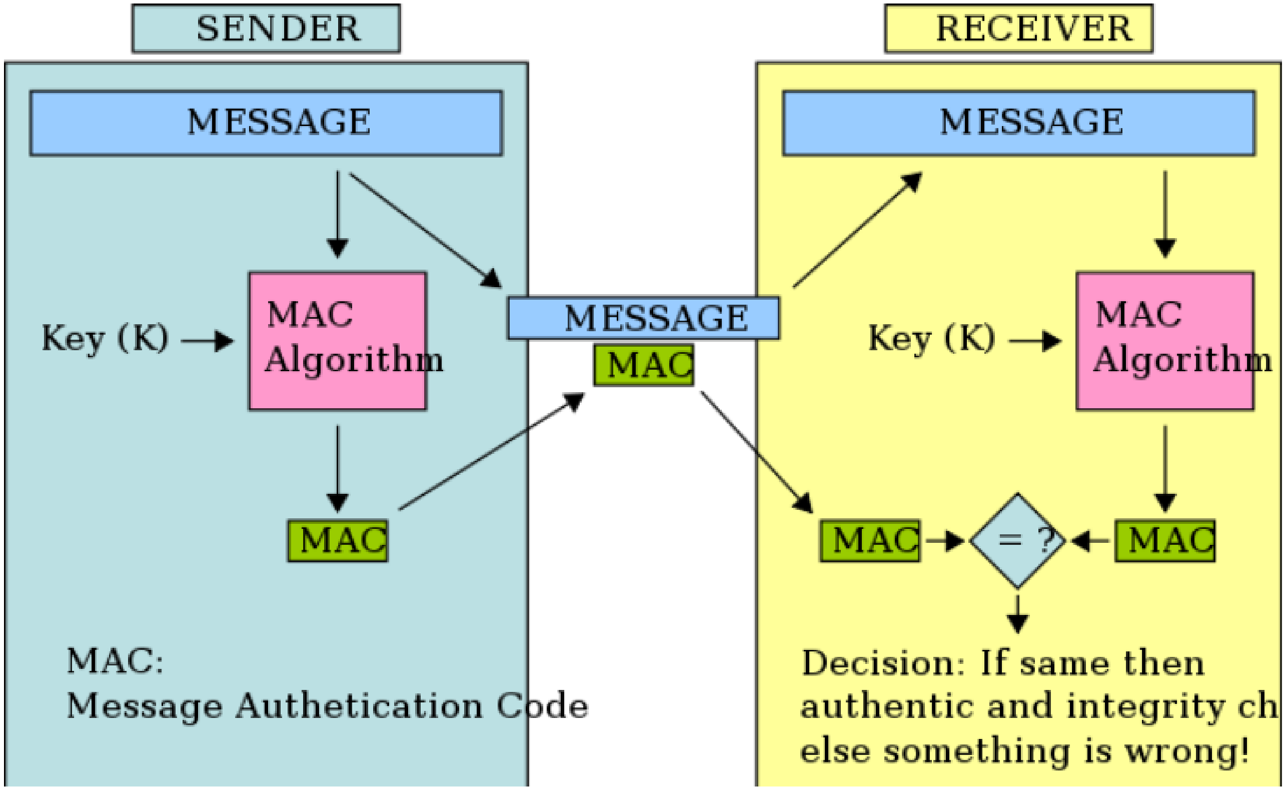
$$Step1 \rightarrow T2 = T1 + d + \delta$$
$$Step2 \rightarrow T4' = T3 - d + \delta$$

$$\delta = ((T2 - T1) - (T4' - T3)) /2$$
$$d = ((T2 - T1) + (T4' - T3)) /2$$

SECURE TIME SYNCHRONIZATION

- Three types of synchronization have been discussed:
  - Secure Pairwise Synchronization
  - Secure Group Synchronization
  - Secure Pairwise Multi-hop Synchronization

Message Authentication Code



MAC:
Message Authetication Code

Decision: If same then
authentic and integrity ch
else something is wrong!

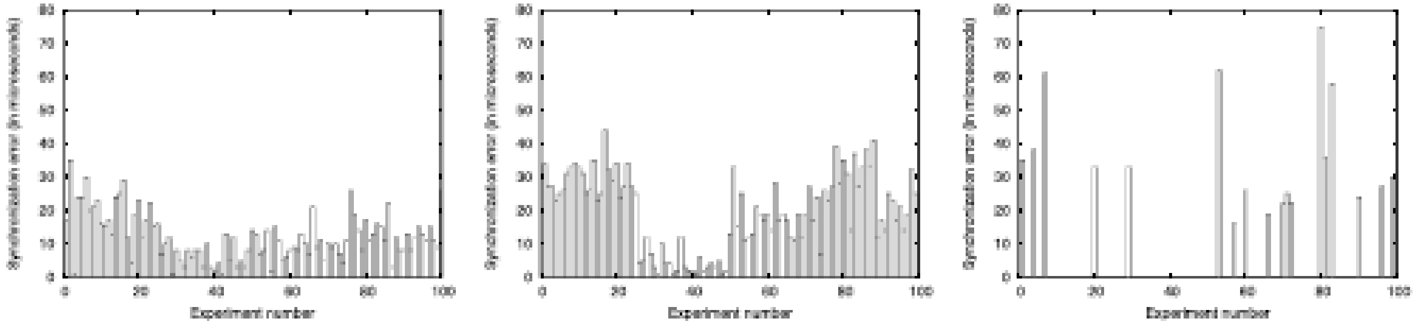
Secure Pairwise Synchronization (SPS)

- Message integrity and authenticity are ensured through the use of Message Authentication Codes (MAC) and a key K_{ab} shared between *A* and *B*.

P1	*sync*
P2	*T2, T3,ack*

If d<= d* then clock offset (δ)
else abort

Results



(a) Nonmalicious setting (b) Pulse delay attack of 10 μs (c) Pulse delay attack of 25 μs

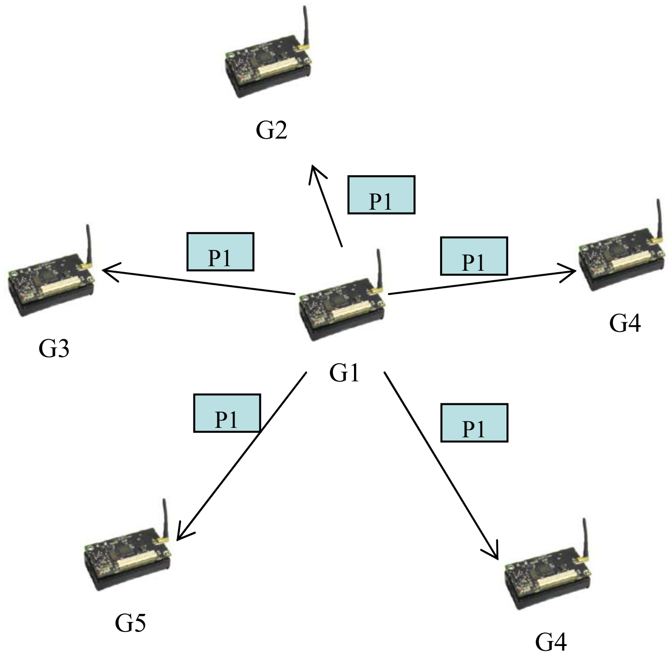
Experiment	Average error	Maximum error	Minimum error	Attack detection probability
Non Malicious	12.05 μs	*35 μs*	*1 μs*	*NA*
∆ = 10 μs	*19.44 μs*	*44 μs*	*1 μs*	1 %
∆ = 25 μs	35.67 μs	*75 μs*	*16 μs*	*82%*

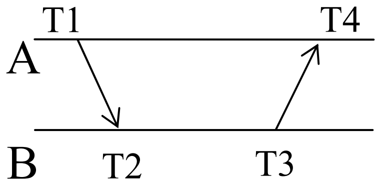
GROUP SYNCHRONIZATION

- 2 Types:
  - **Lightweight Secure Group Synchronization**
    - Resilient to External attacks only
  - **Secure Group Synchronization**
    - Resilient to External attacks as well as internal attacks (Attacks from compromised nodes)

Lightweight Secure Group Synchronization (L-SGS)

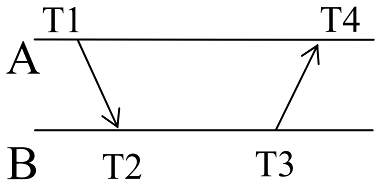
Step 1



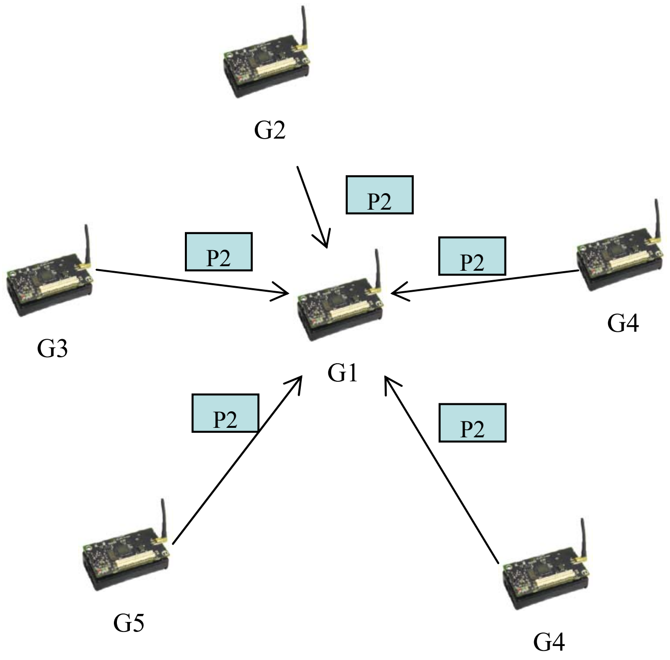


P1 *sync*

Lightweight Secure Group Synchronization (L-SGS)



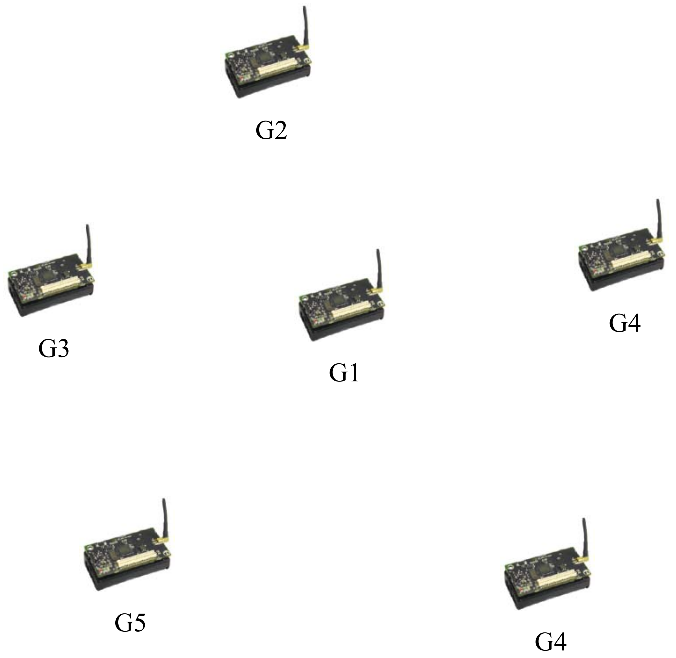
Step 2

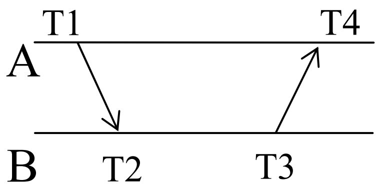


P2 T2, T3 (Every node which receives sync from G1)

Lightweight Secure Group Synchronization (L-SGS)

Step 3



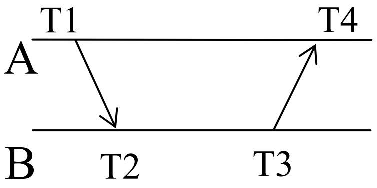


Pr

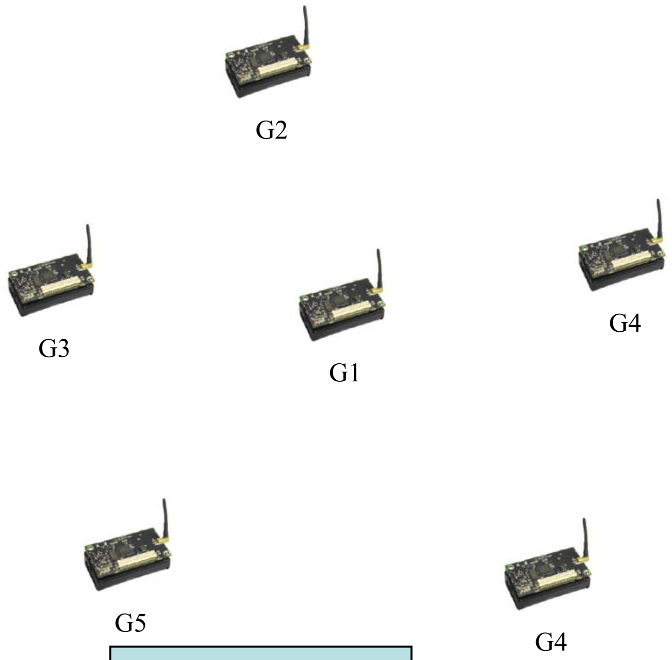
compute d *for every node* d_{ij}

$\mathit{if}\ d_{ij} \le d* \ \mathit{then}\ (\mathit{Clock\ offset})_{ij}\ \mathit{else\ abort}$

Lightweight Secure Group Synchronization (L-SGS)



Step 4



Estimation of the local clock of G_i

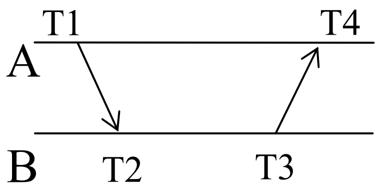
C_{ij}

Local Clock

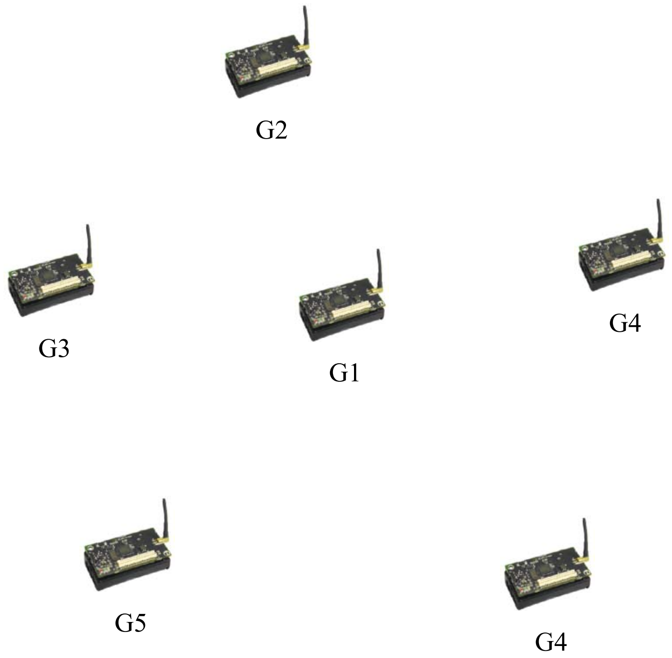
$C_i + (\mathit{Clock\ offset})_{ij}$

Pairwise offset

Lightweight Secure Group Synchronization (L-SGS)



Step 5



Global Clock → C_g^i

Median $(C_i\ , [C_{ij}]^{j=1.....N;j<>n})$

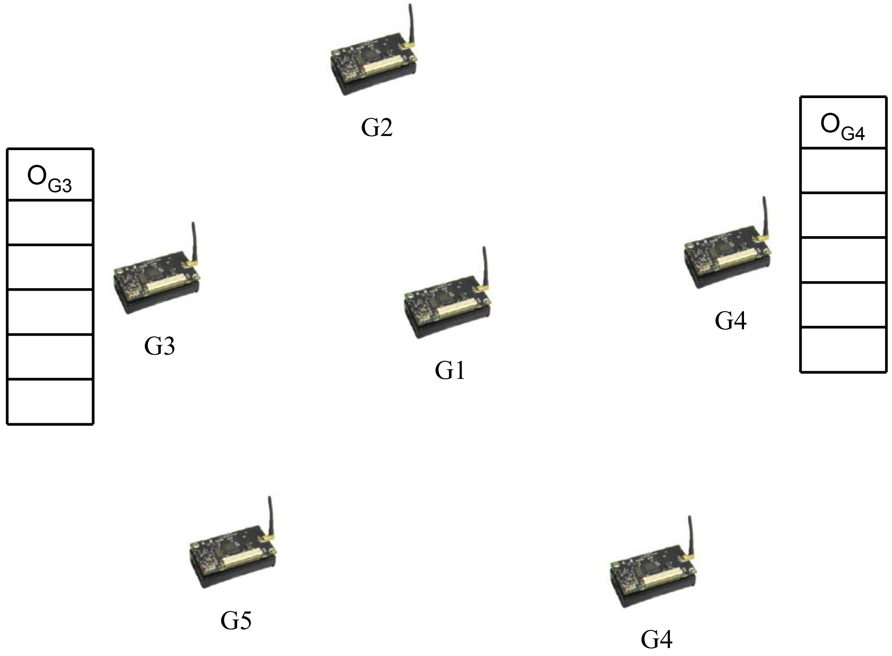
Secure Group Synchronization

- Secure Group Synchronization is resilient to both external and internal attacks
- We will make the use of tables (O_i for node G_i)

Secure Group Synchronization

1^{st} two steps are the same as (L-SGS)

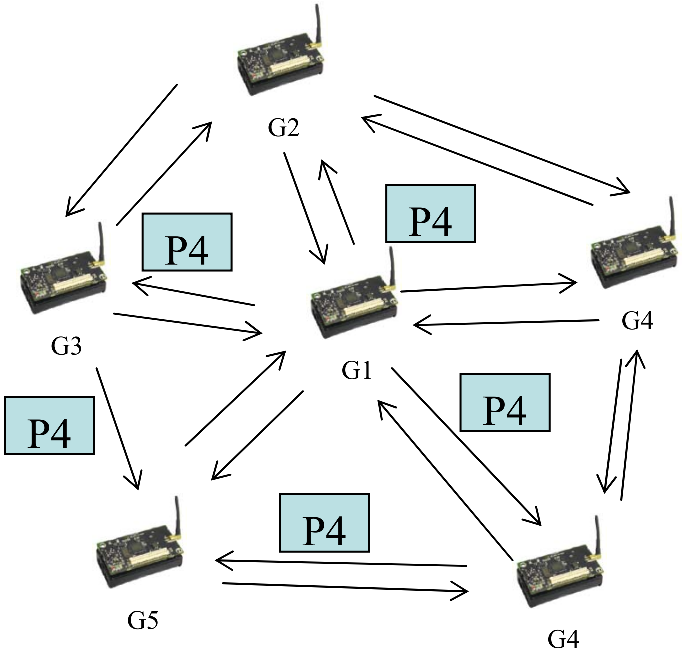
Step 3



$$O_i = O_i \cup \delta_{ij}$$

Secure Group Synchronization

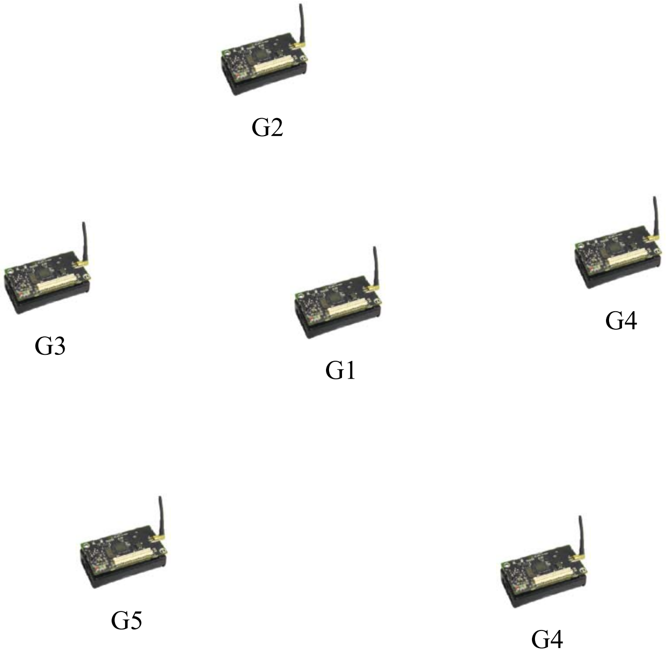
Step 4



P4 O_i

Secure Group Synchronization

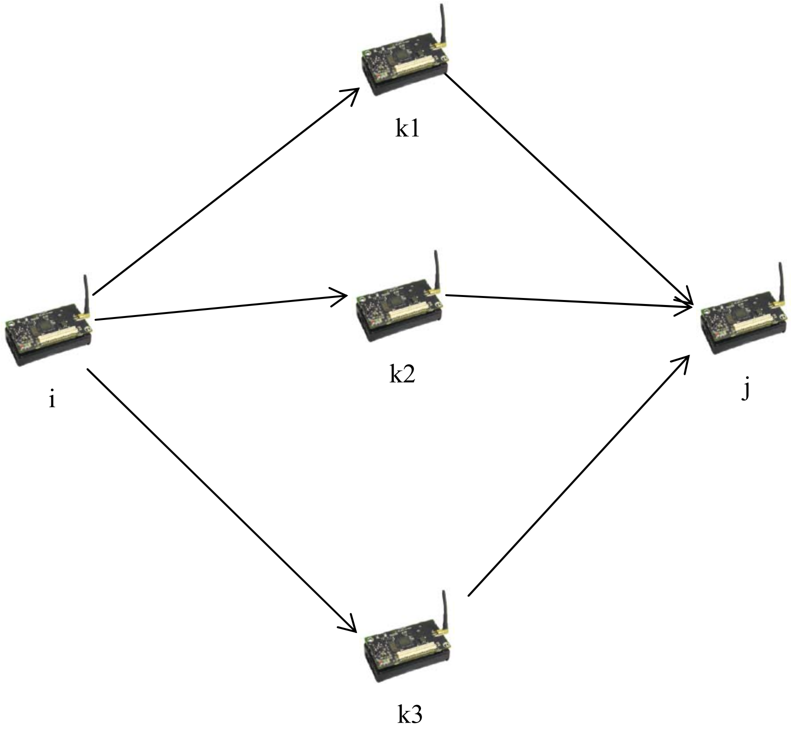
Step 5



Run the SOM($\lfloor (N-1)/3 \rfloor$) algorithm to compute C_{ij}

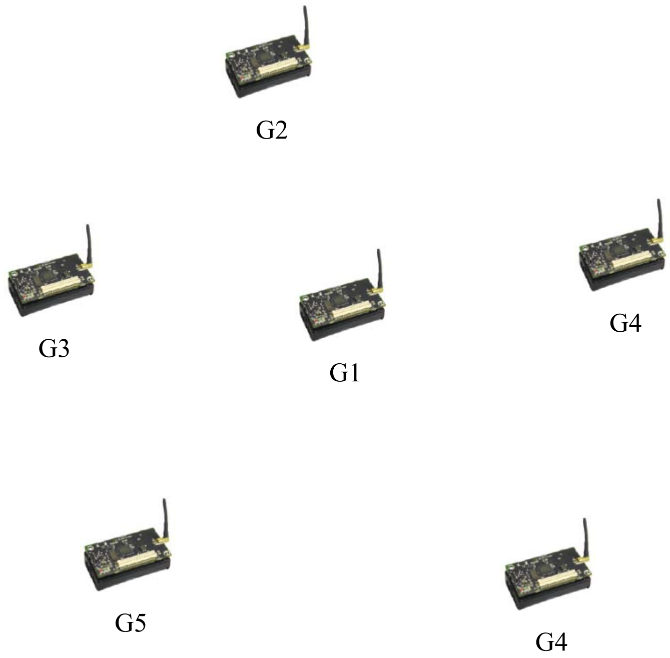
SOM

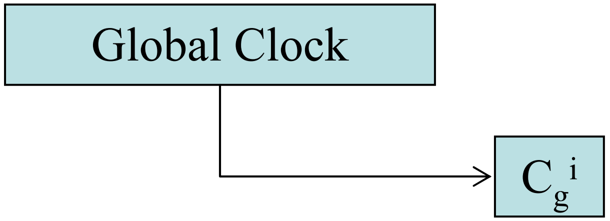
- Recursive Algorithm
- Each node uses other group members to compute C_{ij}



Secure Group Synchronization

Step 5



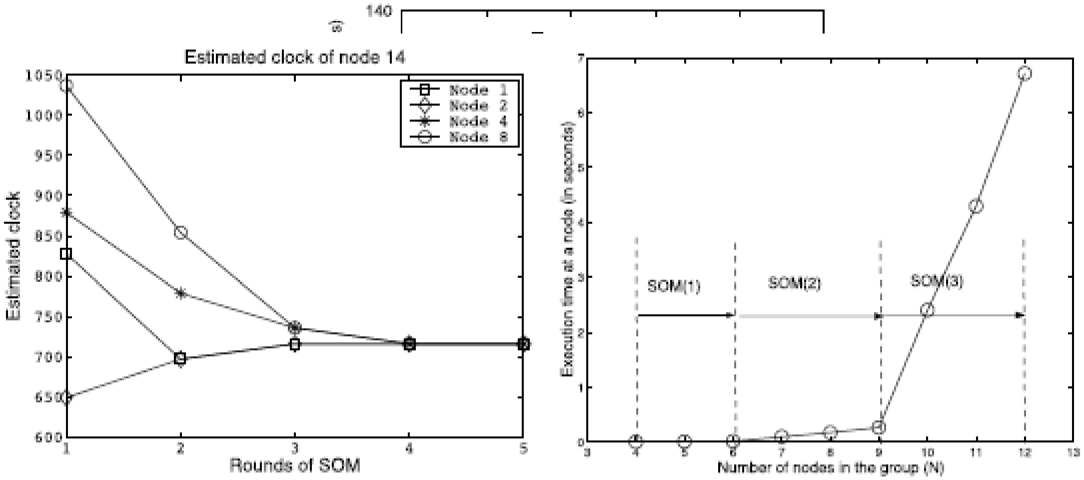


Global Clock

C_g^i

Median $(C_i \; , [C_{ij}]^{j=1.....N;j<>n})$

Results



(a) Performance of SOM in simulations.

(b) Time complexity of SGS on motes.

Maximum error	130 μs
Minimum error	1 μs

N = No. of nodes (14)
C = Compromised nodes
C = (11,12,13,14)

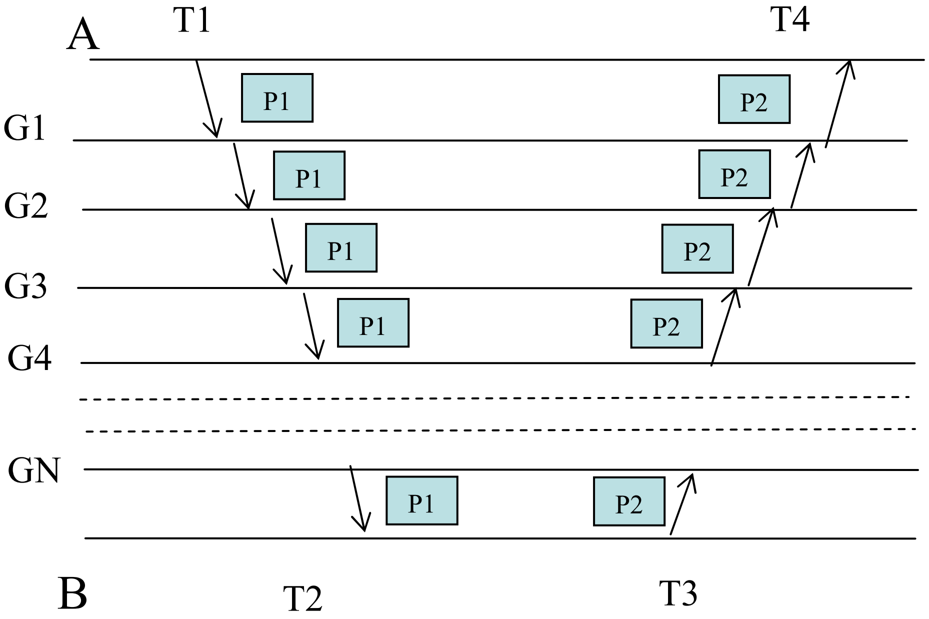
N = No. of nodes
T = Time to finish SGS
SOM(i) = No. of Compromised nodes

Secure Pairwise Multi-hop Synchronization

- Enable distant nodes, multiple hops away from each other, to establish pairwise clock offsets

- Categorized into two types:
  - Secure Simple Multi-hop Synchronization
  - Secure Transitive Multi-hop Synchronization

Secure Simple Multi-hop Synchronization



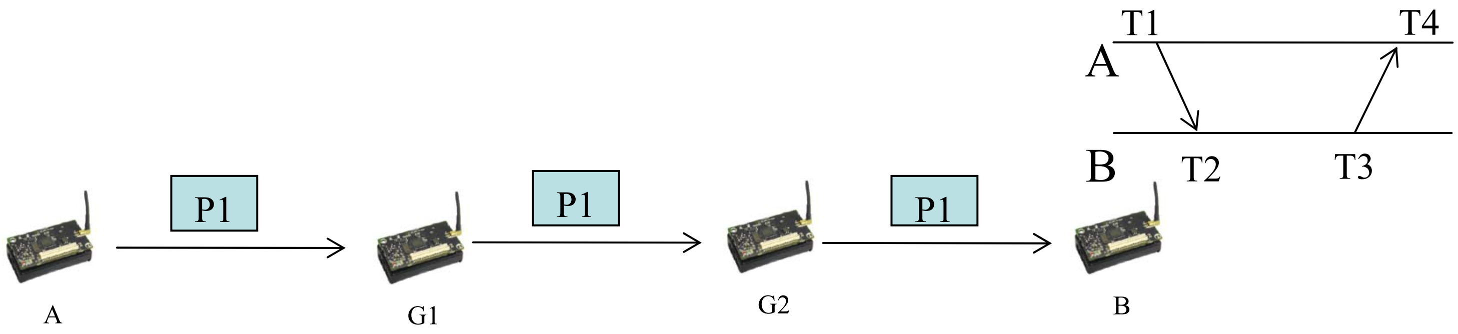
P1 *sync*

P2 *T2, T3,ack*

If d<= dM* then $\delta = ((T2-T1)-(T4-T3))/2$
else abort

Secure Transitive Multi-hop Synchronization

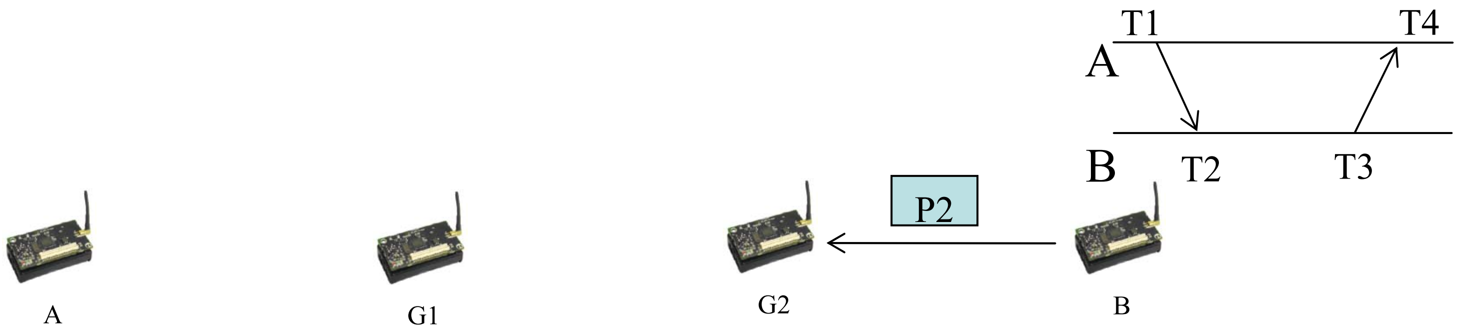
Step 1



P1 *sync*

Secure Transitive Multi-hop Synchronization

Step 2

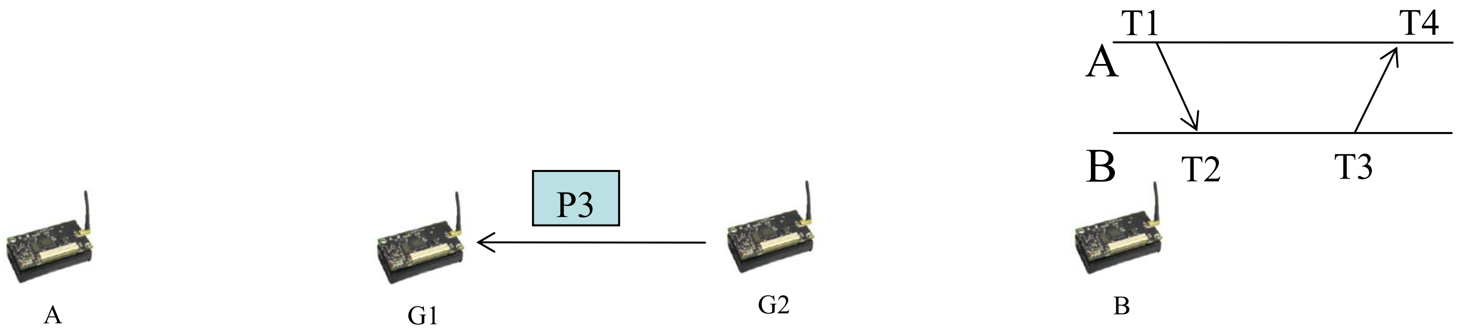


P2 *T2 (B) , T3(B),ack*

→ G2 is synchronized to B

Secure Transitive Multi-hop Synchronization (STM)

Step 3

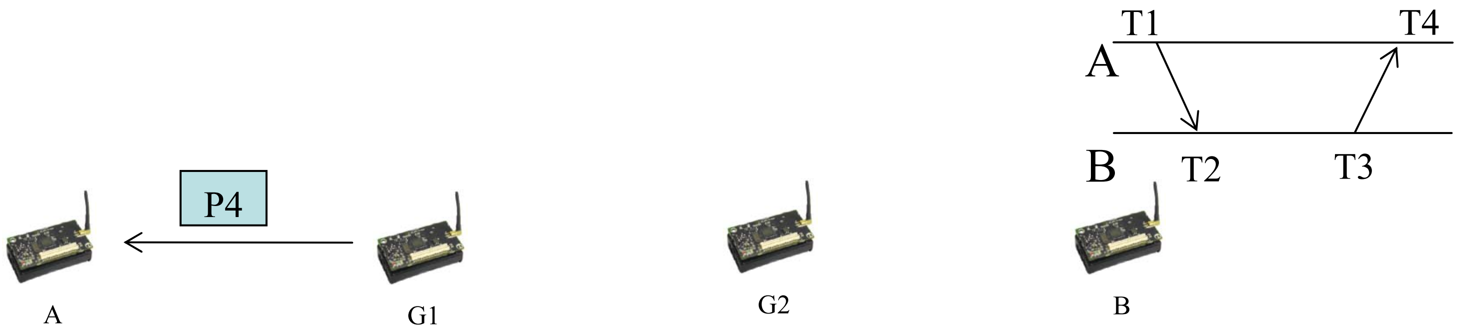


P3 *T2 (G2) , T3(G2),ack*

→ G1 is synchronized to G2

Secure Transitive Multi-hop Synchronization

Step 4



P4 *T2 (G1) , T3(G1),ack*

→ A is synchronized to G1

Conclusion

- SPS achieves the same synchronization precision on a pair of motes as the insecure time synchronization protocols. Even under a pulse-delay attack, SPS can keep the nodes in sync within 40μ*s.*
- SGS is able to synchronize a group of four motes within50μ*s,* even with 1 node used for internal attack
- SPS extended to STM.