Wireless Network Security

Spring 2016

Patrick Tague

Class #8 – Broadcast Security & Key Mgmt

Note on HW#2

- With a fresh install of OMNET++ 4.6, it grabs INET 3.2, but the sample code we gave you only works for INET < 2.99
  - You'll need to downgrade your INET install to use the sample code

Class #8

- Broadcast authentication
- Group key management

Broadcast Communication

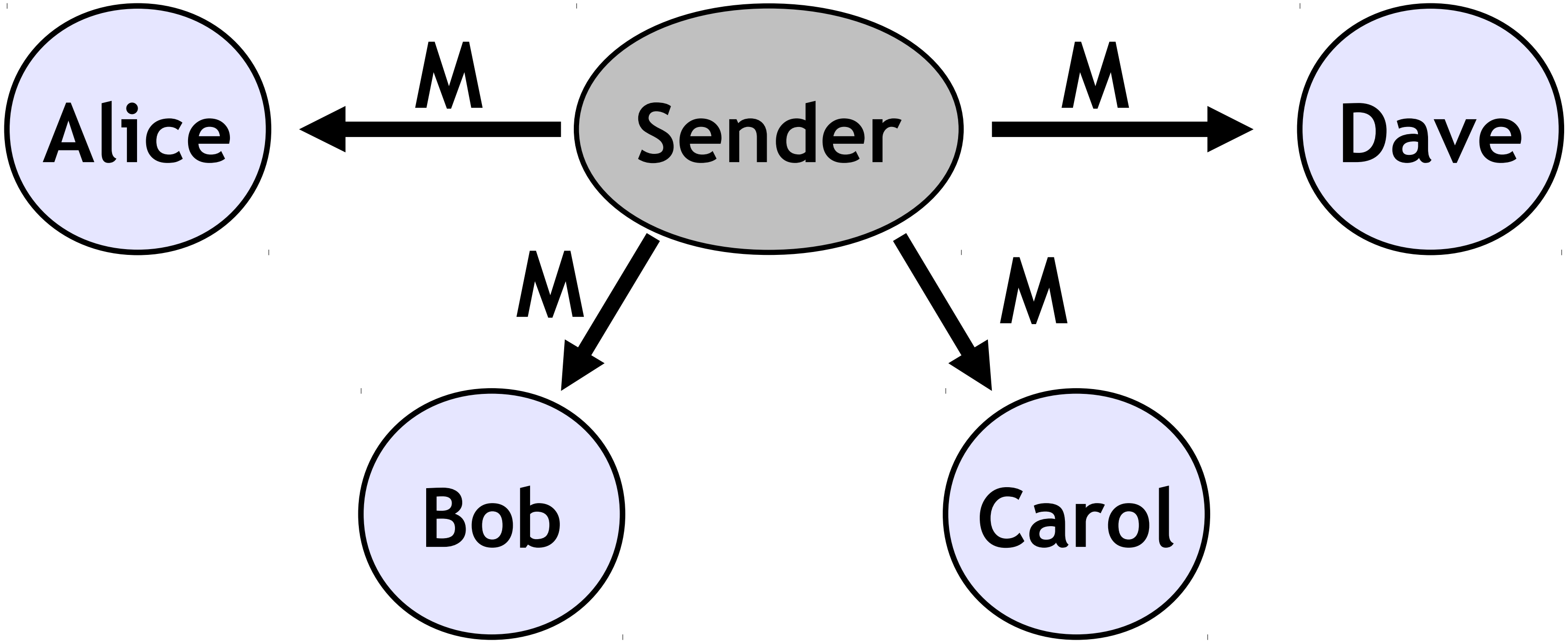
- Wireless networks can leverage the “broadcast advantage” property to send a message to multiple recipients simultaneously
  - In a “star” (like a WiFi network), O(1) transmissions cover all N recipients
  - In general, O(N/d) transmissions cover N recipients with density d, using relaying

Broadcast Security

- To leverage "broadcast advantage"
  - All recipients need to be able to authenticate the transmitter / message from the single transmission
  - All recipients need to be able to decrypt the message from the single transmission
  - Also, the authentication, en/decryption, and key management mechanisms need to be efficient

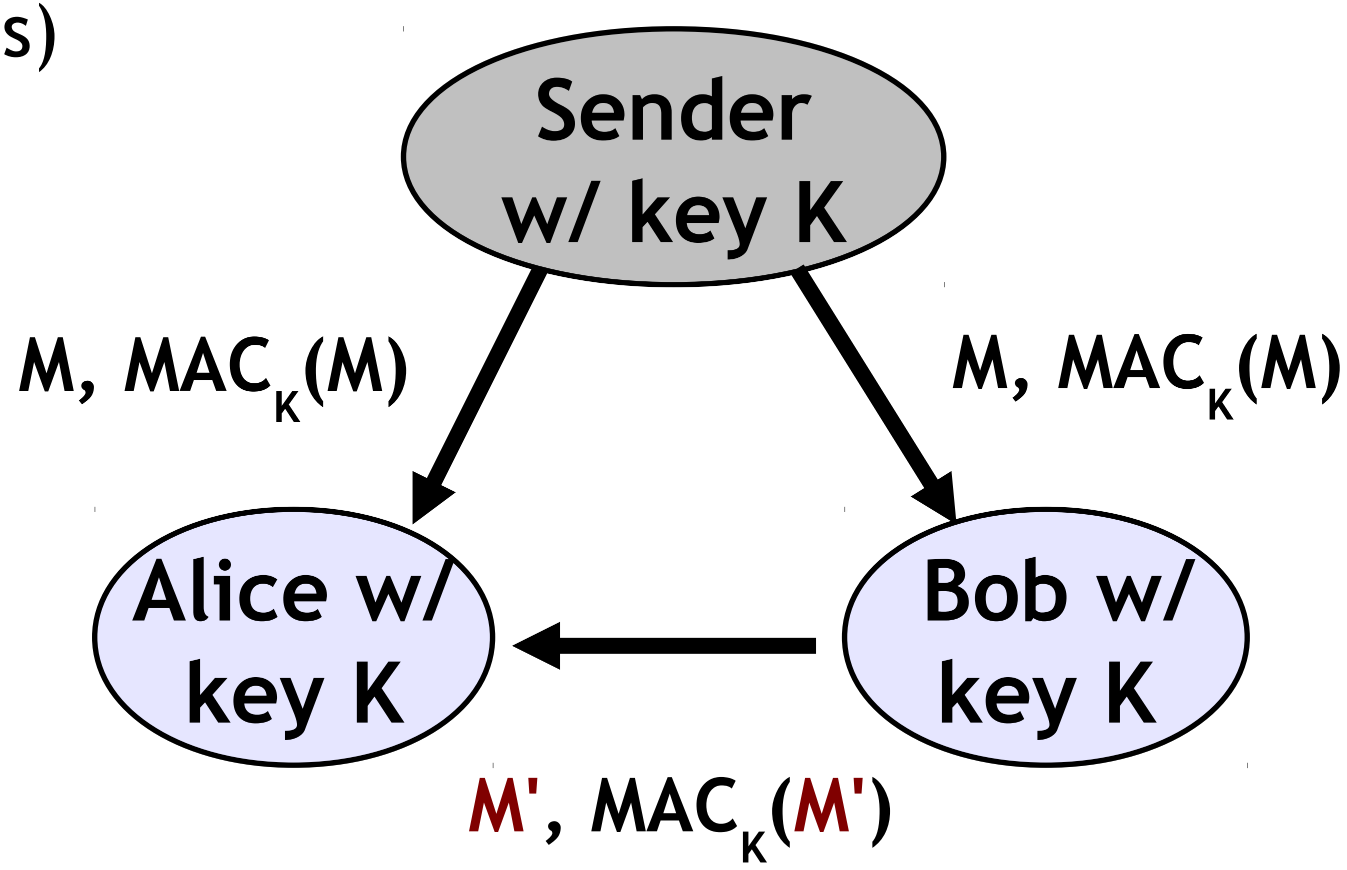
Broadcast Authentication

- Sender wants to broadcast a single message in a wireless network
- To protect against packet injection and other threats, need to **verify the data origin**



Broadcast Auth Mechanisms

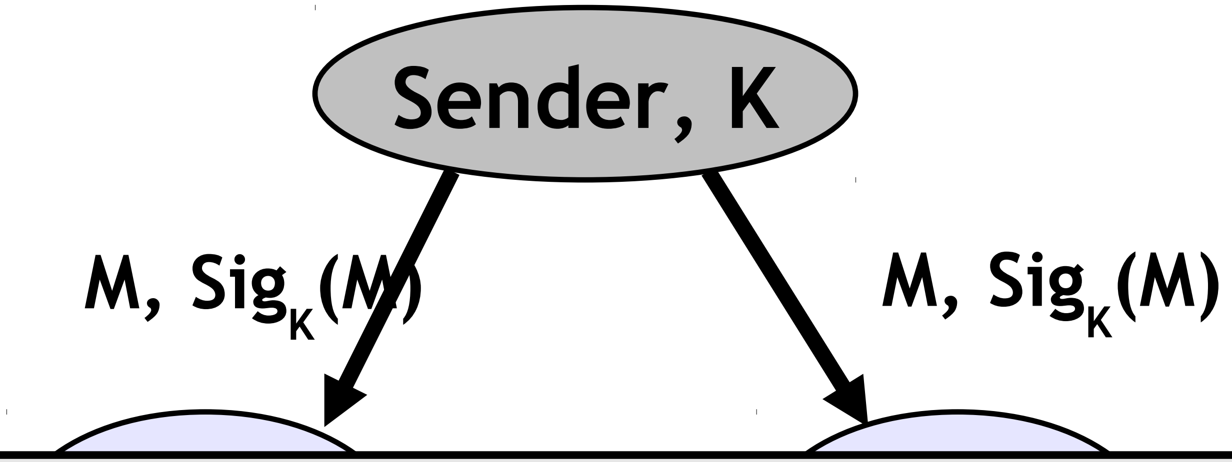
1. Symmetric key crypto and message auth codes (MACs)



Some form of asymmetry is required

Broadcast Auth Mechanisms

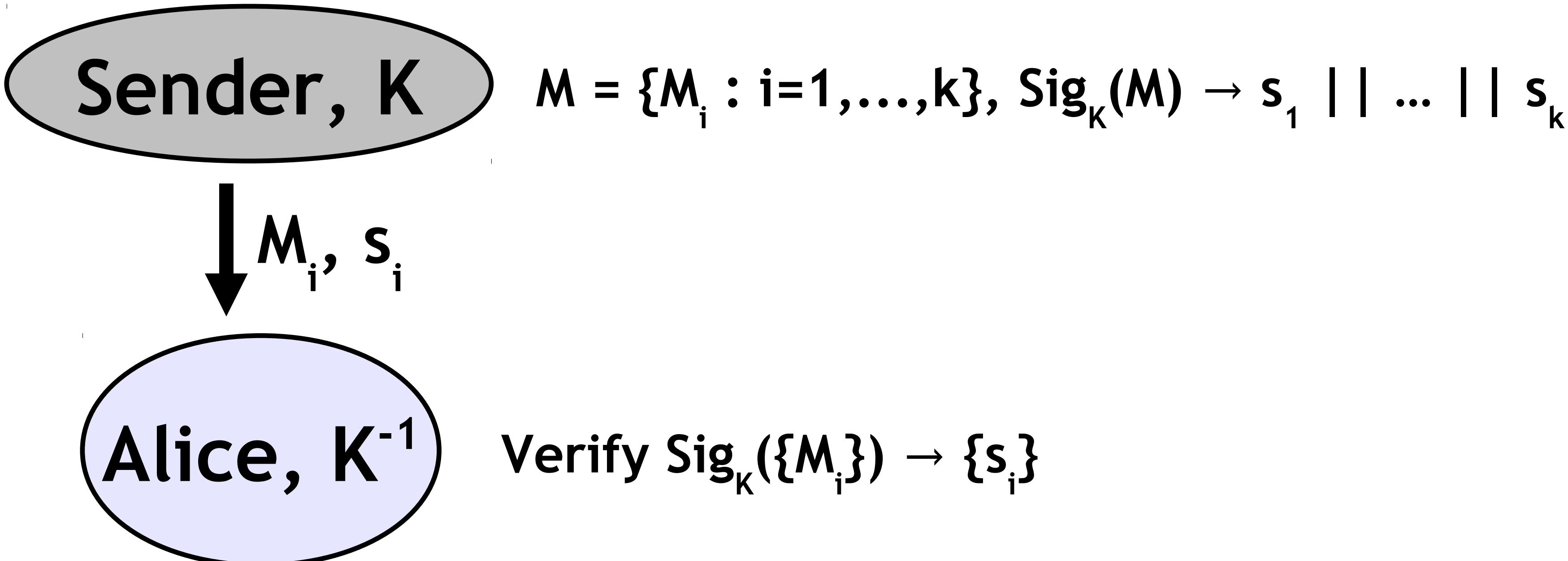
2. Public-key signatures
   - Sender uses a private key to sign the message, all recipients verify with the corresponding public key



- Ex: RSA 1024 bit signatures
  - High generation cost (~10 milliseconds)
  - High verification cost (~1 millisecond)
  - High communication cost (128 bytes/packet)
- Even more costly for low-end processors

Broadcast Auth Mechanisms

3. Packet-block signatures
   - Sign a collection of packets, partition signature over packet block → disperse the cost of signing over larger chunks of data



Sender, K

$M = \{M_i : i=1,\ldots,k\}$, $Sig_K(M) \rightarrow s_1 \,||\, \ldots \,||\, s_k$

M_i, s_i

Alice, K^{-1}

Verify $Sig_K(\{M_i\}) \rightarrow \{s_i\}$

More efficient, but loss of 1 block → no verification

TESLA

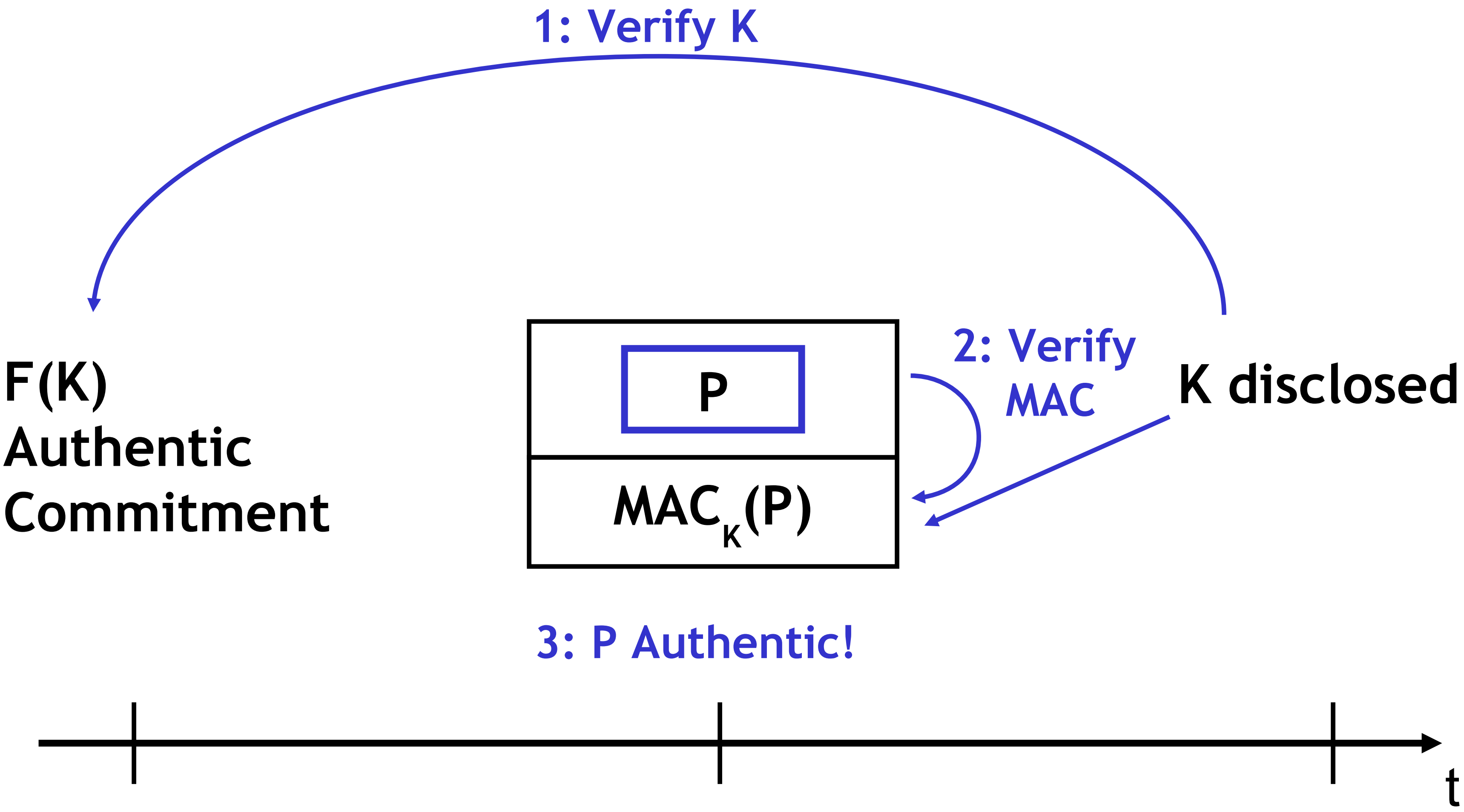
- TESLA = Timed Efficient Stream Loss-tolerant Authentication [Perrig et al., RSA Cryptobytes 2002]

- Uses only symmetric cryptography

- Asymmetry via time
  - Only the correct sender could compute MAC at time t
  - Delayed key disclosure for verification
  - Requires loose time synchronization

Delayed Key Disclosure

F: public one-way function



1: Verify K

F(K)
Authentic
Commitment

P

$MAC_K(P)$

2: Verify MAC

K disclosed

3: P Authentic!

t

One-Way Hash Chains

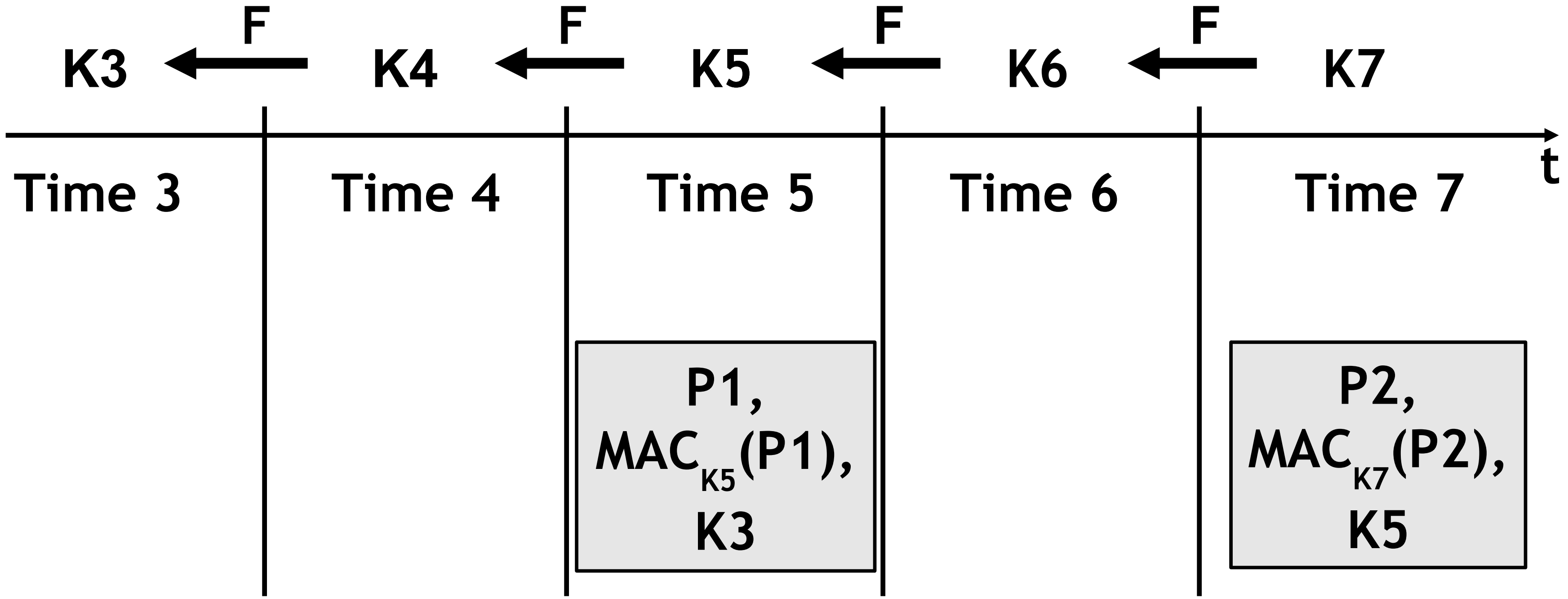
- Versatile cryptographic primitive
  - Pick random r_N and public one-way function F
  - For i=N-1,…,0 : $r_i = F(r_{i+1})$, then publish r_0

$$r_3 \xleftarrow{F} r_4 \xleftarrow{F} r_5 \xleftarrow{F} r_6 \xleftarrow{F} r_7$$

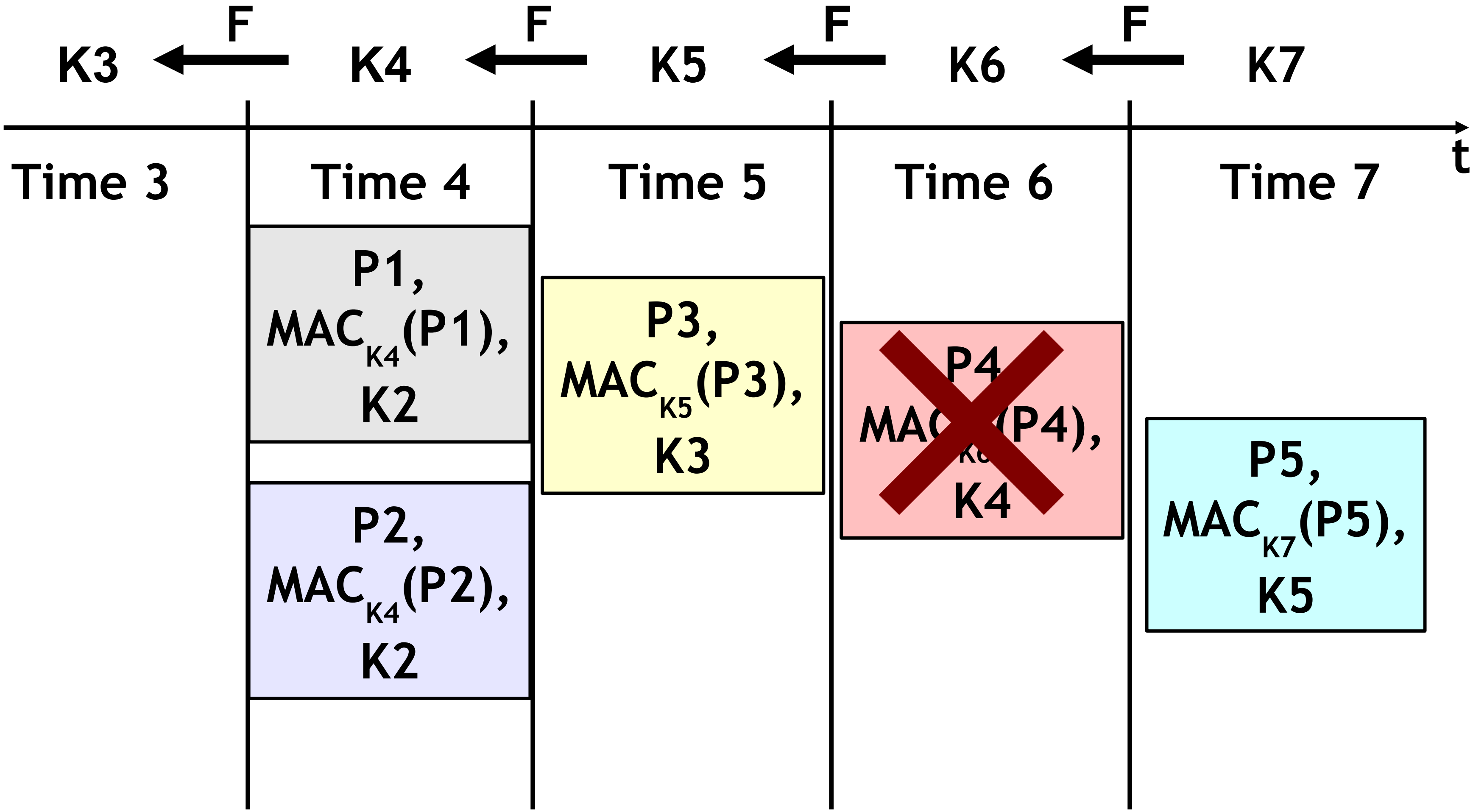
- Properties
  - Use in reverse order of construction: $r_1, r_2, \ldots, r_N$
  - Infeasible to derive r_i from r_j $(j<i)$
  - Efficiently authenticate r_i using r_j $(j<i)$: $r_j = F^{i-j}(r_i)$
  - Robust to missing values

TESLA Schedules

- Keys disclosed 2 time intervals after use
- Receiver setup: Authentic K3, key disclosure schedule



Robustness to Packet Loss



K3 ← F K4 ← F K5 ← F K6 ← F K7

t

Time 3

Time 4

P1, $MAC_{K4}(P1)$, K2

P2, $MAC_{K4}(P2)$, K2

Time 5

P3, $MAC_{K5}(P3)$, K3

Time 6

~~P4, $MAC_{K6}(P4)$, K4~~

Time 7

P5, $MAC_{K7}(P5)$, K5

Asymmetric Properties

- Disclosed value of key chain is a public key, it allows authentication of subsequent messages (assuming time synchronization)

- Receivers can only verify, not generate

- With trusted time stamping entity, TESLA can provide signature property

TESLA Summary

- Low overhead
  - Communication (~ 20 bytes)
  - Computation (~ 1 MAC computation per packet)
- Perfect robustness to packet loss
- Independent of number of receivers
- Delayed authentication
- Applications
  - Authentic media broadcast
  - Sensor networks
  - Secure routing protocols

What about highly-constrained nodes
in wireless sensor networks?

µTESLA for WSN

- Proposed as part of the SPINS architecture [Perrig et al., WiNet 2002]
  - Reduced communication compared to TESLA, key disclosure per epoch instead of per packet
  - Includes several other optimizations for minimum overhead, practical in severely-constrained devices

SNEP for WSN

- SPINS also includes the Secure Network Encryption Protocol (SNEP) to provide data confidentiality, authentication, and freshness [Perrig et al., WiNet 2002]
  - SNEP includes efficient key generation
  - SNEP authenticated + encrypted packet structure:
    - Data encrypted with shared key + counter (for semantic security)
    - MAC over encrypted data

$$A \rightarrow B: \quad \{D\}_{\langle K_{AB}, C_A \rangle}, \ \mathrm{MAC}\left(K'_{AB} C_A \mid\mid \{D\}_{\langle K_{AB}, C_A \rangle}\right)$$

- Optional nonce-exchange for provable freshness

TinySec

- The TinySec architecture provides a practical security suite for wireless sensor networks [Karlof et al., SenSys 2004]
  - TinySec-Auth provides authentication only
  - TinySec-AE provides authenticated encryption

  - Extensive discussion of design trade-offs and simulation results included in the paper

Further Reading

- Broadcast authentication in VANETs
  - Studer et al., ESCAR 2008 / JCN 2009.
  - Raya et al., SASN 2005.
    - More papers @ **http://lca.epfl.ch/projects/ivc/**

- … in WSN
  - Ren et al., WASA 2006.

- DoS-resilient broadcast authentication
  - Gunter et al., NDSS 2004.
  - Karlof et al., NDSS 2004.

In addition to security and performance features of the security protocols, what about the underlying **key management**?

Key Management

- How to add a member to the group without giving them access to past group activities?

- How to remove/revoke a member from the group without giving them access to future group activities?

- How to provide fresh credentials to group members?

Group Key Management

- Group formation, joining, and leaving can be controlled entirely by distribution and revocation of keys
  - A session encryption key (SEK) is given to all group members (used to distribute/collect data)
  - Key encryption keys (KEK) given to group members are used to periodically update SEKs
    - Revocation = not getting an SEK update
    - KEKs may also need to be updated
  - Updating key must be very efficient so it can occur often enough to minimize effects of misbehavior

Challenges

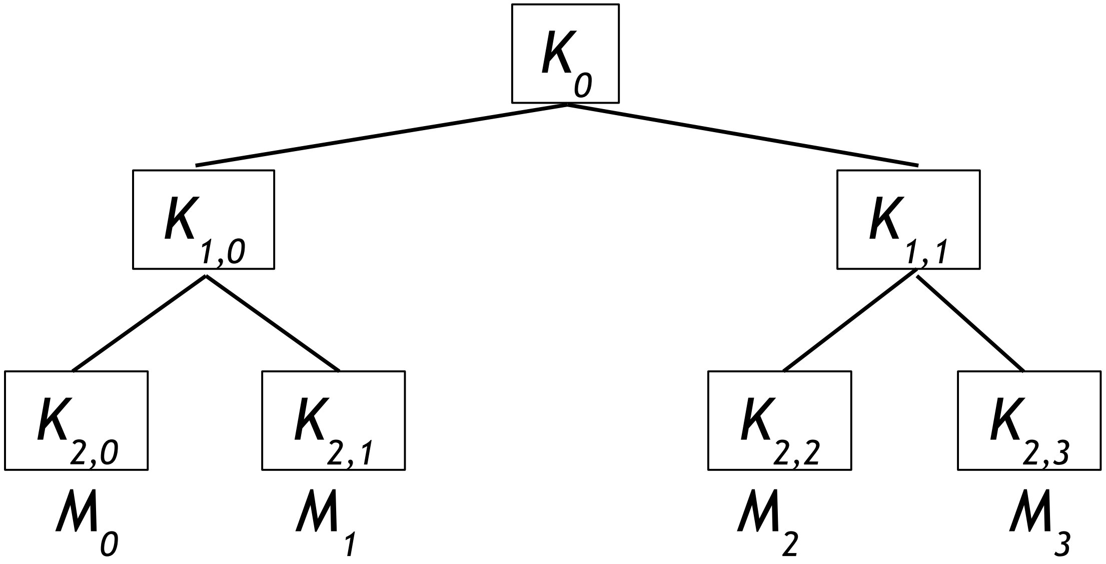
- Simple attack models such as eavesdropping, message injection / tampering, masquerading, etc. can affect the entire security architecture
- Unicast solutions may be infeasible / impractical
- Network and services are dynamic, need to scale
- Various types of overhead to manage
- Initial trust relationship

Scale & Dynamics

- Depending on the scenario, the group size could be 10s, 100s, 1000s, 1000000s, …
- Group membership and service subscription can be dynamic
  - Can change on the order of seconds, minutes, days, months, …
  - Join and leave are random
- Most likely, there's no “one size fits all” method

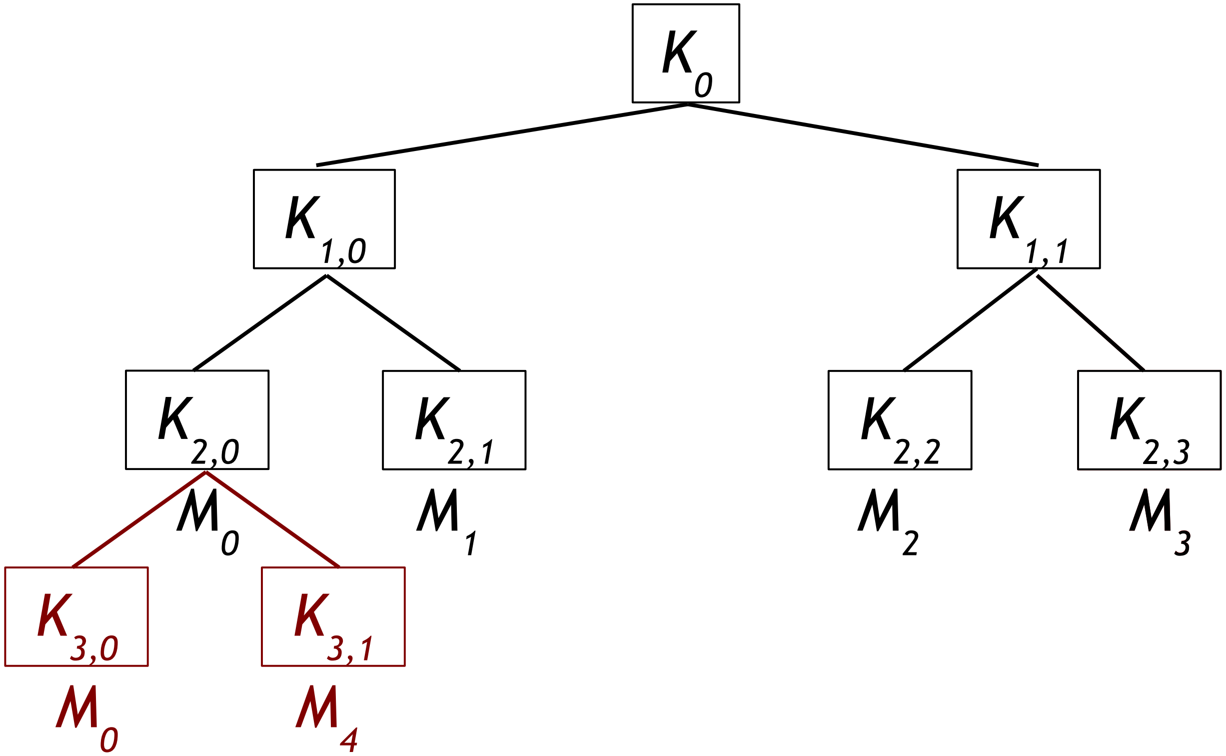
Logical Key Hierarchy

- LKH arranges group members in an *m*-ary tree
  - Tree leaves correspond to members with unique KEKs
  - Internal tree nodes represent group KEKs
  - Tree root represents the SEK
  - Each member gets SEK and KEKs along tree path



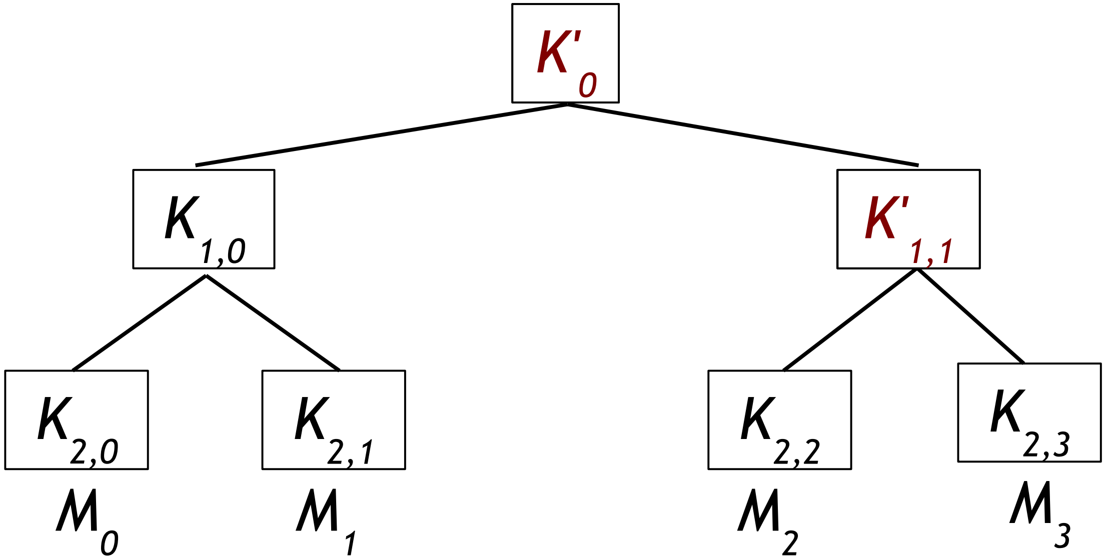
LHK Addition

- If M_4 wants to join a group and the tree is full
  - Start another level of the tree



LHK Removal

- If M_3 wants to leave the group, update SEK/KEKs
  - $\{K'_0, K'_{1,1}\}_{K_{2,2}}$
  - $\{K'_0\}_{K_{1,0}}$

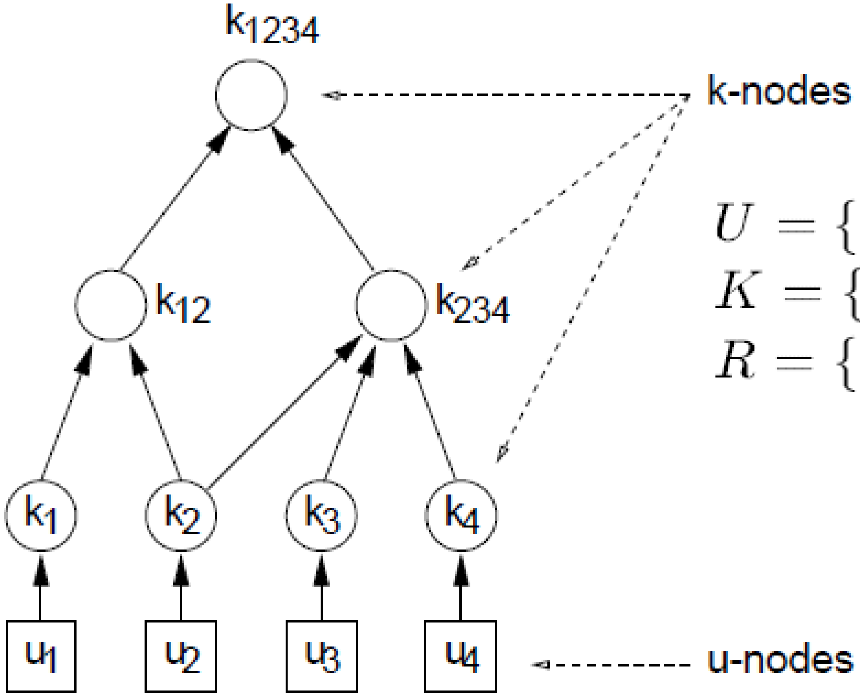


LKH Overhead

- Storage:
  - Authority holds O(N) total keys
  - Each member holds 1 SEK + $O(\log_m(N))$ KEKs
- Communication:
  - Broadcast flood required for every update message, $O(\log_m(N))$ msg/removal
    - Note: every msg may require multiple transmissions...
- Computation:
  - Symmetric en/decryption operations

Generalized Key Graphs

- Key graphs generalize key trees for secure group communication [Wong et al., TR 1997]
  - The authors propose a graph generalization of LKH allowing users to belong to groups arbitrarily instead of using a tree structure



$$
\begin{aligned}
U &= \{u_1, u_2, u_3, u_4\} \\
K &= \{k_1, k_2, k_3, k_4, k_{12}, k_{234}, k_{1234}\} \\
R &= \{\ (u_1, k_1),\ (u_1, k_{12}),\ (u_1, k_{1234}), \\
&\qquad (u_2, k_2),\ (u_2, k_{12}),\ (u_2, k_{234}),\ (u_2, k_{1234}), \\
&\qquad (u_3, k_3),\ (u_3, k_{234}),\ (u_3, k_{1234}), \\
&\qquad (u_4, k_4),\ (u_4, k_{234}),\ (u_4, k_{1234})\ \}
\end{aligned}
$$

How do these update procedures translate to the wireless domain?

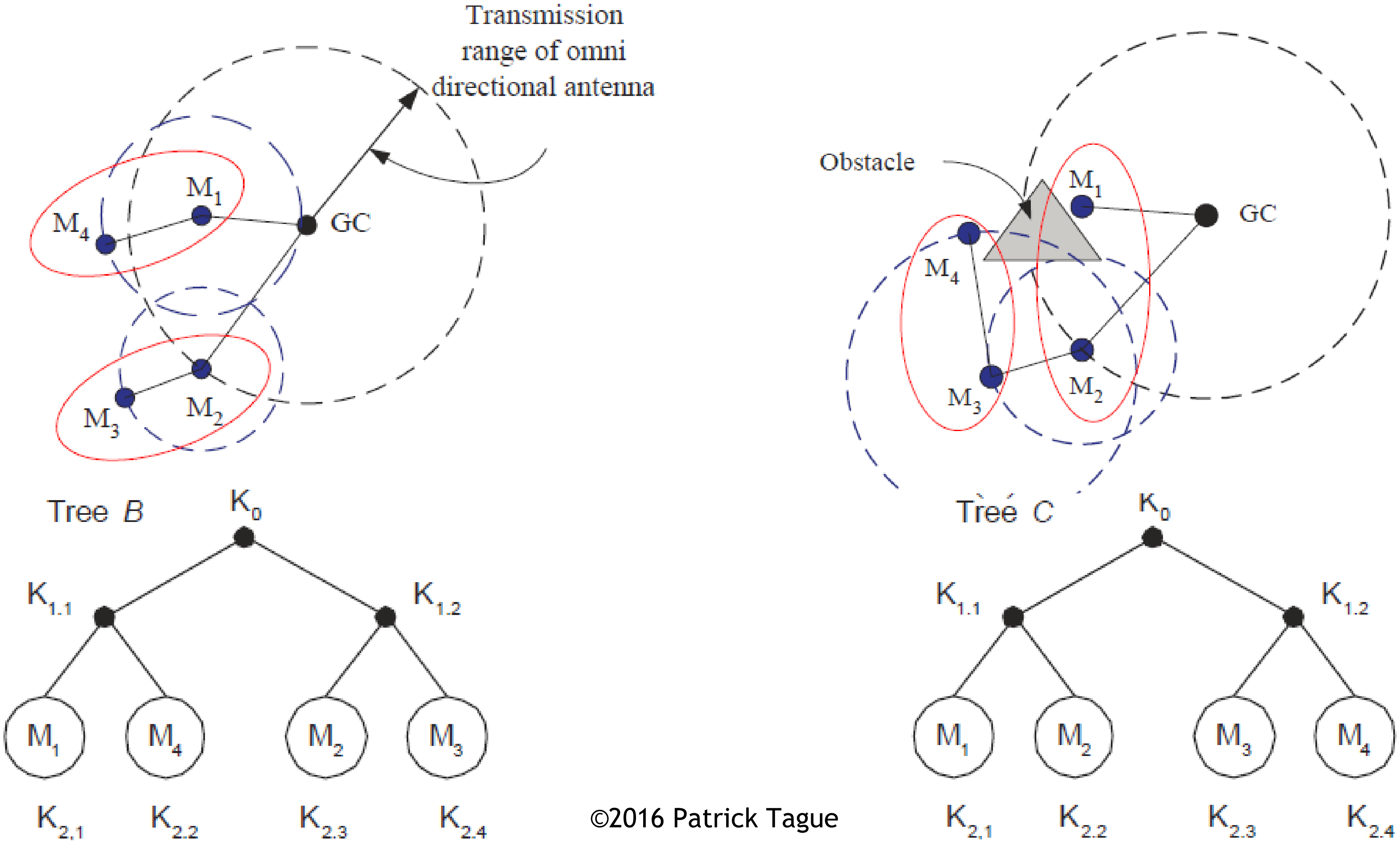
Metrics

- Previous techniques described focus on number of update messages to broadcast
  - What about the physical topology of the network?
  - Relaying messages over multiple wireless links?
  - Energy expenditure of long/lossy links?
  - Broadcast advantage?

Power-Efficient Key Trees

[Lazos & Poovendran, 2005]

- Key updates in large wireless networks (WSNs, MANETs, etc.) should be energy-efficient



But, in all these approaches, there's a catch...

Initial Key Agreement

- All of these key management approaches assume the new user is a valid user that can establish a pairwise KEK with the server
  - Valid user → authentication → keys
  - So, initial key agreement requires pre-existing keys or a secure offline initialization

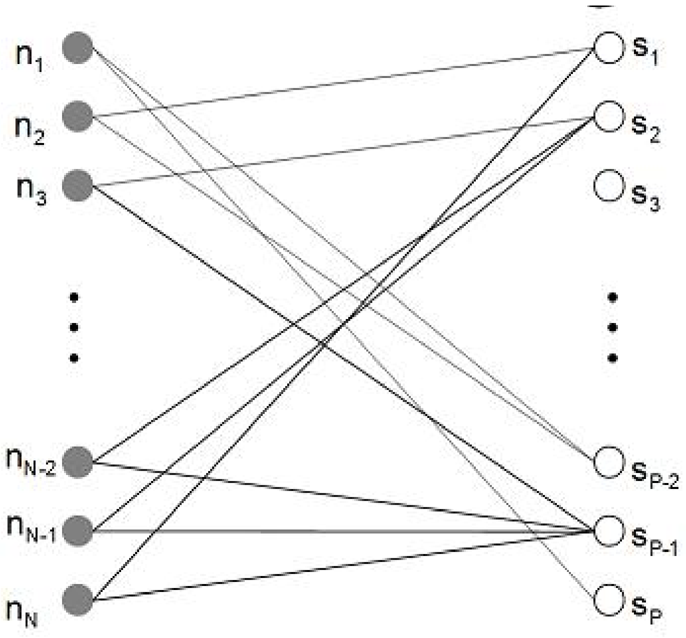
  - Protocols such as Diffie-Hellman and their many variants can help here, as long as they're practical for the context

  - Human-in-the-loop allows for different approaches, e.g., SafeSlinger [Farb et al., 2013]

Key Agreement in WSN

- In challenged systems (WSN), key agreement is often to expensive

- Option: authority assigns symmetric keys (KEK, etc.) prior to deployment, nodes that share SEKs/KEKs after deployment can bootstrap secure links



- See [Eschenauer & Gligor, CCS 2002; Tague & Poovendran, ToSN 2007]

This “pre-distribution” has its own class of associated threats/attacks

I can provide hundreds of papers if you're interested in learning more

Feb 11:

MAC Misbehavior;
OMNET++ Tutorial II