Key management? Wait, what?

IETF 71

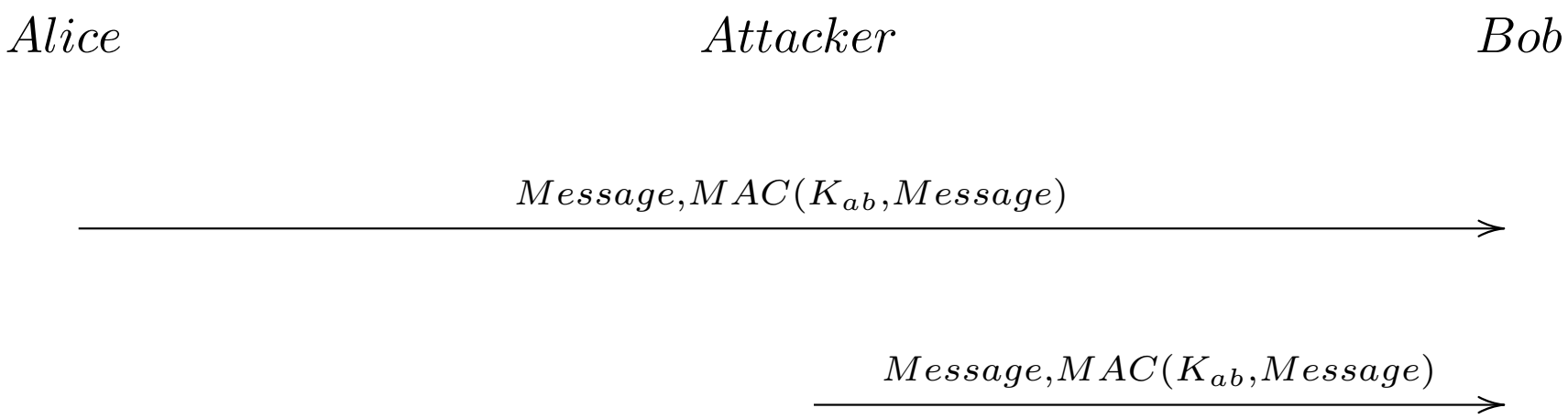
Current State of Routing Protocol Security*

- Routing protocols use manual key management
  - Shared keys are configured at the router interface
  - Traffic is protected with ad-hoc MACs based on those keys
- Why is this bad?
  - Replay and cut-and-paste attacks (see David Ward's presentation)
  - The MACs are generally pretty weak—and hard to upgrade

*We're just tallking about securing adjacencies here

What's a replay attack?

- Alice and Bob share a key (K_{ab})
  - But never change it



- Requires an on-path attacker
- Works whenever association parameters are repeated
  - Keys
  - Other message meta-data protected by the MAC (e.g., host/port)

Defenses against replay attack

- Fresh association parameters
  - Implicit
    - Example: TCP
      - Each connection has its own host/port quartet
      - If in the MAC then you can't replay between connections
      - ... unless you get a host/port collision
  - Explicit
    - Establish a fresh connection identifier
      - Force it to be unique
    - Fresh keys for the association
  - Using a key management protocol
  - This is the standard COMSEC solution
    - Provides generic security without trusting the main protocol

What's a key management protocol?

- We have a number of elements $\mathbf{E} = E_1, E_2, ...$ that want to communicate
  - They have some long term credentials
    - Shared symmetric key/password
    - Asymmetric key pairs
    - Keys + certificates
  - Generally can't use these keys directly for communication
- A key management protocol establishes shared cryptographic state
  - Traffic protection keys
  - Algorithms and other parameters

Why use a KMP?

- Security
  - Establish fresh keys
    - Prevents replay and cut-and-paste attacks
    - Good "cryptographic hygiene"
  - Explicit liveness and peer authentication
- Performance
  - Public key cryptography is very slow
  - Faster to establish symmetric keys and then use them
- Capability discovery
  - Get peer's certificate
    - Not an issue in systems with manual configuration
  - Discover/negotiate algorithms
  - Allows uncoordinated capability upgrade

Deployment Scenarios

- Unicast
  - This is a technically well understood problem (TLS, IPsec, ...)
    - Issue is mapping it onto actual protocols with minimal disruption
  - BGP, LDP
    - Run over TCP
    - Not the topic of this meeting (TCP-AO, etc.)
  - IS-IS, OSPF, RIP
- Multicast/broadcast
  - Less technically well understood
    - Some experience in MSEC WG (GDOI, GSAKMP)
  - IS-IS, OSPF, RIP only

A trivial unicast key management protocol

- Assume Alice, Bob share a key: K_{ab}

$Alice$ Bob

$$\xrightarrow[Algorithms=HMAC-SHA1,HMAC-SHA256]{Name=Alice,Random}$$

$$\xleftarrow[Algorithm=HMAC-SHA1]{Name=Bob,Random}$$

- Two new keys:

$$K_{a\rightarrow b} = HMAC(K_{ab}, Random_{Alice}||Random_{Bob})$$

$$K_{b\rightarrow a} = HMAC(K_{ba}, Random_{Bob}||Random_{Alice})$$

Similar protocols can be used with asymmetric keys

Key Management for Multicast Groups

- We have a set of elements $\mathbf{E} = E_1, E_2, ...$
  - They have some long term credentials
  - We want them to share a single symmetric key, parameters, etc.
- Classic solution: Have a master element (group controller)
  - Generates group key
  - Forms unicast associations with each element
  - Pushes group key to each group member
    * Send $E(K_{E_i}, K_{group})$ to node E_i
- Examples: GDOI, GSAKMP

Key Management with a Group Master