Rekey protocol

Lakshminath Dondeti, Nortel
David McGrew, Cisco
IETF-52, SLC

Rekey protocol issues

- Where does it belong within MSEC?
  - GKMArch or separately?
- What about reliability?
  - Multiple resends, FEC, reliable multicast
- Key tree management
- Back channel anyone?
  - Discussed at IETF-51 (De-registration)
  - Why do we need it?

Spinoff from GKMBB

- Before IETF-51, rekey protocol I-D was separated (GKMBB → GKMArch)
  - Because it is optional
  - Reliability issues
  - Back channel issues
  - Key tree management issues
  - Parent I-D hindering its progress, not really! ☺

Merger proposal

- Spinoff had some hits and misses
- GKMArch progressed much faster
- Rekey issues still unresolved
  - That's a reason to keep it separate: Ran
- Spinoff was a bad idea anyway
  - Might mean splitting GDOI and GSAKMP
- Rekey protocol is part of KM
- Need WG consensus on this

Rekey protocol issues

- Where does it belong within MSEC?
  - GKMArch or separately?
- What about reliability?
  - Multiple resends, FEC, reliable multicast
- Key tree management
- Back channel anyone
  - More or less NO at IETF-51?
  - Why do we need it?

Reliable transport of rekey messages

- Send multiple times and hope all members receive the message
  - Might actually work!
- Use FEC with NACKs, and unicast to finish off
  - Proposed by folks from UT-Austin
- Over a reliable multicast channel

Intelligent retransmissions

- We all know those are the options
- How about sensible retransmissions?
  - Half the members need only one key
- We should take advantage of that!
- GDOI rekey message
  - HDR*, SEQ, SA, KD, [CERT,] SIG
- GSAKMP rekey message
  - {HDR, GrpID, [PT], Rekey Array}SigC, [CertC]

Split the rekey message

- HDR*, SPI, SEQ, PT, [CERT], SIG
  - *Protected by rekey SA
- KD_HDR*, SPI, KDx, SIG
  - x = 1, …, j
  - j indicates number of partitions
- j+1 signatures by the rekey server
- Signature verifications also increase
- Rekey retransmission traffic should be less
- Let us not disallow splitting rekey msgs

What's in KD payload?

Next payload	Reserved	Payload length
Number of key packets		Reserved2
Key packets (variable length)		

Key packet (KP) payload

KD type	Reserved	KD length
SPI size	SPI (variable length)	
Key packet attributes (variable length)		

Rekey message size

- Depends mainly on # of KP payloads
  - 25 B
- logn + 1 key packets in OFC
- n = 2^{16}→16 KP payloads, i.e.,
  - 400 B
  - 1 packet (576 B per packet)
- Current model should work just fine!?

What about batch rekeying?

- O(r log(n/r)) KP payloads to be sent
- $r = 2^6$, $n = 2^{16}$ → 640 KP payloads
  - ~ 16 KB
  - ~ 28 packets
- Reliable transport is more challenging
- Send 1 missing packet instead of 28!

Key tree initialization

- O(nlogn) KPPs during registration
- O(n) during registration +
  O(n) during rekey initialization
- N = 2^{16} → 65536 KP payloads
  - 1600 KB
  - ~ 2845 packets
- We may want to split rekey messages!

Rekey protocol issues

- Where does it belong within MSEC?
  - GKMArch or separately?
- What about reliability?
  - Multiple resends, FEC, reliable multicast
- Key tree management
- Back channel anyone
  - More or less NO at IETF-51?
  - Why do we need it?

Key tree management

- How do members know which keys to decrypt?
  - Use key IDs that don't change
  - Communicate key ID changes
  - Communicate join/leave, and key tree maintenance information
- Should this be part of a rekey message?

Rekey protocol issues

- Where does it belong within MSEC?
  - GKMArch or separately?
- What about reliability?
  - Multiple resends, FEC, reliable multicast
- Key tree management
- Back channel anyone
  - More or less NO at IETF-51?
  - Why do we need it?

Back channel

- Mainly for NACKs in reliable rekeying
- Any other uses?
  - Membership management

Conclusion

- Rekey protocol description as part of GKMArch I-D
- Split the rekey message
- Key tree mgmt info in rekey message
- Back channel
  - Reliable transport of rekey messages
  - Membership management