KSP Update

A distributed fault-tolerant group key selection protocol

Mick Seaman
mick_seaman@ieee.org
KSP Update

- Purpose and motivation (recap)
- Protocol overview
- Examples of protocol use (2, 3, n participants)
- An object oriented description
- State machines and processes
- Proofs – secure, correct, converges (outline)
- Goals (recap)
KSP Update

- **Purpose and motivation (recap)**
- Protocol overview
- Examples of protocol use (2, 3, n participants)
- An object oriented description
- State machines and processes
- Proofs – secure, correct, converges (outline)
- Goals (recap)
Purpose

- **Provide MACsec CA with fresh group keys**
  - Following system initialization
  - As PN space is exhausted
  - Point-to-point and group CAs

- **Support MACsec replay and delay protection**
  - Liveness and timeliness

- **Robust against system failure**
  - Systems may join and leave the CA
  - Authentication Server not guaranteed accessible
Motivation

Retain important LAN capabilities & performance..

- Natural multicast and broadcast
  - Full mesh pt-to-pt not the same performance

- Rapid reconfig for fault-tolerant reliability
  - Orders of magnitude faster than IP recovery

.. with low incremental cost over pt-to-pt only

Head off poor timer based & loss sensitive designs
KSP Update

- Purpose and motivation (recap)
- **Protocol overview**
- Examples of protocol use (2, 3, n participants)
- An object oriented description
- State machines and processes
- Proofs – secure, correct, converges (outline)
- Goals (recap)
Protocol overview

- Key contribution, generation, and identification
- KSPDU design, step by step
- Quantifying protocol simplicity
Key contribution (KC)

128 bits from each participant

• New KC on reinit
• New KC whenever derived SAK out of PN space
Data key (SAK) generation

Pseudo-random function of CAK and each KC

- Independently calculated by each participant
- Ensures every participant has contributed to every key used for transmission
- SAK and high water mark PN recorded, lest participant changes result in SAK reuse
- Else participant forces new SAK by submitting new KC
Key identifier (KI)

128 bit exclusive-or of all KCs

- Confirms calculation of same key by all (to high probability)
- Provides no additional information to attacker (independent of SAK generation)
- Collisions can lose data, not security
- Labels previous key(s) in support of continuous connectivity
KSPDU design, step by step (1)

- Each participant makes a Key Contribution
  
  \( \text{KSPDU} = \text{KC} \)

- + Key Identifier, so agreement can be recognised
  
  \( \text{KSPDU} = \text{KC, KI} \)

- + receive flag, when set by all transmission can start

- + transmit flag, when unused by all transmission can stop

  \( \text{KSPDU} = \text{KC, KI.r.t} \)
KSPDU design, step by step (2)

- Two keys provide continuity of communication, through membership changes and PN exhaustion
  
  KSPDU = KC, LKI.r.t, OKI.r.t

- Need to be bound to transmitter’s MACsec SAs
  
  KSPDU = SCI, LAN, OAN, KC, LKI.r.t, OKI.r.t

- Lowest acceptable PNs bound delay
  
  KSPDU = SCI, LAN, OAN, KC, LKI.r.t, LPN, OKI.r.t, OPN

- Member Identifier distinguishes prior participant instances
  
  KSPDU = SCI, MI, LAN, OAN, KC, LKI.r.t, ...
KSPDU design, step by step (3)

• Message number prevents replay & out-of-order delivery
KSPDU = SCI, MI, MN, LAN, OAN, KC, LKI.r.t, ...

• Including peers’ MI, MN proves liveness & timeliness
KSPDU = SCI, MI, MN, ..., (MI, MN, .. MI, MN)

• Distinguishing live and potential peers prevents premature key choice and speeds liveness proofs
KSPDU = SCI, MI, MN, ..., (MI, MN, ..) (MI, MN, ..)
KSPDU design, step by step (4)

- CKI identifies master key (CAK) for integrity protection
  \[ \text{KSPDU} = \text{CKI}, \text{IV}, \text{SCI}, \text{MI}, \text{MN}, ..., \text{ICV} \]
- Using random IV means do not have to recurse key gen.
  \[ \text{KSPDU} = \text{CKI}, \text{IV}, \text{SCI}, \text{MI}, \text{MN}, ..., \text{ICV} \]
- Correct ICV proves current possession of CAK
  \[ \text{KSPDU} = \text{CKI}, \text{IV}, \text{SCI}, \text{MI}, \text{MN}, ..., \text{ICV} \]
- Integrity, not confidentiality, allows debug by field operations without CAK knowledge/disclosure
KSPDU design, step by step (5)

- Ethertype identifies the KSP protocol
  
  $\text{KSPDU} = DA, \text{SA, ET, CKI, IV, SCI, MI, MN, ..., ICV}$

- Multicast address allows single transmission to reach all peers, address used restricts peers to single LAN
  
  $\text{KSPDU} = DA, \text{SA, ET, CKI, IV, SCI, MI, MN, ..., ICV}$
Quantifying protocol simplicity (1)

An objective non-emotional basis

- Beyond ‘simplicity is familiarity’
- Identifies potential for joint state explosion
- System state = Participant state ** participants
- Particularly important for multicast, n > 2 participants
- Exposes many sub-protocol partitionings as facile
Quantifying protocol simplicity (2)

P = participant state
a, b = messages
+ is reception, adds to produce new P
- is transmission, taking away a message from P

Then, for the simplest protocols

- P + a = P + a + a messages are ‘idempotent’
- P - a = P except for transmit limiters
- P + a + b = P + b + a messages commute, misordering immaterial
KSP Update

- Purpose and motivation (recap)
- Protocol overview
- Examples of protocol use (2, 3, n participants)
- An object oriented description
- State machines and processes
- Proofs – secure, correct, converges (outline)
- Goals (recap)
Example: 2 participants

Stations $S_A$, $S_B$ each with MI+MN of A+.., B+..

Messages comprise: Actor | Live list | Potential list

$S_A \rightarrow A+1, KC_A || \rightarrow S_B \ldots \ (1)$

$S_A \leftarrow B+1, KC_B || A+1 \leftarrow S_B \ldots \ (2)$

$S_A \rightarrow A+2, KC_A, Kl_{AB.r} | B+1 | \rightarrow S_B \ldots \ (3)$

B now receiving and transmitting using $SAK_{AB}$

$S_A \leftarrow B+2, KC_B, Kl_{AB.rt} | A+2 | \leftarrow S_B \ldots \ (4)$

A now receiving and transmitting using $SAK_{AB}$

Exchange equivalent to 4-way handshake
Example: 3rd participant joins

.. continuing prior example. $S_A, S_B$ continue data transfer with $SAK_{AB}$, but represent that as OKI in protocol, omitted from following description for simplicity

$S_C \leftarrow B+2, KC_B, KL_{AB}\text{.rt} | A+2 | \leftarrow S_B \ldots (4)$

$S_C \rightarrow C+1, KC_C || A+2, B+2 \ldots (5)$

$S_A, S_B \leftarrow$

$S_A \rightarrow A+3, KC_A, KL_{ABC}\text{.r} | B+2, C+1 | \ldots (6)$

$S_B \rightarrow B+3, KC_B, KL_{ABC}\text{.r} | A+3, C+1 | \ldots (7)$

$S_B, S_C \leftarrow (6) \ldots S_A, S_C \leftarrow (7)$

$S_C \rightarrow C+2, KC_C, KL_{ABC}\text{.rt} | A+3, B+3 | \ldots (8)$

$S_B, S_C \leftarrow (8) \ldots$ all now rxing, txing $SAK_{ABC}$
Example: participant leaves

.. $S_B$ leaves after $S_A$, $S_B$, $S_C$ have agreed $SAK_{ABC}$. Say $S_A$ times out $S_B$ first, $KI_{ABC}$ will now be OKI, omitted for simplicity.

$S_A \rightarrow A+n, KC_A, Kl_{AC}.r | C+m | B+3 \quad ... (9)$

... finally $S_C$ times out $S_B$

$S_C \rightarrow C+m+1, KC_C, Kl_{AC}.rt | A+n | B+3 \quad ... (10)$

$S_A \leftarrow$

both now rxing, txing $SAK_{AC}$
KSP Update

- Purpose and motivation (recap)
- Protocol overview
- Examples of protocol use (2, 3, n participants)
- An object oriented description
- State machines and processes
- Proofs – secure, correct, converges (outline)
- Goals (recap)
An object oriented description

• Why
• Notation
• The big picture – Kay and Ksp
• A single Ksp instance
• The small picture – a KSPDU

See ../docs2004/af-seaman-ksp-object-machines-001.pdf
KSP Update

- Purpose and motivation (recap)
- Protocol overview
- Examples of protocol use (2, 3, n participants)
- An object oriented description
- **State machines and processes**
- Proofs – secure, correct, converges (outline)
- Goals (recap)
State machines and processes

- Relationship to the OO description
- Ksp Key Machines (KKM)
- Actor Machine
- Peer Machines
- Receive KSPDU processing
- Deciding to use a key
KSP Update

• Purpose and motivation (recap)
• Protocol overview
• Examples of protocol use (2, 3, n participants)
• An object oriented description
• State machines and processes
• Proofs – secure, correct, converges (outline)
• Goals (recap)
Proofs (outline)

- What needs to be proved
- Threats
- Correctness
- Convergence
Threats

- **Passive attacker** – can observe all frames but not remove, or add, or control equipment.
- **Active attacker** – can observe, modify, selectively deliver, add frames, control equipment, power but not physically modify equipment.
- **Thief** – can remove equipment containing master key and attempt to use elsewhere on network.
Correctness

Attacker cannot:

- Learn key by any observation or manipulation
- Force reuse of a key nonce pair

Because the SAK (data key):

- Is a pseudo-random function using a CAK unknown to the attacker, KSP security does not at all depend on analysing protocol messages (which carry clear data)
- Is a function of KCs from all participants
  - Changed whenever derived key/nonce history forgotten
  - Depends on participants, not reusable otherwise
Convergence

- Protocol will converge on to a useable key following a short known bounded time after all messages are correctly delivered and system power remains unchanged (4 to 6 seconds depending on detail).
- Attacker only adding traffic, including replay of all messages with same master key can only add a one time fixed delay to convergence, not prevent it.
- Pure “wire-cutting” attacks cannot be prevented.
KSP Update

- Purpose and motivation (recap)
- Protocol overview
- Examples of protocol use (2, 3, n participants)
- An object oriented description
- State machines and processes
- Proofs – secure, correct, converges (outline)
- Goals (recap)