MACsec with Pre-shared Keys

James A. McIntosh

v0.1 – April 27, 2023

Introduction

- **Certain applications, such as automotive applications, require fast start-up times.**
  - These times can be on the order of milliseconds, including device link-up times, etc.
  - These time constraints can make key exchange difficult while maintaining security in the system.
- **Many devices in these applications are intended to be small and inexpensive.**
  - These devices will not likely want to implement large cryptographic blocks, such as public key cryptography.
  - Many will also want to avoid needing an on-chip CPU.

Ideas

- **There are several ideas that I've considered as potential solutions, each with its own pros and cons (going from simplest to most complex).**
  - Pre-shared SAKs
    - These are installed in a "safe" environment and are intended to have long lives (years?).
    - XPN is likely used in this case.
  - Pre-shared "Key Exchange Keys"
    - They might use simple encryption of SAKs
    - They might be true KEKs as used in MKA defined in 802.1X and use AES key wrap/unwrap.
  - Pre-shared CAKs as used in MKA defined in 802.1X
    - Can derive KEKs and ICKs given inputs from the host (requires AES-CMAC)
    - Can derive SAKs given inputs from the host (requires AES-CMAC)
    - Can receive SAKs from the network (requires AES key unwrap)

- **Beyond these, a CPU is likely required.**

Pre-shared SAKs

- **These will need non-volatile storage for multiple (how many?) pre-shared SAKs**
- **The keys are expected to have a long life, perhaps years**
  - This will likely necessitate XPN for all but the slowest data rates.
- **Provisions will be needed to install or invalidate the keys in a "safe" environment.**
  - In the factory.
  - At the local auto shop where a new part might be installed.
- **How do we make this safe?**
- **Once a key is compromised, it will have to be invalidated and replaced, shortening the life of the part.**
- **Once all the keys are invalidated, the part is "bricked".**

Pre-shared Key Exchange Keys

- **Many considerations are the same as the pre-shared SAK case, except that the session keys are loaded at each start-up from the host.**
- **There are different possible methods that could be used here.**
  - Simple Method: SAKs are simply encrypted with the pre-shared key and passed to the device where it is decrypted
    - This might use basic AES.
    - What are the strengths and weaknesses of this idea?
  - Less Simple Method: The pre-shared key is the KEK defined in 802.1X for MKA.
    - This would be slightly more complex and would require AES key unwrap in the device.
    - Is this a real improvement over just simple encryption of the key?

- **In general, this seems safer than the pre-shared SAK case, but has more complexity and a slightly longer start-up time.**

Pre-shared CAKs

- **Many considerations are the same as the other pre-shared key cases in terms of installation, key invalidation, etc.**
- **With pre-shared CAKs, the device can:**
  - Derive KEKs and ICKs given inputs from the host (requires AES-CMAC)
  - Derive SAKs given inputs from the host (requires AES-CMAC)
  - Receive SAKs from the network (requires AES key unwrap)

- **This seems generally safer than the previous cases, but has more complexity yet, and even longer start-up times.**

Discussion?

Backup

CAK Key Hierarchy defined in 802.1X

Key Hierarchy

- **Basic key hierarchy:**
  - CAK used to derive ICK and KEK
  - SAK can be from strong RNG or derived from CAK, RNG, and other data

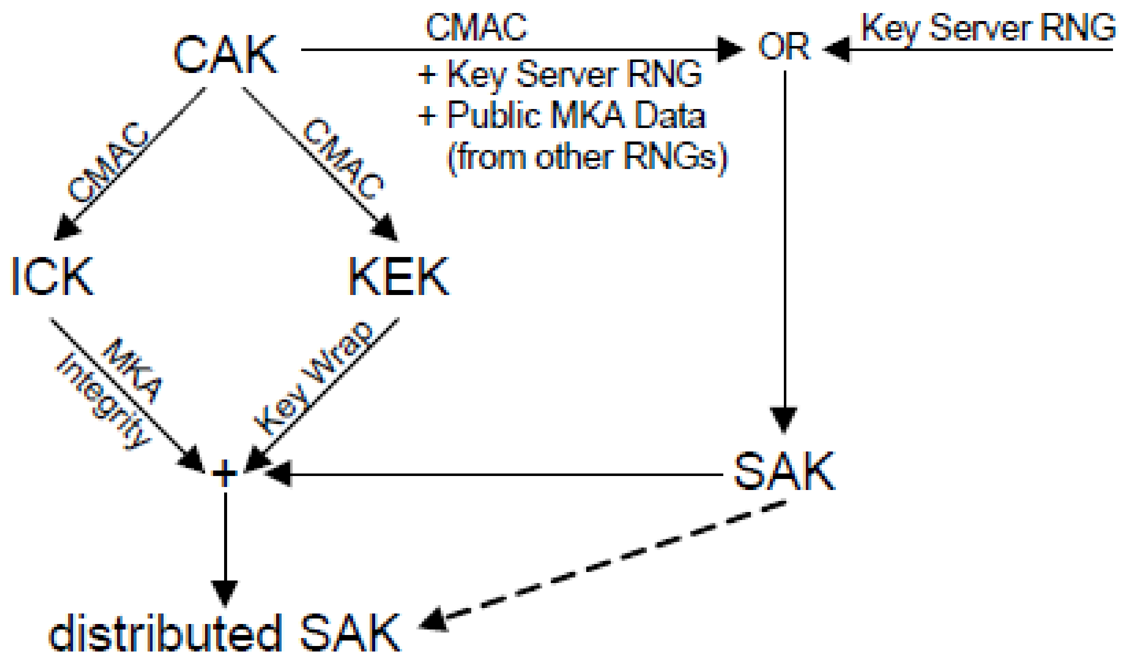

Figure 6-3—MKA key hierarchy

Group CAKs Distributed via Pairwise CAKs

- Pairwise CAKs are uses to generate the pairwise ICKs a KEKs.
- These, in turn, are used to distribute a group CAK.
- This group CAK is used to generate the group ICK and KEK, which is used to distribute the SAKs

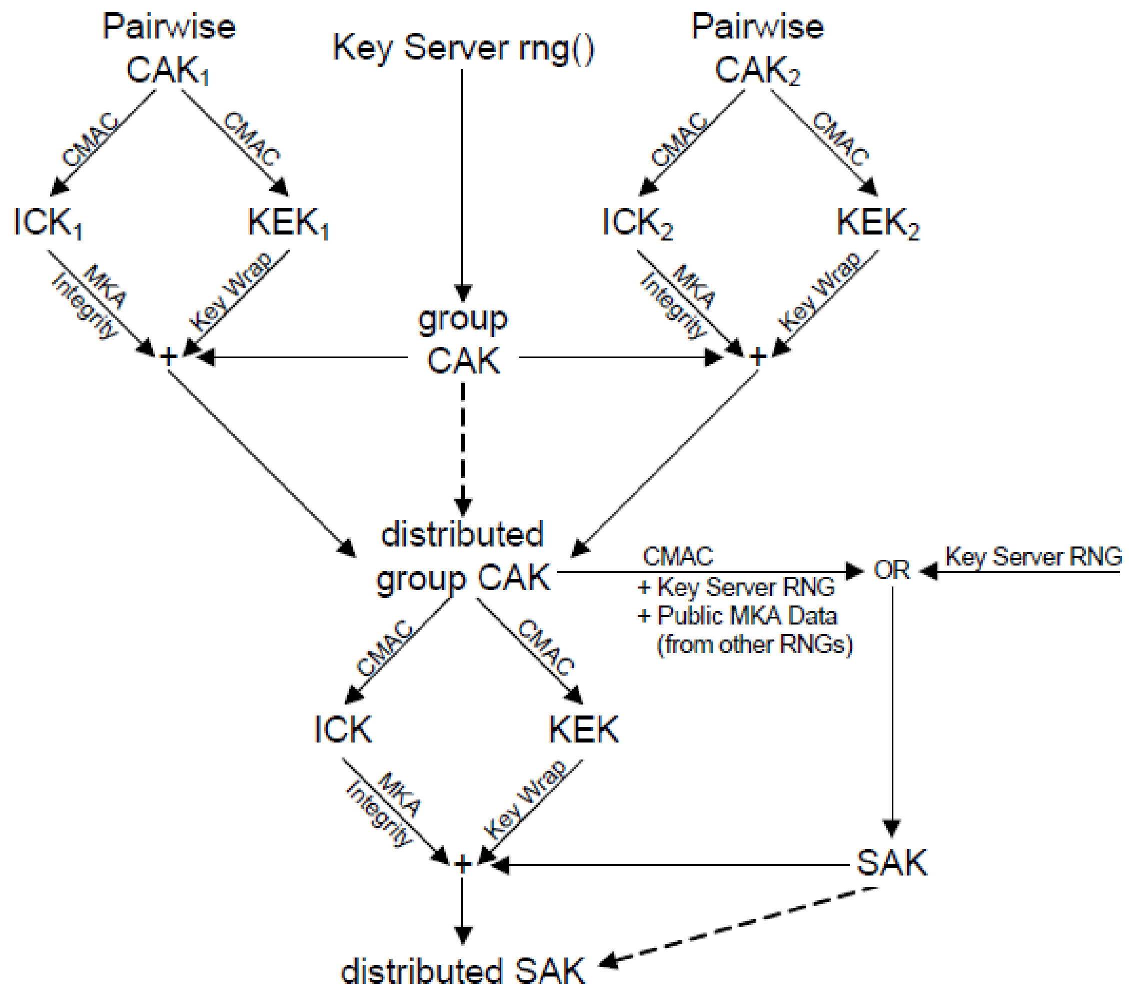

Figure 6-4—Use of pairwise CAKs to distribute group SAKs