Key Encryption as per T10/06-103

David L. Black (author)
Jack Harwood (presenter)
Problem and Design Goals

- 05-446 only specifies encryption key transfer in clear
  - Keys can be entirely too valuable to an attacker
  - Transfer by reference isn’t sufficient

- Goal: means of encrypting keys for network transfer
  - iSCSI and Fibre Channel (FCP) are primary concerns
  - IPsec (iSCSI) and FC-SP encrypt everything
    - Too high a price (e.g., hardware) to just protect keys
  - Just encrypting keys (wrapping) can be done in software
    - And is transparent to protocol bridges.

- Goal: Drop in replacement for key transfer in clear
  - Customer can decide which to use
    - E.g., check this box to use encryption for keys sent to device
  - Defends against eavesdropper, but not active attacker
Design Constraint: Keep it Simple

- Make it possible to implement in reasonable amount of time
  - And complete the specification in reasonable amount of time 😊
- No Authentication
  - Authentication identities and credentials violate drop-in replacement
  - TCG and others have or will develop authentication protocols
    - This proposal can support other authentication protocols
  - No certificates: let others play in this tarpit
- No Additional Authorization or Access Control
  - Preserve 05-446 model of what allows an initiator to key a device
- No FIPS certification
  - Passing keys in clear can’t be FIPS certified, why should this be?
- No copy manager support (not in 05-446 either)
Functionality Outline

1. Algorithm and Parameter Selection
   • Crypto algorithms, key sizes

2. Key Exchange
   • Create a shared secret key that an eavesdropper can’t see

3. Key Derivation
   • Based on key exchange, create keys for encryption, integrity of wrapped keys

4. Wrap Device encryption key(s)
   • Encrypt, add integrity check, sequence number

5. Send Wrapped Keys to Device

6. Decrypt and Verify
   • Verify: Integrity check, replay prevention
Functionality Realization: Significant IETF IPsec Reuse

1. Algorithm and Parameter Selection
   • Device server announces algorithms and parameters

2. Key Exchange
   • Diffie-Hellman phase of IPsec IKEv2 (IETF RFC 4306)

3. Key Derivation
   • IPsec IKEv2 key derivation (IETF RFC 4306)

4. Wrap Device encryption key(s)
   • IPsec ESP (IETF RFC 4303)
   • AES-GCM (IETF RFC 4106) will be one algorithm

5. Send Wrapped Keys to Device
   • SECURITY PROTOCOL OUT with new Key Format value

6. Decrypt and Verify
   • Based on ESP (IETF RFC 4303)
Algorithm and Parameter Selection

- Announcement by Device Server
  - There are a few required options
  - Device server picks one of each and announces them
  - Avoids complex negotiation mechanisms
  - Avoids defending against man-in-the-middle downgrade

- What is announced (6 x 16-bit numbers):
  1. Protocol version number (of this protocol)
  2. Diffie-Hellman group for Key Exchange
  3. Pseudo-random function for Key Derivation
  4. Encryption Algorithm
  5. Key Length for Encryption Algorithm
  6. Integrity Algorithm (not used with combined-mode encryption)

- Announcement includes all important IKEv2 parameters
  - Issue: Prefer not to change to initiator choosing from device capabilities
Diffie-Hellman Key Exchange

- Based on discrete log problem:
  - Given $g^x \mod n$, hard to compute $x$ when $n$ is a large prime
  - $x$ and $y$ are random numbers, $g^{xy} \mod n$ is result

```
g^x
```
```
g^y
```
```
g^{xy}
```
Proposed Key Exchange

- Nonces passed along with Diffie Hellman values
  - Nonce = Freshly-generated truly random number
  - Nonces allow DH values to be reused
- Device Server generates SPI to identify resulting keys
  - SPI = Security Parameters Index
- Two SCSI commands accomplish the key exchange:
  1. SECURITY PROTOCOL IN (proposed to page 12h)
  2. SECURITY PROTOCOL OUT (proposed to page 12h)
  - Same commands handle algorithm and parameter announcement
  - No payload diagrams yet (sorry)
- Proposed Diffie-Hellman Groups
  - 2048-bit and 3072-bit (IETF RFC 3526)
Key Derivation

• Result of DH Key exchange is not an encryption key
  – Large number with low relative entropy (randomness)
  – 2048-bit result has less than 128 bits of randomness

• Encryption needs high relative entropy key
  – At least 100 bits for 128 bit key

• Key Derivation spans this gap
  – Keyed pseudo-random function compresses nonces and result of
    DH key exchange to produce required session key(s)

• Proposed pseudo random functions
  – HMAC_SHA1, AES_XCBC (IETF RFC 2104, 3664)
Wrap Device Encryption Keys

- Encapsulation Format: IPsec ESP (IETF RFC 4303)
  - Adds SPI, Sequence Number, Initialization Vector (IV) and Integrity Check Value (ICV) to encrypted Key

- Encryption Algorithms: AES-GCM (IETF RFC 4106)
  - Galois Counter Mode
  - Combined mode: Single pass encrypts and generates ICV
  - Efficient: ICV is significantly less computation than a secure hash integrity check (e.g., HMAC_SHA1)

- Need to pick a second algorithm
  - Proposal: AES-CCM (IETF RFC 4309)
  - Another combined mode, simpler specification and implementation
  - AES-GCM and AES-CCM are the proposed IEEE P1619.1 tape encryption algorithms
Send Wrapped Keys, Decrypt and Verify

• Define Key Format 02h as ESP-wrapped keys
  – Use 05-446’s SECURITY PROTOCOL OUT command to send

• Anti-replay: Sequence numbers shall be used in order
  – Device server rejects any out-of-order usage
    ▪ Alternative: IPsec-like small window of acceptable sequence numbers
  – Sequence number at device server only advances when decrypt and verify succeeds

• Decrypt and Verity
  – AES-GCM or AES-CCM and its integrity check
  – Discard results if integrity check fails
Adding Additional Keying Algorithms

- SPI links key generation to usage
  - Can use other key generation algorithms
  - Device Server has to ensure that SPI’s are unique

- Some authentication algorithms can generate keys
  - TCG is a likely source

- Requirements on additional keying algorithms
  - Specify key exchange, key derivation, and how device server supplies SPI to initiator

- Device server only has to associate session (ESP) keys and ESP parameters with SPI
  - Can forget how keys were generated after generation is done.