Just Fast Keying (JFK)

Angelos Keromytis (Columbia University)

Bill Aiello (AT&T Labs - Research)
Matt Blaze (AT&T Labs - Research)
Steve Bellovin (AT&T Labs - Research)
Ran Canetti (IBM T.J. Watson Research Center)
John Ioannidis (AT&T Labs - Research)
Omer Reingold (AT&T Labs - Research)

Draft

- draft-ietf-ipsec-jfk-00.txt

Requirements

- Security
- Simplicity
- Low number of roundtrips
- Minimize "options"
- DoS mitigation
- Provably secure
- Protection of Initiator from active identity discovery attacks

The Protocol

I->R: Ni, g^x

R->I: Ni, Nr, GRPINFO, IDr, g^y, Sig(g^y, GRPINFO),
HMAC{Hkr}(g^i, g^r, Ni, Nr)

I->R: Ni, Nr, g^i, g^r, CK,
HMAC{Hkr}(g^i, g^r, Ni, Nr),
E{Ke}(IDi, sa, Sig(Ni, Nr, g^x, g^y, IDr, sa))

R->I: E{Ke}(Sig(Ni, Nr, g^x, g^y, IDr, sa, sa'), sa')

Important details

- Ke and "application" keying material derived from g^xy, Ni, Nr
- Responder (and Initiator) can reuse g^x and g^y, key material changes
- sa, sa’ purposely left undefined at this stage
  - Uni/Bi-directional SA establishment
  - Selectors

More details

- Responder does not keep state on receiving Msg 1
- Responder exposes identity in Msg 2
  - Initiator is protected from active identity discovery attacks
- HMAC is produced/verified by the Responder only
  - Can be something other than HMAC
- HMAC is used to quickly discard DoS packets
  - Patricia trie, O(w) lookup
  - Reset when Hkr is changed
- IP embedded in cookie
  - Implementation issue

Implementations

- 2 Java, 1 C, 1 Perl (student) implementations
  - Course project
- Java implementations interoperate with each other
  - C, Perl are not there yet...
- Less than 3 weeks worth of effort
- http://www.cs.columbia.edu/~angelos/JFK/

Some numbers

- Line counts (without crypto libraries)
  - Perl: 1742 lines
  - Java (1): 3025 lines
  - Java (2): 4023 lines (+1809 UI)
  - C: 3102 lines
- Message sizes
  - Msg1: 80 bytes
  - Msg2: 260 + certs
  - Msg3: 350 + certs
  - Msg4: 135 + certs

Identity protection

- Given DoS, security, active ID protection requirements, JFK is optimal
- If passive ID protection for both parties required, follow Hugo's suggestion of a 4-round SIGMA, which puts SIGMA in the format of JFK

Less-Bulky Joint (LBJ) protocol

I->R: Ni, g^x

R->I: Ni, Nr, GRPINFO, g^y,
HMAC{Hkr}(g^r, Ni, Nr)

I->R: g^x, g^y, Ni, Nr, HMAC{Hkr}(g^y, Ni, Nr),
C = E{Ke1}(IDi, sa, Sig(g^x, g^y, Ni, Nr,
GRPINFO)),
HMAC{Ka}('I', C)

R->I: D = E{Ke2}(IDr, sa', Sig(g^x, g^y, Ni, Nr)),
HMAC{Ka}('R', D)

Implementation

- JFK->LBJ implementation took less than a day
- 4054 lines of Java (+1804 UI)
- http://www.cs.columbia.edu/~angelos/LBJ/