OAKLEY Key Determination Protocol

author: Hilarie Orman
Computer Science Department
University of Arizona
May 1996
draft-ietf-ipsec-oakley-01.txt

OAKLEY: A *Family* of *Generic* Protocols for Key Distribution

1. Shared Key: common keying information state

- key name, key material
- identities of the (two) parties
- selected, agreed-upon algorithms (i.e.,, encryption, hashing, authentication.)

2. Key Distribution: key generation (e.g., by Diffie-Hellman modular exponentiation) as opposed to key transport (i.e., key wrapping in other keys)

3. Generic protocols: flexible selection of mutually agreeable algorithms for

- encryption (e.g., DES, IDEA)
    - o group representation (e.g., MODP, ECP, EC2N)
- hashing (e.g., MD5, SHA)
- authentication (e.g., RSA, DSS)

4. Family of Protocols: flexible selection of protocol

- modes:
    - o aggressive, conservative, main, quick, new_group
- services:
    - o anti-clogging (i.e., stateless cookie exchange)
    - o identity authentication and non-repudiation (via use of digital signatures)
    - o Perfect Forward Secrecy and Back Traffic Protection for keys
    - o identity privacy
    - o Perfect Forward Secrecy for private identities

OAKLEY: Protocol Properties

1. Clogging (Denial of Service) Avoidance

- key generation by modular exponentiation = > intensive computation
  => possibility of clogging (denial of service) attacks
- avoidance measure : use of "cookies"
  - o (stateless) cookie
    - receipient's net (IP) address and port no.
    - sender's net (IP) address and port no.
    - secret local value of owner (that expires periodically)
  - o cookie distribution to others => willingness to perform intensive computation upon return of own cookie
  - o must check receipt of own cookie against sender's net. address

2. Authentication is independent of key generation

- authentication validates binding <*identity, keying material*>
- authentication performed *before* key generation
- authentication performed by a different method (e.g., use of public-key certificates)

3. Key generation - keying material:

- Diffie-Hellman exponentials (with default or privately shared groups)
- secret, high-entropy values
- pre-distributed keys

4. Assumed Certificate Distribution and Validation Policies

OAKLEY: Examples of Protocol Rules (Conventions)

Message Parsing Rules

- "left-to-right parsing
- null field = multiprecision integer zero

Protocol Negotiation Rules

- Cookie rules:
  - o CKY-I = 0, CKY-R = 0 => cookie request / conservative prot.
  - o CKY-I = 0, CKY-R =/= 0 => cookie response / conservative prot.
  - o CKY-I = /= 0, CKY-R = 0 => aggressive, main prot.
- Null values returned by responder, w/ subsequent fields null => unacceptable offer
- GRP = null, g^x field = null => unacceptable accept first proposal in EHAO

Service Negotiation Rules

- GRP = null, nonce in g^x field => accept first proposal for no PFS for Keys and IDs
- GRP, g^x, IDs = non-null => request PFS for Keys but not IDs
- GRP, g^x = non-null, IDs = null => request PFS for Keys and IDs

State Transition Rules

- timeout / retransmission
- cookie verification
- signature, prf verification

Need: a *specification language* for families of generic protocols

Aggressive Example 1: PFS for Keys, Public IDs, Digital Signatures

Initiator **Responder**

CKY-I, 0, OK_KEYX, GRP, g^x, EHAO, NIDP, ID(I), ID(R), N_i , 0 ,

$\longrightarrow$

S { ID(I) | ID(R) | Ni | 0 | GRP | g^x | EHAO } K_i

CKY-R, CKY-I, OK_KEYX, GRP, g^y, EHAS, NIDP, ID(R), ID(I), N_r , N_i ,

$\longleftarrow$

S { ID(R) | ID(I) | N_r | N_i | GRP | g^y | EHAS } K_r

CKY-I, CKY-R, OK_KEYX, GRP, g^x, EHAO, NIDP, ID(I), ID(R), N_i , N_r ,

$\longrightarrow$

S { ID(I) | ID(R) | N_i | N_r | GRP | g^x | EHAO } K_i

KEYID = CKY-I | CKY-R **sKEYID = prf (N_i | N_r , g^{xy} | CKY-I | CKY-R)**

Notes: State Transition

- o first choices: H, A => state signature verification (certificates not shown)
- o cookie verification (e.g., Initiator: CKY-I vs. net. address of incoming message)
- o weak liveness check: ability to repeat other party’s cookie in order

Main Mode (ISAKMP: Identity Protection Exchange)

Initiator **Responder**

CKY-I, 0, OK_KEYX, GRP, EHAO $\longrightarrow$

$\longleftarrow$ CKY-R, CKY-I, OK_KEYX, GRP, EHAS

CKY-I, CKY-R, OK_KEYX, GRP, g^x, EHAO, NIDP, N_i, 0, $\longrightarrow$

$\longleftarrow$ CKY-R, CKY-I, OK_KEYX, GRP, g^y, EHAS, NIDP, N_r, N_i,

CKY-I, CKY-R, OK_KEYX, GRP, g^x, EHAO, IDP* , $\boxed{\text{ID(I), 0,}}$ $\longrightarrow$

$\boxed{S\ \{\ \text{CKY-I} \mid \text{CKY-R} \mid \text{ID(I)} \mid N_i \mid N_r \mid \text{GRP} \mid g^{xy} \mid \text{EHAO}\ \}\ K_i}$

$\longleftarrow$ CKY-R, CKY-I, OK_KEYX, GRP, g^y, EHAS, IDP*, $\boxed{\text{0, ID(R),}}$

$\boxed{S\ \{\ \text{CKY-R} \mid \text{CKY-I} \mid \text{ID(R)} \mid N_r \mid N_i \mid \text{GRP} \mid g^{xy} \mid \text{EHAS}\ \}\ K_r}$

where * key = prf { 0 , g^{xy} }

KEYID = CKY-I | CKY-R **sKEYID = prf (N_i | N_r , g^{xy} | CKY-I | CKY-R)**

Aggressive Example 2: PFS for Keys, Private IDs via Proxy R'

Initiator **Responder**

$\longrightarrow$ CKY-I , 0 , OK_KEYX , GRP , g^x , EHAO , NIDP , 0 , ID(R') ,
E { ID(I) , ID(R) , E { N_i } K_r } $K_{r'}$

$\longleftarrow$ CKY-R , CKY-I , OK_KEYX , GRP , g^y , EHAS , NIDP , ID(R') , 0 ,
E { ID(R) , ID(I) , N_r } K_i , prf { Kir , ID(R) | ID(I) | N_r | N_i | GRP | g^y | g^x }
where Kir = prf { 0 , N_i | N_r }

$\longrightarrow$ CKY-I , CKY-R , OK_KEYX , GRP , 0 , 0 , NIDP , 0 , ID(R') ,
prf { Kir , ID(I) | ID(R) | N_i | N_r | GRP | g^x | g^y }

KEYID = CKY-I | CKY-R **sKEYID = prf (N_i | N_r , g^{xy} | CKY-I | CKY-R)**

Notes: State Transition (proxy R' maintains state)
- o first choices: E, H (prf); certificates not shown
- o cookie verification (e.g., Initiator: CKY-I vs. net. address of incoming message)
- o weak liveness check: ability to repeat other party's cookie in order

Aggressive Example 3: Private IDs and W/O Diffie-Hellman

Initiator **Responder**

CKY-I , 0 , OK_KEYX , 0 , *0* , EHAO , NIDP , 0 , ID(R') , →
E { ID(I) , ID(R) , s Ki } Kr' , ~~prf { Kir , ID(R) | ID(I)~~

← CKY-R , CKY-I , OK_KEYX , 0 , *0*, EHAS , NIDP , ID(R') , 0 ,
E { ID(R) , ID(I) , sK_r } K_i , prf { Kir , ID(R) | ID(I) | sK_r | sK_i }

where Kir = prf { 0 , sK_i | sK_r }

CKY-I , CKY-R , OK_KEYX , 0 , *0*, EHAS , NIDP , 0 , ID(R') , →
prf { Kir , ID(I) | ID(R) | sK_i | sK_r }

KEYID = CKY-I | CKY-R **sKEYID = prf (K_{ir} , CKY-I | CKY-R)**

Notes: State Transition (proxy R' maintains state); replay detection ? sKEYID expiration

- o first choices: E, H (prf); certificates not shown
- o cookie verification (e.g., Initiator: CKY-I vs. net. address of incoming message)
- o weak liveness check: ability to repeat other party's cookie in order

Conservative Example: PFS for Keys and for Private IDs

Initiator **Responder**

0 , 0 , OK_KEYX →

← 0 , CKY-R , OK_KEYX

CKY-I , 0 , OK_KEYX , GRP , g^x , EHAO →

← CKY-R , CKY-I , OK_KEYX , GRP , g^y , EHAS

CKY-I , CKY-R , OK_KEYX , GRP , g^x , IDP* , $\boxed{ID(I) , ID(R) , E\{N_i\}K_r}$ →

where * key = prf { 0 , g^{xy} }

← CKY-R , CKY-I , OK_KEYX , GRP , 0 , 0 , IDP* , $\boxed{E\{N_r , N_i\}K_i , ID(R) , ID(I)}$

$\boxed{prf \{ N_r \mid N_i , GRP \mid g^{xy} \mid ID(R) , ID(I) \}}$

CKY-I , CKY-R , OK_KEYX , GRP , 0 , 0 , IDP*, →

$\boxed{prf \{ N_i \mid N_r , GRP \mid g^{xy} \mid ID(I) , ID(R) \}}$

KEYID = CKY-I | CKY-R **sKEYID = prf (N_i | N_r , g^{xy} | CKY-I | CKY-R)**

Notes: State Transition

- o first choices: E, H (prf) ; certificates not shown
- o cookie verification (e.g., Initiator: CKY-I vs. net. address of incoming message)
- o weak liveness check: ability to repeat other party's cookie in order

Quick_Mode Example

Initiator **Responder**

$$\xrightarrow{\text{KEYID},\ \text{INEWKEYRQ},\ N_i,\ \text{prf}(\text{sKEYID},\ N_i)}$$

$$\xleftarrow{\text{KEYID},\ \text{INEWKEYRS},\ N_r,\ \text{prf}(\text{sKEYID},\ 1 \mid N_r \mid N_i)}$$

$$\xrightarrow{\text{KEYID},\ \text{INEWKEYRP},\ 0,\ \text{prf}(\text{sKEYID},\ 0 \mid N_i \mid N_r)}$$

$$\mathbf{NKEYID} = \mathbf{N_i \mid N_r} \qquad \mathbf{sKEYID} = \mathbf{prf}(\mathbf{sKEYID},\ \mathbf{N_i \mid N_r})$$

Note: Extensive use of QUICK_MODE consumes the entropy of g^{xy} and, hence, QUICK MODE should be used sparingly

External_Key Example

Initiator **Responder**

KEYID , IEXTKEY , N_i , prf (sKEYID , N_i) →

← KEYID , IEXTKEY, N_r , prf (sKEYID , 1 | N_r | N_i)

KEYID , IEXTKEY, K_{ir} *xor* sNEWKEYID*, prf (K_{ir} , sNEWKEYID | N_i | N_r) →

where K_{ir} = prf (sKEYID, | N_i | N_r)

NKEYID = prf (sKEYID, 1 | N_i | N_r) **sNKEYID = K_{ir} *xor* (K_{ir} *xor* sNEWKEYID)**

Note: *length* (sKeyID) >= *length* (sNEWKEYID)

New_Group Selection

Initiator **Responder**

KEYID , INEWGRP , Desc(New_Group), Na ,

prf (sKEYID , Desc(New_Group) | Na)

KEYID , INEWGRPRS , Nb, Na ,

prf (sKEYID , Nb | Na , Desc(New_Group))

KEYID , INEWGRPACK ,

prf (sKEYID , Na | Nb , Desc(New_Group))

Notes:

- o Na , Nb =/= Ni , Nr or nonces used for current GRPID
- o Desc (G) = group descriptor
- o GRPID < - > Desc(NEWGRP) must be stored anew (with new IDs in KEYID)

Group Descriptors - 2 Examples

Group Type: *MODP* /* modular exponentiation group, mod P*/
Size of Field (in bits): $\lceil \log_2 P \rceil$ a 32-bit integer
Defining Prime P: a multi-precision integer
Generator G: a multi-precision integer $2 \le G \le P-2$
optional:
Largest prime factor of P-1 : the multiprecision integer Q
Strength of Group: a 32-bit integer (approx. the no. of key bits protected; $\log_2$ of workfactor)

Group Type: *ECP* /* elliptic curve group, mod P */
Size of Field (in bits): $\lceil \log_2 P \rceil$ a 32-bit integer
Defining Prime P: a multi-precision integer
Generator (X, Y): two multi-precision integers $(X, Y \le P)$
Parameters of the curve A, B: two multi-precision integers $(A, B \le P)$
optional:
Largest prime factor of group order : the multi-precision integer
Order of the group: a multi-precision integer
Strength of Group: a 32-bit integer (approx. the no. of key bits protected; $\log_2$ of workfactor)

elliptic curve equation: $Y^2 = X^3 + AX + B$