Security protocols

Lecture 7

Key Agreement

Diffie-Hellman Key Agreement

- Shared prime p and generator g of Z_p^* ; ra, rb – private random
  - 1. A -> B: $g^{ra} \bmod p$
  - 2. B -> A: $g^{rb} \bmod p$

    $K = (g^{ra})^{rb} \bmod p = (g^{rb})^{ra} \bmod p = g^{ra*rb} \bmod p$

  - intractability of Diffie-Hellman problem (DDH)

Diffie-Hellman Key Agreement

- Shared prime p and generator g of Z_p^* ; ra, rb – private random
  - 1. A -> B: $g^{ra} \bmod p$
  - 2. B -> A: $g^{rb} \bmod p$

    $K = (g^{ra})^{rb} \bmod p = (g^{rb})^{ra} \bmod p = g^{ra*rb} \bmod p$

  - intractability of Diffie-Hellman problem (DDH)
  - man-in-the-middle attack
    - 1. A -> C_B : $g^{ra} \bmod p$
    - 1.' C_A -> B : $g^{rc} \bmod p$
    - 2.' B -> C_A : $g^{rb} \bmod p$
    - 2. C_B -> A : $g^{rc} \bmod p$

      $K_{AC} = g^{ra*rc} \bmod p$; $K_{CB} = g^{rc*rb} \bmod p$

One-Pass Key Establishment with ElGamal encr.

- xa, xb – private DH long-term keys
- ya = $g^{xa} \bmod p$; yb = $g^{xb} \bmod p$

  - A -> B : $g^{ra} \bmod p$

  K = $g^{ra*xb} \bmod p = yb^{ra} \bmod p$

  freshness of $g^{ra} \bmod p$?

One-Pass Key Establishment with ElGamal encr.

- xa, xb – private DH long-term keys
- ya = $g^{xa} \bmod p$; yb = $g^{xb} \bmod p$

(Agnew, Mullin, Vanstone)

- A -> B : $g^{ra} \bmod p$, $s = t * yb^{ra} \bmod p$

  $t = s^* \text{ inv}(g^{ra*xb}) \bmod p$; $K = s^t \bmod p = (g^{xa*xb})^t \bmod p$

(Nyberg, Rueppel) t - time

- A -> B : $g^{ra} \bmod p$, $s = ra - xa*\#(t, yb^{ra} \bmod p) \bmod p$

  $K = g^{ra*xb} \bmod p$; verify $g^{ra} \bmod p \stackrel{?}{=} g^s ya^{\#(t, K)}$

(Lim, Lee)

1. A -> B: {Na}S
2. B -> A: {A, Nb}(S xor Na)
3. A -> B: {Na}(S xor Na) $S = f(g^{xa*xb} \bmod p)$; K = Na xor Nb

Diffie-Hellman Authenticated Key Exchange

- xa, xb – private long-term keys
- ya = g^{xa} mod p ; yb = g^{xb} mod p

MTI (Matsumoto-Takashima-Imai) (1986)

- A -> B: g^{ra} mod p
- B -> A: g^{rb} mod p

K = $(g^{rb})^{xa}$ yb^{ra} mod p = $(g^{ra})^{xb}$ ya^{rb} mod p = $(g^{ra*xb+rb*xa})$ mod p

KEA (Key Exchange Algorithm) – NSA (1998)

- A -> B: g^{ra} mod p
- B -> A: g^{rb} mod p

K = $(g^{rb})^{xa}$ + yb^{ra} mod p = $(g^{ra})^{xb}$ + ya^{rb} mod p

+ check g^{rb}, yb in subgroup G of order q q | (p-1)

Diffie-Hellman Authenticated Key Exchange

xa, xb – private long-term keys; $ya = g^{xa} \bmod p$; $yb = g^{xb} \bmod p$

Unified Model Protocol (NIST SP-800)

- A -> B: $g^{ra} \bmod p$
- B -> A: $g^{rb} \bmod p$

$K = ((g^{rb})^{ra} \bmod p \,||\, g^{xa*xb} \bmod p) = ((g^{ra})^{rb} \bmod p \,||\, g^{xa*xb} \bmod p)$

MQV protocol (IEEE P1363-2000)

- A -> B: $g^{ra} \bmod p$ $ta = (g^{ra} \bmod p)_w$ low w bits
- B -> A: $g^{rb} \bmod p$ $tb = (g^{rb} \bmod p)_w$

$sa = (ra + ta*xa) \bmod q$; $sb = (rb + tb*xb) \bmod q$ $q \mid (p-1)$

$K = (g^{rb}\, yb^{tb})^{sa} \bmod p = (g^{rb}\, ya^{ta})^{sb} \bmod p$

+ check -> forward secrecy

STS protocol (with explicit authentication)

- Station-to-Station protocol – forward secrecy
  (compromise of the long-term keys does not compromise the session keys established in previous protocol runs)

(Diffie) shared prime p and generator g of Z_p^*

- 1. A -> B : ta = g^{ra} mod p
- 2. B -> A : tb = g^{rb} mod p, {Sig_B(tb, ta)}K'
- 3. A -> B : {Sig_A(ta, tb)}K' verify

K = $(ta)^{rb}$ mod p = $(tb)^{ra}$ mod p K' = f(K) (ra, rb – ephemeral keys)

(Lowe's attack)

- 1. A -> C_B : Cert(A), Cert(B), ta = g^{ra} mod p
- 1'. C -> B : Cert(C), Cert(B), ta = g^{ra} mod p
- 2'. B -> C : Cert(B), Cert(C), tb = g^{rb} mod p, {Sig_B(tb, ta)}K'
- 2. C_B -> A : Cert(B), Cert(A), tb = g^{rb} mod p, {Sig_B(tb, ta)}K'
- 3. A -> C_B : Cert(A), Cert(B), {Sig_A(ta, tb)}K'

STS protocol

modified (avoid Lowe's attack)

- 1. A -> B : $ta = g^{ra} \bmod p$
- 2. B -> A : $tb = g^{rb} \bmod p$, $Sig_B(tb, ta, A)$
- 3. A -> B : $Sig_A(ta, tb, B)$ verify

$K = (ta)^{rb} \bmod p = (tb)^{ra} \bmod p$

STS protocol using MAC

- 1. A -> B : $ta = g^{ra} \bmod p$
- 2. B -> A : $tb = g^{rb} \bmod p$, $Sig_B(tb, ta)$, $[tb, ta]K'$
- 3. A -> B : $Sig_A(ta, tb)$, $[ta, tb]K'$

$K = (ta)^{rb} \bmod p = (tb)^{ra} \bmod p$ $K' = f(K)$

Oakley protocol (RFC 2412)

- aggressive mode
  - 1. A -> B: CK_A, ta = g^{ra} mod p, list, A, B, Na, Sig_A(A,B,Na,ta,list)
  - 2. B -> A: CK_B, CK_A, tb = g^{rb} mod p, algo, B, A, Nb, Na,
    Sig_B(B,A,Nb,Na,tb,ta,algo) verify
  - 3. A -> B: CK_A, CK_B, ta, algo, A, B, Na, Nb,
    Sig_A(A,B,Na,Nb,ta,tb,algo) verify
    Z = $(ta)^{rb}$ mod p = $(tb)^{ra}$ mod p K = [CK_A, CK_B, Z](Na,Nb)

- conservative mode
  - 1. A -> B: OK CK_A, CK_B - cookies (to mitigate DoS attacs)
  - 2. B -> A: CK_B
  - 3. A -> B : CK_A, CK_B, ta = g^{ra} mod p, list
  - 4. B -> A : CK_B, CK_A, tb = g^{rb} mod p, algo K_E = #(Z)
  - 5. A -> B : CK_A, CK_B, ta, {A, B, {|Na|}Kb}K_E K_M = #(Na, Nb)
  - 6. B -> A : CK_B, CK_A, { {|Nb, Na|}Ka, B, A, [B, A, tb, ta, algo]K_M }K_E
  - 7. A -> B : CK_A, CK_B, { [A, B, ta, tb, algo]K_M }K_E verify MAC
    Z = $(ta)^{rb}$ mod p = $(tb)^{ra}$ mod p K = [CK_A, CK_B, Z](Na,Nb)

Oakley protocol with encryption

- aggressive mode
  - 1. A -> B: CK_A, ta = g^{ra} mod p, list, B, $\{|A, B, Enc_B(Na)|\}_{KB'}$
  - 2. B -> A: CK_B, CK_A, tb = g^{rb} mod p, algo, $\{|B, A, Nb|\}_{KA}$,
    $[B,A,Nb,Na,tb,ta,algo]_K$
  - 3. A -> B: CK_A, CK_B, $[A, B, ta, tb, algo]_K$ verify MAC
    K = $[CK_A, CK_B, (ta)^{rb} \text{ mod } p = (tb)^{ra} \text{ mod } p](Na,Nb)$

SKEME protocol

- SKEME (for IPsec)
  - 1. A -> B: {|A, Na|}Kb , ta = g^{ra} mod p
  - 2. B -> A: {|Nb|}Ka, tb = g^{rb} mod p, [ta, tb, B, A]K’
  - 3. A -> B : [tb, ta, A, B]K’

    K = $(ta)^{rb}$ mod p = $(tb)^{ra}$ mod p ; K’ = #(Na, Nb)

- SKEME without DH
  - 1. A -> B: {|A, Na|}Kb , ra
  - 2. B -> A: {|Nb|}Ka, rb, [ra, rb, B, A]K’
  - 3. A -> B : [rb, ra, A, B]K’

    K = [[rb, ra, A, B]K’]K’ ; K’ = #(Na, Nb)

SKEME and IKE protocol

- IKE (Internet Key Exchange) main IPSec
  - 1. A -> B: CK_A, list CK_A, CK_B - cookies (to mitigate DoS attacs)
  - 2. B -> A: CK_B, algo
  - 3. A -> B: CK_A, CK_B, ta = g^{ra} mod p, Na |Na|, |Nb| ϵ <64,2048>
  - 4. B -> A: CK_A, CK_B, tb = g^{rb} mod p, Nb
  - 5. A -> B : CK_A, CK_B, {A, Sig_A([ta, tb, CK_A, CK_B, list, A]K')}K
  - 6. B -> A : CK_A, CK_B, {B, Sig_B([tb, ta, CK_B, CK_A, list, B]K')}K

Z = $(ta)^{rb}$ mod p = $(tb)^{ra}$ mod p K' = [Z]NaNb
K = SKDF(K') symmetric key derivation function

SIGMA and IKEv2 protocol

- SIGMA-I (Krawczyk, Canetti Sign-and-MAC)
  1. A -> B : ta = g^{ra} mod p
  2. B -> A : tb = g^{rb} mod p, {B, Sig_B(ta, tb), [B]K_M }K_E
  3. A -> B : {A, Sig_A(tb, ta), [A]K_M }K_E

  Z = $(ta)^{rb}$ mod p = $(tb)^{ra}$ mod p $K_M = f_M(Z)$; $K_E = f_E(Z)$

- IKEv2 (Internet Key Exchange) IPSec RFC 7296 (simplified)

  IKE_INIT
  - A -> B: list, ta = g^{ra} mod p, Na |Na|, |Nb| ϵ <64,2048>
  - B -> A: algo, tb = g^{rb} mod p, Nb

  IKE_AUTH
  - A -> B : {A, Sig_A(list, ta, Na, Nb, [A]K_M)}K_E verify MAC
  - B -> A : {B, Sig_B(algo, tb, Nb, [B]K_M)}K_E verify MAC

  Z = $(ta)^{rb}$ mod p = $(tb)^{ra}$ mod p; K_M = [Z]NaNb; K_E = f(K_M); K = SKDF(K_M)
  CK – on demand (DoS)

JFK – Just Fast Keying

ISO/IEC 11770-3 Key agreement

- 12 key agreement mechanisms
- analysed using te Scyther tool
- mechanism 11 (TLS handshake)
  - the RSA version has problems in authentication,
  - protocol does not provide forward secrecy

TLS – Transport Layer Security

- 1995 SSL (Netscape Communications)
- 1999 TLS 1.0 RFC 2246
- 2006 TLS 1.1 RFC 4346
- 2008 TLS 1.2 RFC 5246
- 2018 TLS 1.3 RFC 8446

- reliable transport protocol (over TCP)

protocol model for TLS

- 0. A -> B: A, Na, Sid, Pa
- 1. B -> A: Nb, Sid, Pb
- 2. B -> A: {|B, Kb|}inv(Ks)
  - (Ks := pubkey of the certification authority – this is a certificate)
- 3. A -> B: {|A, Ka|}inv(Ks)
- 4. A -> B: {|PMS|}Kb
- 5. A -> B: {|#(Nb,B,PMS)|}inv(Ka)
- 6. A -> B: {Finished}Keygen(A, Na, Nb, M)
  - where M = PRF(PMS,Na,Nb) (PseudoRandomFunction)
  - Finished = H(M,messages) for all messages 0 - 5
- 7. B -> A: {Finished}Keygen(B, Na, Nb, M)

Simplified TLS

- 1. A -> B: Na
- 2. B -> A: Nb PMK – Pre-master key from A
- 3. A -> B: {|PMK|}Kb, Sig_A(Mess_Seq1), {Mess_Seq2}K
- 4. B -> A: {Mess_Seq3}K K = [Na, Nb]PMK

Mess_Seq1 = #(Na, Nb, {|PMK|}Kb) ...

- simplified (based on Diffie-Hellman)
  - 1. A -> B: Na
  - 2. B -> A: Nb PMK = g^{a*b} mod p
  - 3. A -> B: Sig_A(Mess_Seq1), {Mess_Seq2}K
  - 4. B -> A: {Mess_Seq3}K K = [Na, Nb]PMK

TLS with Perfect Forward Secrecy

- TLS with forward secrecy
  - 1. A -> B: Na
  - 2. B -> A: Nb, g^{rb} mod p, Sig_B(Na, Nb, p, g, A))
  - 3. A -> B: g^{ra} mod p

  Z = g^{ra*rb} mod p delete ra, rb (ephemeral keys)

  MasterK = [Na, Nb]Z

  SessK = #(MasterK expansion, Na, Nb)

  ECDHE – ephemeral DH with elliptic curve cryptography

  PFS – perfect forware secrecy - previous session keys are not compromised even if the long term keys are

KEM key encapsulation mechanisms

- encryption of new random number
- public $K \in K_E$, private $K^{-1} \in K_D$
- encapsulation $Encap_K() = (c, k)$ k – new sym. key , $c \in C$
- decapsulation $Decap_{K-1}(c) = k$

  - A: $(c_A, k_A) = Encap_B()$
    A -> B: c_A
  - B: $k_A = Decap_B(c_A)$, $(c_B, k_B) = Encap_A()$
    B -> A: c_B
  - $K = k_A, k_B$

EKE – Encrypted Key Exchange

π – shared password

- A -> B : A , $\{ g^{ra} \bmod p \}\pi$
- B -> A : $\{ g^{rb} \bmod p \}\pi$, {Nb}K
- A -> B : {Na, Nb}K
- B -> A : {Na}K

ra, rb – ephemeral keys

$K = g^{ra * rb} \bmod p$

PAK – Password Authenticated Key exchange

π – shared password g – generator Z_p p-1=q*t

- A -> B : $(g^{ra} \bmod p) * H(A,B,\pi)^t = m$
  $K = g^{ra * rb} \bmod p$
- B -> A : $g^{rb} \bmod p$, $H'(A,B, m, g^{rb}, K)$
  K = ...
- A -> B : $H''(A,B, m, g^{rb}, K)$

SPEKE - Secure Password Exponential Key Exchange

π – shared password g – generator Z_p p-1=q*t
$P = \pi^2 \bmod p$

- A -> B : $P^{ra} \bmod p$

  $K = P^{ra * rb} \bmod p$
- B -> A : $P^{rb} \bmod p$, $H(P^{ra}, P^{rb}, K, \pi)$

  K = ...
- A -> B : $H'(P^{ra}, P^{rb}, K, \pi)$

$Z = H''(K)$

RSA-based protocols

π – shared password, $n = p*q$, $d*e \bmod (p-1)*(q-1) = 1$

- A -> B : A , n, {e}π
- B -> A : $\{K^e \bmod n\}\pi$