Cryptographic Escalatory Protocols

Masaryk University in Brno

Faculty of Informatics

Jan Krhovják

Introduction & Terminology

- Key establishment
  - Diffie-Hellman (DH) protocol
    - β – prime; recommended safe prime $\beta=2\gamma+1$, where γ is prime
    - α – generator of multiplicative group Z_β^* (called DH base)
    - We will assume that α, β are known to all: A=user/client; B=server;
    - A->B: $\alpha^x \bmod \beta$; A<-B: $\alpha^y \bmod \beta$; (will be referred as DH values)
    - Key K= $\alpha^{xy} \bmod \beta$ based on secrets of both sides
  - Pure DH is vulnerable to "man in the middle" attack
- Mutual authentication
  - DH and exchange of authentication signatures (basic STS)
    - A->B: $\alpha^x \bmod \beta$; A<-B: $\alpha^y \bmod \beta$, $E_K(s_B(\alpha^y,\alpha^x))$; A->B: $E_K(s_A(\alpha^x,\alpha^y))$
    - A must know B's authentic public key, and vice versa
- Identifiers of the sides A and B will be omitted

Encrypted Key Exchange (EKE)

- **Simplified description**
  - Side A and side B share secret password P
  - A->B: $E_P(VA)$;
    - The key pair with public key VA is generated by side A
    - VA is encrypted by using symmetric cryptosystem and password P
  - A<-B: $E_P(E_{VA}(K))$;
    - P is used to obtain VA; then secret session key K is generated
    - K is asymmetrically encrypted by VA and symmetrically by P
  - A->B: $E_K(R_A)$; A<-B: $E_K(R_A, R_B)$; A->B: $E_K(R_B)$;
    - Necessary to resist replay attacks and to check the key correctness
    - R_X can be random number or timestamp
- **EKE works very well with DH, but can work also with cryptosystems as RSA or ElGamal**

EKE:
Problems and Solutions I

- The message encrypted by password P must be indistinguishable from a random number
  - To prevent off-line brute force (and dictionary) attack
- EKE implementation with RSA
  - How to encode RSA public key (n,e) to be indistinguishable?
    - It is impossible – you can always test if n have a prime factors
    - Only e can be encrypted by password P
      - The value e is ever odd – $gcd(e,\phi(n))=1$
      - Solution => add binary 1 to last bit with probability 1/2
- EKE implementation with ElGamal
  - No such problem of encoding public key
    - Public keys are generated as $\alpha^r \mod \beta$
    - Uniquely distributed in interval $[1, \beta-1]$

EKE:
Problems and Solutions II

- Is the odd VA=e really problem? How is it dangerous?
- The attacker can try to decrypt $E_P(VA)$
  - He can use all possible values P' – i.e. $D_{P'}(E_P(VA))$
    - The even result => he can rule out half of candidate P'
  - Each session uses different public key
    - Next trial decryptions will exclude different values
  - The decrease in keyspace is logarithmic
  - This is so called "partition attack"
- ElGamal may allow minimal partition
  - The numbers mod β are encrypted (β can be encoded to n bits)
    - If trial decryption yields the value in $[\beta, 2^n-1]$ => partition is allowed
      - β close to 2^n-1 exclude only few candidates => attack ineffective
      - β close to 2^{n-1} => effective "partition attack"

EKE: Problems and Solutions III

- Encryption of number by symmetric cryptosystem with larger block size
  - Padding zero bits => partition attack
  - Padding bits should be random

- Solution of last two problems in one operation
  - Basic assumption
    - Integers to be encrypted are mod β
    - Block size is m bits, where $2^m > \beta$
  - $x = \lfloor 2^m/\beta \rfloor$ is number of times our interval fits into block size
    - Choose $j \in [0, x{-}1]$ and add $j\beta$ to input value (non modulo arithmetic)
    - If input is less then $2^m{-}x\beta$ => choose $j \in [0, x]$
    - The recipient knows P and β => easy to obtain correct initial value

Diffie-Hellman Encrypted Key Exchange (DHEKE)

- **DHEKE is EKE based on DH protocol**
  - DH values are encrypted by password P
    - $E_P(\alpha^x \bmod \beta)$ and $E_P(\alpha^y \bmod \beta)$
  - No separate transmission of K is needed
    - K is derived from a value $\alpha^{xy} \bmod \beta$

- **A complete protocol (base for complex protocols)**
  - A->B: $E_P(\alpha^x \bmod \beta)$;
  - A<-B: $E_P(\alpha^y \bmod \beta)$, $E_K(R_B)$;
  - A->B: $E_K(R_A, R_B)$;
  - A<-B: $E_K(R_A)$;

- **Size of β protected by password may be shorter ☺**

Augmented Encrypted Key Exchange (AEKE)

- In fact augmented DHEKE
  - Verifier-based: server does not store cleartext passwords
  - The attacker who obtain one-way encrypted password file cannot mimic the user/client to the server
    - He still can mimic the server to the user/client
    - He still can mount attack against the one-way encrypted passwords
  - The key pair (public key VA, private key SA) is generated from P
    - Server stores VA as verifier; user/client can generate key pair from P
- A complete protocol
  - A->B: $E_{VA}(\alpha^x \bmod \beta)$; A<-B: $E_{VA}(\alpha^y \bmod \beta), E_K(R_B)$;
  - A->B: $E_K(R_A, R_B)$;   A<-B: $E_K(R_A)$;
  - A->B: $E_K(E_{SA}(K))$;
- (A)EKE provides good replacement for Interlock Protocol

Minimal Encrypted Key Exchange (MEKE)

- **Refinement of DHEKE**
  - More efficient version
  - Instead traditional $K=\alpha^{xy} \bmod \beta$ is set $K=h(\alpha^{xy} \bmod \beta)$
    - Necessary mainly for pure EKE
      - Disclosure of session key must not allow attack on P

- **A complete protocol**
  - A->B: $E_P(\alpha^x \bmod \beta)$;
  - A<-B: $\alpha^y \bmod \beta$, $E_K(R_B)$;
  - A->B: $E_K(f(R_B))$;
    - f can be one-way hash function or encryption function with key K

Dual-workfactor Encrypted Key Exchange (DWEKE)

- Protocols are susceptible to "password chaining" attacks
  - User typically shares a password with trusted server
    - P is used to protect normal auth. and key-distribution messages
    - P is also used to protect messages with new selected password
    - If attacker knows the some password => he can decrypt all new passwords and all subsequent communication
- Protocol outline
  - Trade-off between security and efficiency
    - In classical DHEKE can be size of β protected by P quite short
      - Revealed P => DH exponents can be easily solved => revealed K
      - Long β => K will remain secure (at the cost of decreased efficiency)
    - Basic idea of DWEKE
      - Critical password exchange is protected by long modulus β
      - Normal day-to-day messages are protected by short modulus β

Simple Password Encrypted Key Exchange (SPEKE)

- **Very similar to DHEKE**
  - SPEKE uses function f instead of generator α
    - f has only one parameter – shared password P
    - The result of this function is some base for exponentiation
      - Not necessary generator of the whole group

- **First part of protocol**
  - A->B: $f(P)^x \bmod \beta$; A<-B: $f(P)^y \bmod \beta$;
  - $K=f(P)^{xy} \bmod \beta$ or $K=h(f(P)^{xy} \bmod \beta)$

- **Verification stage**
  - Based on random numbers
    - A->B: $E_K(R_A)$; A<-B: $E_K(R_A, R_B)$; A->B: $E_K(R_B)$;
  - Based on hash functions
    - A->B: $h(h(K))$; A<-B: $h(K)$;

SPEKE: Practical Issues

- Selection of f(P) in the case that $\beta=2\gamma+1$, where γ is prime
  - Recommended $f(P)=P^{(\beta-1)/\gamma} \bmod \beta = P^2 \bmod \beta$ (to have order γ)
  - Insecure is $f(P)=2^P \bmod \beta$

- No encryption in the first part of SPEKE
  - The attacker can reduce the keyspace
  - "Subgroup confinement" attack – version "man in the middle"
    - δ is a small prime factor of $\beta-1$
    - The exchanged values can be raised to a power $(\beta-1)/\delta$
    - This convert them to generators of small group of order δ
    - K can be guessed with probability $1/\delta$ or found by brute force attack
    - Safe primes only reduces the number of small subgroups
      - The key should be tested to be not an element of such subgroup

ASPEKE, BSPEKE, BEKE

- All of these protocols are verifier-based
- ASPEKE
  - ○ Straightforward application of technique from AEKE to SPEKE
- BSPEKE & BEKE
  - ○ Instead public key techniques from AEKE uses a second DH exchange
  - ○ It is important to prove user's/client's knowledge of P
- Is it really necessary? Yes …
  - ○ First phase of these protocols is based on DHEKE or SPEKE
    - The difference is only in using verifier instead password and in required derivation of session key by using hash function
    - Direct knowledge of password isn't proved

Secure Remote Password Protocol (SRP)

- **SRP is new type of verifier-based protocol**
  - No use of encryption => simple design (remember EKE)
- **Basic settings**
  - β = large safe prime, α = generator of Z_β^*, P = plaintext password
  - A knows the password P, x = H(P,S) is based on salt send by B
  - B knows the verifier v = α^x mod β and S = salt
- **Protocol core**
  - A generates random a and B random b, u (all from [2, β–1])
  - A->B: C=α^a mod β
  - A<-B: u, D=v+α^b mod β
  - A and B compute common K'=α^{ab+bux} mod β and final K=h(K')
    - A: K'=(D–α^x)$^{\alpha+ux}$ B: K'=(Cv^u)^b
- **Rest of the protocol is the verification stage**

Password Derived Moduli (PDM)

- Based on DH key establishment
  - PDM is also verifier based protocol
  - PDM performance: expensive at the client; efficient at the server
- Password P is used to crate safe prime modulo β
  - P can be a seed of PRNG used to search appropriate β
  - Side B don't know P => β is stored as verifier
  - DH base $\alpha=2$, nonce is denoted as R
- A complete protocol
  - A->B: $2^x \bmod \beta$
  - A<-B: $2^y \bmod \beta$, R, $h(2^{xy} \bmod \beta)$
  - A->B: $h(R, 2^{xy} \bmod \beta)$
- Session key K can be derived from $2^{xy} \bmod \beta$

PDM: Avoiding Information Leakage

- Attacker can use trial passwords P' to derive β
  - P' can be ruled out <=> either DH value is greater then β
  - Can be solved by discarding x (or y) if $2^x \bmod \beta$ (or $2^y \bmod \beta$) is greater then smallest possible β
  - Due to efficiency should be probability of discarding x or y low

- Solution
  - β will be from narrow range close to power of 2
    - Can be done by fixing high order 64 bits to 1
    - If prime have 700 bits, the rest 636 bits will be enough to choose β
    - Moreover, β is only from the fraction of 700-bit space
    - Probability that $2^x \bmod \beta$ or $2^y \bmod \beta$ will be greater then smallest possible β will be only $1/2^{64}$

Conclusion

- Escalatory protocols = very elegant authentication and key establishment methods/techniques
  - Based on low entropy data (e.g. PINs or passwords)
  - Some protocols can use also EC instead DH group
- Problem of many such protocols
  - Formal proofs of their security and correctness don't exist
  - Several protocols are also patented
- Standardization of these protocols is in progress
  - IEEE P1363.2 draft version 20

- Other well-known escalatory protocols
  - AMP, AuthA, OKE, PAK, S3P, SNAPI …