CO 487 - Notes

Bilal Khan (b54khan)
bilal.khan@student.uwaterloo.ca

December 9, 2024

Contents

1 Symmetric Encryption 1

2 Hash Functions 3

3 MACs 3

4 PRNGs 4

5 Passwords 4

6 Public Key Cryptography 4

7 Diffie-Hellman 5

8 Signatures 7

9 Key Management 8

1 Symmetric Encryption

Kerckhoff's principle The adversary knows everything about the SKES, except the particular key k chosen by Alice and Bob. (Avoid security by obscurity!!)

Adversary's Goals in Breaking Symmetric Key Encryption:

1. Recover the secret key.
2. Systematically recover plaintext from ciphertext (without necessarily learning the secret key).
3. Learn some partial information about the plaintext from the ciphertext (other than its length).

Security Definitions

- If the adversary can achieve goals 1 or 2, the SKES is said to be *totally insecure* (or *totally broken*).

- If the adversary cannot learn any partial information about the plaintext from the ciphertext (except possibly its length), the SKES is said to be *semantically secure*.
- Hiding length information is very hard. This topic falls under the heading of *traffic analysis*.

Security by obscurity The adversary knows everything about the SKES, except the particular key k chosen by Alice and Bob.

Passive attacks

- Ciphertext-only attack: The adversary only sees some encrypted ciphertexts.
- Known-plaintext attack: The adversary also knows some plaintext and the corresponding ciphertext.

Active attacks:

- Chosen-plaintext attack: The adversary can also choose some plaintext(s) and obtain the corresponding ciphertext(s).
- Chosen-ciphertext attack: The adversary can also choose some ciphertext(s) and obtain the corresponding plaintext(s). Includes the powers of chosen-plaintext attack.

Limits on Computational Powers

- Information-theoretic security: The adversary has infinite computational resources.
- Complexity-theoretic security: The adversary is a "polynomial-time Turing machine".
- Computational security: The adversary has X number of real computers/workstations/supercomputers. ("computationally bounded"). Equivalently, the adversary can do X basic operations, e.g., CPU cycles.

Pseudorandom bit generator (PRBG) is a deterministic algorithm that takes as input a short random seed, and outputs a longer pseudorandom sequence, also known as a keystream.

Indistinguishability The output sequence should be indistinguishable from a random sequence.

Unpredictability If an adversary knows a portion c_1 of ciphertext and the corresponding plaintext m_1, then she can easily find the corresponding portion $k_1 = c_1 \oplus m_1$ of the output sequence. Thus, given portions of the output sequence, it should be infeasible to learn any information about the rest of the output sequence.

Stream cipher period For a good stream cipher we need that the period of the keystream output sequence is larger than the length of the plaintext.

Stream ciphers A5/1, RC4, Salsa20, ChaCha20, main idea: use a PRBG to generate a keystream, and then XOR the keystream with the plaintext to get the ciphertext.

Security

- Diffusion: each ciphertext bit should depend on all plaintext and all key bits.
- Confusion: the relationship between key bits, plaintext bits, and ciphertext bits should be complicated.
- Cascade or avalanche effect: changing one bit of plaintext or key should change each bit of ciphertext with probability about 50%
- Key length: should be small, but large enough to preclude exhaustive key search.

AES/DES substitution-permutation network. round key xor-ed into the state. DES uses a Feistel network.

ECB mode encrypt each block independently. violates semantic security. padding required.

CBC mode $C_i = E(K, C_{i-1} \oplus P_i), C_0 = IV$. padding required.

CFB mode $C_i = E(K, C_{i-1}) \oplus P_i, C_0 = IV$. semantically secure.

OFB mode $O_i = E(K, O_{i-1}), C_i = O_i \oplus P_i, O_0 = IV$. xor the plaintext into the ciphetext "stream".

2 Hash Functions

MD4, MD5, SHA-1, SHA-2, SHA-3

Preimage resistance Hard to invert given just an output.

2nd preimage resistance Hard to find a second input that has the same hash value as a given first input.

Collision resistance Hard to find two different inputs that have the same hash values.

3 MACs

provides confidentiality (semantic security) and integrity (EUF-CMA).

HMAC $HMAC(K, m) = H((K \oplus opad) \parallel H((K \oplus ipad) \parallel m))$, secure against length extension attacks.

MAC-then-encrypt / MAC-and-encrypt : Insecure, no guarantee MAC doesn't leak plaintext bits

Encrypt-then-MAC : Secure, but requires padding.

4 PRNGs

Indistinguishability.

$$PRG : \{0,1\}^\lambda \to \{0,1\}^\ell$$
$$PRF : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\ell$$
$$KDF : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\ell$$

Pseudorandom generator deterministic function that takes as input a random seed $k \in \{0,1\}^\lambda$ and outputs a random-looking binary string of length ℓ.

Expanding a strong short key into a long key (e.g., stream cipher): $HMAC_k(1), HMAC_k(2), HMAC_k(3), \ldots$

Pseudorandom function deterministic function that takes as input a random seed $k \in \{0,1\}^\lambda$ and a (non-secret) label in $\{0,1\}^*$ and outputs a random-looking binary string of length ℓ.

Deriving many keys from a single key: $HMAC_k(label)$

Key derivation function deterministic function that takes as input a random seed $k \in \{0,1\}^\lambda$ and a (non-secret) label in $\{0,1\}^*$ and outputs a random-looking binary string of length ℓ.

Turn longer non-uniform keys into shorter uniform keys $HMAC_{label}(k)$

5 Passwords

- Knowledge-Based (Something you know)
- Object-Based (Something you have)
- ID-Based (Something you are)
- Location-based (Somewhere you are)

Rainbow tables Hash begin/end of chain instead of every hash.

Salting protect against rainbow tables.

6 Public Key Cryptography

allows for non-repudiation.

RSA

$$E((n,e),m) = m^e \mod n$$
$$D((n,d),c) = c^d \mod n$$

1. Choose random primes p and q with $\log_2 p \approx \log_2 q \approx \ell/2$.
2. Compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$.
3. Choose an integer e with $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$.

4. Compute $d = e^{-1}$ in $\mathbb{Z}_{\varphi(n)}$.

   (a) Use the Extended Euclidean Algorithm to compute d.
   If the Extended Euclidean Algorithm succeeds, then you are guaranteed that $\gcd(e, \varphi(n)) = 1$.

   Note: $de \equiv 1 \pmod{\varphi(n)}$ by definition of e^{-1}.

5. The public key is (n, e).

6. The private key is (n, d).

Fermat's Little Theorem

Theorem 6.1 (Fermat's Little Theorem)**.** *Let p be a prime. For all integers a, it holds that*

$$a^p \equiv a \pmod{p}$$

Moreover, if a is coprime to p, then

$$a^{p-1} \equiv 1 \pmod{p}$$

Modular operations addition/subtraction is $O(l)$, multiplication/inversion is $O(l^2)$, exponentiation is $O(l^3)$.

Square-and-multiply $a^b \bmod n$, $\log n$ iterations of modular multiplications that each take $O(l^2)$ time, so $O(l^3)$ time total.

7 Diffie-Hellman

$$\mathbb{Z}_n^* = \{1, 2, \ldots, n-1 : \gcd(a, n) = 1\}$$

Order of $x \in \mathbb{Z}_n^*$ Smallest positive integer k such that $x^k \equiv 1 \pmod{n}$.

Generator of $\mathbb{Z}_n^*$ An element $g \in \mathbb{Z}_n^*$ such that every $y \in \mathbb{Z}_n^*$ can be written as g^k for some $k \in \mathbb{Z}$.

$\mathbb{Z}_p^*$ for any prime p, every generator has order $p - 1$.

Group a set G with a binary operation $\cdot$ that satisfies:

- Associativity: for all $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- Identity: there exists an element $\mathcal{O} \in G$ such that for all $a \in G$, $a \cdot \mathcal{O} = \mathcal{O} \cdot a = a$
- Inverses: for all $a \in G$, there exists an element $a^{-1} \in G$ such that $a \cdot a^{-1} = a^{-1} \cdot a = \mathcal{O}$

Cyclic group if it can be generated by a single element

Diffie-Hellman key exchange has forward secrecy.

- Alice and Bob agree on a group G and a generator g of G
- Alice chooses a random integer $a \in \mathbb{Z}_n^*$ and computes $g^a \mod p$
- Bob chooses a random integer $b \in \mathbb{Z}_n^*$ and computes $g^b \mod p$
- Alice and Bob exchange $g^a \mod p$ and $g^b \mod p$
- Alice computes $(g^b)^a \mod p$
- Bob computes $(g^a)^b \mod p$
- Alice and Bob use $g^{ab} \mod p$ as their shared secret key

CDH $g^{ab} \mod p$ is hard to compute given g, $g^a \mod p$, and $g^b \mod p$.

DDH hard to distinguish between $g^{ab} \mod p$ and a random element of $\mathbb{Z}_p^*$.

DLOG given g and $g^a \mod p$, it is hard to compute a.

ElGamal

- Publish one half of the Diffie-Hellman key exchange as the public key.
- Include the other half of the key exchange as the first half of the ciphertext.
- Encrypt the plaintext under the shared secret key as the second half of the ciphertext
- Alice and Bob agree on a large prime p and a $g \in \mathbb{Z}_p^*$ of large prime order q
- Choose a random $x \in \mathbb{Z}_q^*$
- Set $k_{\text{pubkey}} = g^x \mod p$
- Set $k_{\text{privkey}} = x$
- To encrypt a message $m \in \mathbb{Z}_p^*$, choose a random $r \in \mathbb{Z}_q^*$
- $E(k_{\text{pubkey}}, m) = (g^r \mod p, m \cdot k_{\text{pubkey}}^r \mod p)$
- To decrypt a ciphertext (c_1, c_2), compute $m = c_2 \cdot (c_1^x)^{-1} \mod p$

Attack types

Passive attacks

- Key-only attack: The adversary is given the public key(s). We always assume the adversary has the public key(s).
- Chosen-plaintext attack: The adversary can choose some plaintext(s) and obtain the corresponding ciphertext(s). Equivalent to a key-only attack.

Active attacks

- Non-adaptive chosen-ciphertext attack: The adversary can choose some ciphertext(s) and obtain the corresponding plaintext(s).
- (Adaptive) chosen-ciphertext attack: Same as above, except the adversary can also iteratively choose which ciphertexts to decrypt, based on the results of previous queries

Totally insecure if the adversary can obtain the private key.

One-way if the adversary cannot decrypt a given ciphertext.

Semantically secure if the adversary cannot learn any partial information about a message.

RSA security Hard if integer factorization is hard. Not semantically secure since its deterministic. No deterministic encryption is semantically secure, but randomized encryption is not sufficient for semantic security.

Subexponential time representated as a pair of $L_n[\alpha, c]$ where the value of α determines the hardness.

Shor's algorithm $O(\log^2 n)$ gates to factor.

Diffie-Hellman Integrated Encryption Scheme (DHIES) elgamal encryption with a MAC.

8 Signatures

RSA malleability given $c = m^e \mod n$ for an unknown m, for any $x \in \mathbb{Z}_n^*$, we can construct $c' = (x^e \cdot c) \mod n = (xm)^e \mod n$. Selective forgery under a chosen message attack.

Full-domain hash RSA (FDH-RSA) compute a signature on the hashed message. Hash function needs to be collision/preimage/second preimage resistant.

Digital Signature Algorithm (DSA)

- Setup:
  - A prime p, a prime q dividing $p-1$, an element $g \in \mathbb{Z}_p^*$ of order q, a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$
- Key generation:
  - Choose $\alpha \in_R \mathbb{Z}_q^*$ at random. Return $(k_{\text{pubkey}}, k_{\text{privkey}}) = (g^\alpha \mod p, \alpha)$
- Signing: To sign a message $m \in \{0,1\}^*$,
  - Choose $k \in_R \mathbb{Z}_q^*$ at random
  - Calculate $r = (g^k \mod p) \mod q$ and $s = \frac{H(m)+\alpha r}{k} \mod q$

- Repeat if k, r, or s are zero. Otherwise, return signature $\sigma = (r, s)$

- Verification: Given $k_{\text{pubkey}} = g^{\alpha}$, m, and $\sigma = (r, s)$,
  - Check $0 < r < q$ and $0 < s < q$ and $(g^{\frac{H(m)}{s}} g^{\frac{\alpha r}{s}} \mod p) \mod q = r$

Elliptic Curve Digital Signature Algorithm (ECDSA)

- Setup:
  - A prime p, a prime q, an elliptic curve E over $\mathbb{Z}_p$ of cardinality $|E| = q$, a generator $P \in E$ of order q, and a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$.
- Key generation:
  - Choose $\alpha \in_R \mathbb{Z}_q^*$ at random.
  - $(k_{\text{pubkey}}, k_{\text{privkey}}) = (\alpha P, \alpha)$
- Signing: To sign a message $m \in \{0,1\}^*$,
  - Choose $k \in_R \mathbb{Z}_q^*$ at random
  - Calculate $r = x_k P \mod q$, and $s = \frac{H(m)+\alpha r}{k} \mod q$
  - Repeat if k, r, or s are zero.
  - The signature is $\sigma = (r, s)$.
- Verification: Given αP, m, and (r, s),
  - Check $0 < r < q$ and $0 < s < q$
  - Check that the x-coordinate of $\frac{H(m)}{s} P + \frac{r}{s}(\alpha P)$ is congruent to r modulo q.

9 Key Management

Certificates subject identity, subject public key, issuer signature, expiration date etc.

TLS handshake v1.3 generates a new ephemeral ECDSA key pair for each connection, signed by the server's certificate. Client always verifies server's key w its certificate. Optional client-to-server authentication.

Bitcoin Signs transaction (including prev transaction id) using ECDSA key pair. proof of work by finding a hash value of sha-256 with 47 leading zeros.