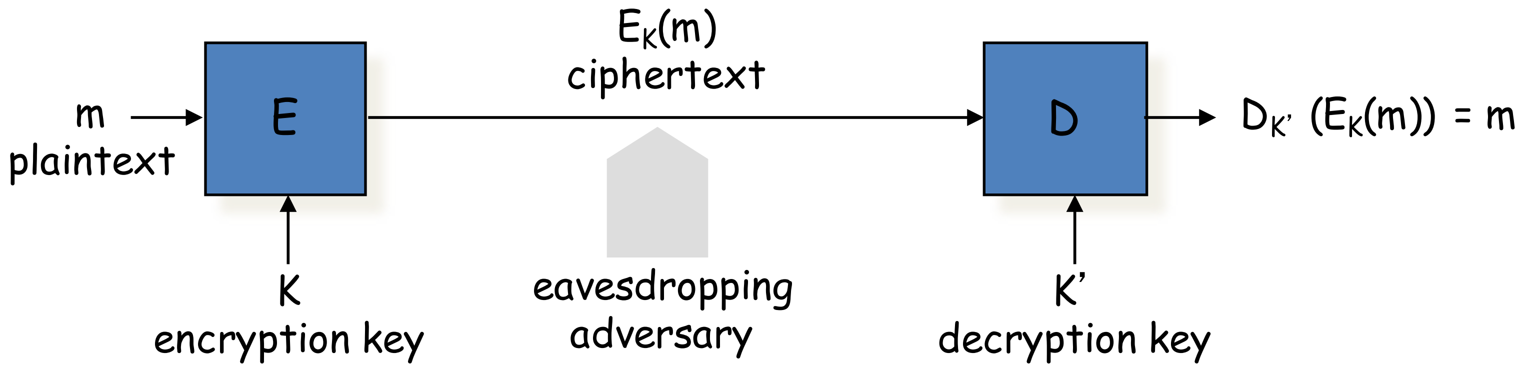
18734 Recitation

Cryptography

Tools: Black Box Auditing, The QII tool

Classical model of encryption



- Goal of the adversary:
  - to systematically recover plaintexts from ciphertexts
  - to deduce the (decryption) key

Slide: Jean-Pierre Hubaux

Basic Cryptographic Concepts

- Encryption scheme (symmetric and public key)
- Signature scheme
- Message authentication code
- Hash function

- A network protocol like SSL is built using these primitives

Symmetric Encryption Scheme

- *Key generation* algorithm
  - Produces a key that is used for encryption and decryption

- Algorithm to *encrypt* a message
- Algorithm to *decrypt* a ciphertext
- Correctness:
  - Decrypting a ciphertext obtained by encrypting message m with the corresponding key k returns m

$$dec(enc(m,k),k) = m$$

- (Symbolic) Security:
  - A ciphertext cannot be decrypted without access to the key

Example Symmetric Encryption Scheme: One time pad (OTP)

- *Key generation* algorithm
  - generate random bits
- Algorithm to *encrypt* a message
  - enc(m,k) = $m \oplus k$
- Algorithm to *decrypt* a ciphertext
  - dec(c,k) = $c \oplus k$
- Correctness:

$$dec(enc(m,k),k) = dec(m \oplus k, k) = (m \oplus k) \oplus k = m$$

Example Symmetric Encryption Scheme: One time pad (OTP)

- *Key generation* algorithm
  - generate random bits
- Algorithm to *encrypt* a message
  - $enc(m,k) = m \oplus k$
- Algorithm to *decrypt* a ciphertext
  - $dec(c,k) = c \oplus k$
- (Symbolic) security: A ciphertext cannot be decrypted without access to the private decryption key
  - Entropy(m) = Entropy(m given c)

Public-Key Encryption Scheme

- *Key generation* algorithm
  - Produces private decryption & public encryption key pair
- Algorithm to *encrypt* a message
- Algorithm to *decrypt* a ciphertext
- Correctness:
  - Decrypting a ciphertext obtained by encrypting message m with the corresponding encrytion key returns m

$$dec(enc(m, pk(A)), sk(A)) = m$$

- (Symbolic) Security:
  - A ciphertext cannot be decrypted without access to the private decryption key

Example Public-Key Encryption Scheme

- *Key generation* algorithm
  - Generate public key: e, secret key: d
  - s.t. ed = 1 (mod n)
- Algorithm to *encrypt* a message
  - $enc(m,e) = m^e \bmod n$
- Algorithm to *decrypt* a ciphertext
  - $dec(c,d) = c^d \bmod n$
- Correctness:
  - $dec(enc(m, e), d) = dec(m^e \bmod n, d) = m^{ed} \bmod n = m.$
- (Symbolic) Security:
  - A ciphertext cannot be decrypted without access to the private decryption key.

RSA Signatures

- Key Generation
  - Generate primes p, q; $N = pq$
  - Public key = e; private key = d   s.t.
    $ed = 1 \bmod (p-1)(q-1)$
- Sign
  - $C = M^d \bmod N$
- Verify
  - Check $M \bmod N = C^e \bmod N$
  - Note $C^e \bmod N = M^{ed} \bmod N = M \bmod N$

Signature Scheme

- *Key generation* algorithm
  - Produces private signing & public verification key pair
- Algorithm to *sign* data
- Algorithm to *verify* signature
- Correctness:
  - Message signed with a signing key verifies with the corresponding verification key

$$verify(m, sign(m,d), e) = ok$$

- Security:
  - A signature cannot be produced without access to the private signing key

Signature Scheme

- *Key generation* algorithm
  - private signing & public verification key pair (e, d=1/e)
- Algorithm to *sign* data
  - sign(m, e) = m^e
- Algorithm to *verify* signature
  - verify(m, c, d) = return *ok* iff m == c^d
- Correctness:
  - *verify(m,sign(m,d),e)* = *ok*. Satisfied?
- Security:
  - A signature cannot be produced without access to the private signing key. Satisfied?

Message Authentication Code (MAC)

- *Key generation* algorithm
  - Produces a key
- Algorithm to *mac a* message
- Algorithm to *verify* a mac on a message
- Correctness:
  - Message mac-ed with key verifies with the same key

$$verify(k, m, mac(k,m)) = ok$$

- Security:
  - A MAC cannot be produced without access to the key

Similar to signature, but uses symmetric key

What property does a signature have, but a MAC does not?

Hash Functions

- Algorithm to *hash a* message m to a fixed length output *hash(m)*
- Security (Collision resistance)
- Given hash function *hash: X* $\rightarrow$ *Y*, cannot find a collision, i.e. x, x' $\in$ X s.t. x ≠ x' and hash(x) = hash(x')
  - Hash(password) ≠ Hash(pa$sword)

Hash Functions

- Algorithm to *hash a* message m to a fixed length output *hash(m)*
  - *hash(m) = m % 10, where m is an integer*

- Security (Collision resistance)

  Given hash function *hash:* $X \rightarrow Y$, cannot find a collision, i.e. x, x' $\in$ X s.t. x ≠ x' and hash(x) = hash(x').   Satisfied?

SHA1, SHA3, MD5, etc.