CNT4406/5412 Network Security

Introduction to Cryptography

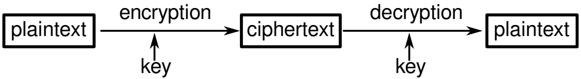
Zhi Wang

Florida State University

Fall 2014

What is Cryptography

- Mangling data into unintelligible form in a manner allowing lossless unmangling
  - ➠ usually one-to-one in size (unlike compression)
  - ➠ RSA is different
  - ➠ other services: integrity check and authentication



Cryptography Caveats

- We normally cannot prove a cipher is secure, instead we assume it is secure until unproven
  - ➠ arms race of cryptographers and cryptanalysts (Fred) improves it
  - ➠ cryptography systems usually have an algorithm and a key
  - ➠ publish the algorithm while keeping the key secret

Fundamental Tenet of Cryptography

If lots of smart people have failed to solve a problem, then it probably won't be solved (soon).

Computational Difficulty

- Algorithm should be efficient to compute but significantly difficult for a brute-force cryptanalysis
  - ➠ Brute-force cryptanalysis: try all keys until “looks like” plaintext
  - ➠ a longer key means more work for brute-force cryptanalysis
  - ➠ encryption: O(N+1), brute-force: $O(2^{N+1})$

Computational Difficulty

- Algorithm should be efficient to compute but significantly difficult for a brute-force cryptanalysis

  ➠ Brute-force cryptanalysis: try all keys until "looks like" plaintext

  ➠ a longer key means more work for brute-force cryptanalysis

  ➠ encryption: O(N+1), brute-force: $O(2^{N+1})$

- Advances in computing benefit cryptographer more, but make old uses of cryptography easier to break

  ➠ DES (56 bit key) was standardized in 1977. It took 56 hours to break it in 1998, less than 1 day in 2008

Breaking an Encryption Scheme

- Ciphertext only
  - ➠ Fred has access to enough ciphertext only
  - ➠ brute-force search until finding a “recognizable plaintext”

Breaking an Encryption Scheme

- Ciphertext only
  - ➠ Fred has access to enough ciphertext only
  - ➠ brute-force search until finding a “recognizable plaintext”
- Known plaintext
  - ➠ Fred can obtain some $< plaintext, ciphertext >$ pairs
  - ➠ he has no control over what the plaintext is

Breaking an Encryption Scheme

- Ciphertext only
  - ➠ Fred has access to enough ciphertext only
  - ➠ brute-force search until finding a “recognizable plaintext”
- Known plaintext
  - ➠ Fred can obtain some $< plaintext, ciphertext >$ pairs
  - ➠ he has no control over what the plaintext is
- Chosen plaintext
  - ➠ Fred can choose a plaintext and have its ciphertext computed

Notation

Symbol	Meaning
$\oplus$	XOR, exclusive or
$\|$	concatenation (e.g., $ab\|cd = abcd$)
$K\{message\}$	encrypted with secret key K
$\{message\}_{Bob}$	encrypted with Bob's public key
$[message]_{Bob}$	signed by Bob with its private key

Trivial Ciphers

- Caesar cipher: shift each letter by 3
  ➠ A→ D, B → E

Trivial Ciphers

- Caesar cipher: shift each letter by 3
  - ➠ A→ D, B → E
- Captain midnight secret decoder ring: shift each letter by n
  - ➠ IBM → HAL (n = 1)
  - ➠ 26 possibilities

Trivial Ciphers

- Caesar cipher: shift each letter by 3
  ➠ A→ D, B → E
- Captain midnight secret decoder ring: shift each letter by n
  ➠ IBM → HAL (n = 1)
  ➠ 26 possibilities
- Monoalphabetic cipher: arbitrary mapping of one letter to another
  ➠ 26! (about 4×10^{26}) possibilities
  ➠ letter frequencies is preserved, making it vulnerable to letter frequency analysis

Breaking Monoalphabetic Cipher

- Match high/low-frequency n-grams in the language to the ciphertext until a “recognizable plaintext” (n = 1, 2, 3)
  ➠ English: e, t, a, o, I, n, s, h, r...

Example:

```
Si spy net work, big fedjaw iog link kyxogy
```

Breaking Monoalphabetic Cipher

- Match high/low-frequency n-grams in the language to the ciphertext until a "recognizable plaintext" (n = 1, 2, 3)
  ➠ English: e, t, a, o, I, n, s, h, r...

Example:

```
Si spy net work, big fedjaw iog link kyxogy
```

```
enxtybwpjld.fai..gkso...r.
abcdefghijklmnopqrstuvwxyz
```

Breaking Monoalphabetic Cipher

- Match high/low-frequency n-grams in the language to the ciphertext until a "recognizable plaintext" (n = 1, 2, 3)
  ➠ English: e, t, a, o, I, n, s, h, r...

Example:

```
Si spy net work, big fedjaw iog link kyxogy

          enxtybwpjld.fai..gkso...r.
          abcdefghijklmnopqrstuvwxyz

To the bad guys, for making our jobs secure
```

Fun Fact about Words

- Scrambled words can still be parsed by the human brain

Aoccdrnig to rscheearch at Cmabrigde uinervtisy, it deosn't mttaer waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteres are at the rghit pclae. The rset can be a tatol mses and you cansitll raed it wouthit a porbelm. Tihs is bcuseae we do not raed ervey lteter by itslef but the wrod as a wlohe.

Types of Cryptographic Functions

- Hash function

  ➠ zero keys

Types of Cryptographic Functions

- Hash function
  ➠ zero keys
- Secret key cryptography (symmetric cryptography)
  ➠ one shared key

Types of Cryptographic Functions

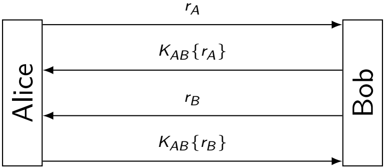
- Hash function
  ➠ zero keys
- Secret key cryptography (symmetric cryptography)
  ➠ one shared key
- Public key cryptography (asymmetric cryptography)
  ➠ two keys (public and private)

Secret Key Cryptography

- One key shared by both participators
  - ➠ one key, two operations (encryption and decryption)
  - ➠ how to securely agree on the key?
  - ➠ examples: DES, IDEA, AES...

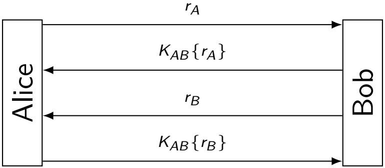
Applications of SKC

- Transmitting data over insecure channel
  ➠ eavesdropping, modification, deletion



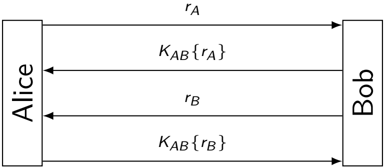
Applications of SKC

- Transmitting data over insecure channel
  ➠ eavesdropping, modification, deletion
- Secure storage on insecure media
  ➠ Google deletes your (replicated) data by discarding the key



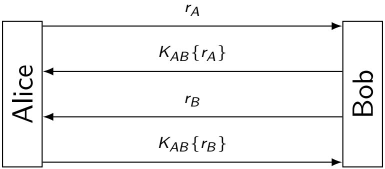
Applications of SKC

- Transmitting data over insecure channel
  ➠ eavesdropping, modification, deletion
- Secure storage on insecure media
  ➠ Google deletes your (replicated) data by discarding the key
- Data integrity check to prevent forgery of data checksum



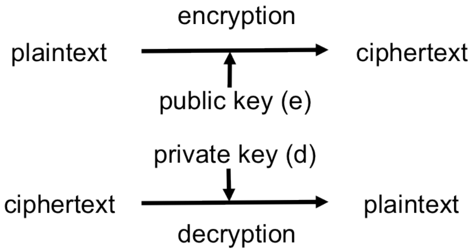
Applications of SKC

- Transmitting data over insecure channel
  ➠ eavesdropping, modification, deletion
- Secure storage on insecure media
  ➠ Google deletes your (replicated) data by discarding the key
- Data integrity check to prevent forgery of data checksum
- Authentication: prove knowledge of the key w/o revealing it
  ➠ Trudy can impersonate Alice to Bob by reflection attack



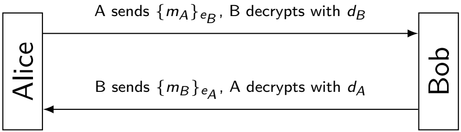
Public Key Cryptography

- Public key for encryption, and private key for decryption
  ➠ one operation (encryption), two keys (inverse of each other)
  ➠ anybody can encrypt a message, but only the owner of the private key can decrypt it



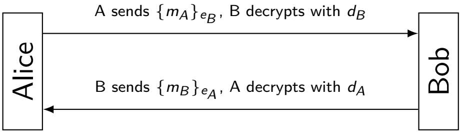
Applications of PKC

- Transmitting data over insecure channel


Alice
Bob
A sends $\{m_A\}_{e_B}$, B decrypts with d_B
B sends $\{m_B\}_{e_A}$, A decrypts with d_A

Applications of PKC

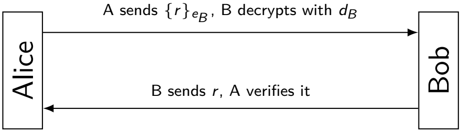
- Transmitting data over insecure channel


A sends $\{m_A\}_{e_B}$, B decrypts with d_B
Alice
Bob
B sends $\{m_B\}_{e_A}$, A decrypts with d_A

- Secure storage on insecure media
  ➠ generate a secret key for data encryption, then encrypt the key with PKC because PKC is slow

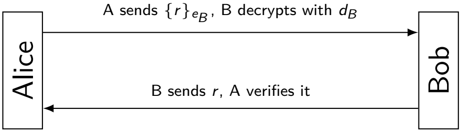
Applications of PKC

- Authentication

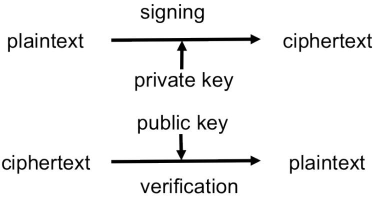

A sends $\{r\}_{e_B}$, B decrypts with d_B
B sends r, A verifies it
Alice
Bob

Applications of PKC

- Authentication


A sends $\{r\}_{e_B}$, B decrypts with d_B
Alice
Bob
B sends r, A verifies it

- Digital signature provides integrity and non-repudiation



Hash Function

- Hash function computes a fixed-length (short) number from a message of arbitrary length
  - ➠ given m, easy to compute $h(m)$
  - ➠ given $h(m)$, no easy way to find a m_1 that hashes to $h(m)$
  - ➠ computationally infeasible to find $h(m_1) = h(m_2)$

Applications of Hash Function

- Password hash
  - ➠ store a password hash instead of the password itself
  - ➠ use salt to mitigate the rainbow table attack $h(p|s)$

Applications of Hash Function

- Password hash
  - ➠ store a password hash instead of the password itself
  - ➠ use salt to mitigate the rainbow table attack $h(p|s)$
- Message integrity
  - ➠ use keyed hash to protect message integrity ($h(m|key)$)

Applications of Hash Function

- Password hash
  - ➠ store a password hash instead of the password itself
  - ➠ use salt to mitigate the rainbow table attack $h(p|s)$
- Message integrity
  - ➠ use keyed hash to protect message integrity $(h(m|key))$
- Message fingerprint and downline load security
  - ➠ publish with the data its md5 hash (download a copy of firefox)

Applications of Hash Function

- Password hash

  ➠ store a password hash instead of the password itself

  ➠ use salt to mitigate the rainbow table attack $h(p|s)$
- Message integrity

  ➠ use keyed hash to protect message integrity $(h(m|key))$
- Message fingerprint and downline load security

  ➠ publish with the data its md5 hash (download a copy of firefox)
- Digital signature efficiency

  ➠ sign a hash of the message instead of the message itself (Public key encryption is inefficient)

Summary

- What is cryptograph
- Three ways to break cryptography
- Trivial ciphers
- Cryptographic functions and their applications

- Next lecture: secret key cryptography