CSCI 454/554 Computer and Network Security

Topic 2. Introduction to Cryptography

Outline

- Basic Crypto Concepts and Definitions
- Some Early (Breakable) Cryptosystems
- “Key” Issues

Basic Concepts and Definitions
Cryptography

- **Cryptography**: the art of secret writing
- Converts data into unintelligible (random-looking) form
  - Must be **reversible** (can recover original data without loss or modification)
- **Not** the same as compression
  - $n$ bits in, $n$ bits out
  - Can be combined with compression
    - What’s the right order?

Cryptography vs. Steganography

- **Cryptography** conceals the contents of communication between two parties
- **Anonymous communication** conceals who is communicating
- **Kerckhoffs’s principle**
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge
- **Steganography** (hiding in plain sight) conceals the very existence of communication
  - Examples?
    - Watermark
    - Info leakage
  - Security through obscurity
    - Defense in depth
    - Open source software?

Encryption/Decryption

- Plaintext: a message in its original form
- Ciphertext: a message in the transformed, unrecognized form
- Encryption: the process that transforms a plaintext into a ciphertext
- Decryption: the process that transforms a ciphertext to the corresponding plaintext
- Key: the value used to control encryption/decryption.
Cryptanalysis

- “code breaking”, “attacking the cipher”
- Difficulty depends on
  - sophistication of the cipher
  - amount of information available to the code breaker
- Any cipher can be broken by exhaustive trials, but rarely practical
  - When can you recognize if you have succeeded?

Ciphertext Only Attacks

- Ex.: attacker can intercept encrypted communications, nothing else
  - when is this realistic?
- Breaking the cipher: analyze patterns in the ciphertext
  - provides clues about the encryption method/key

Known Plaintext Attacks

- Ex.: attacker intercepts encrypted text, but also has access to some of the corresponding plaintext (definite advantage)
  - When is this realistic?
- Requires plaintext-ciphertext pairs to recover the key, but the attacker cannot choose which particular pairs to access.
  - Makes some codes (e.g., mono-alphabetic ciphers) very easy to break
Chosen Plaintext Attacks

- Ex.: attacker can choose any plaintext desired, and intercept the corresponding ciphertext
  - When is this realistic?
- Choose exactly the messages that will reveal the most about the cipher

Chosen Ciphertext Attacks

- Ex.: attacker can present any ciphertext desired to the cipher, and get the corresponding plaintext
  - When is this realistic?
- Isn’t this the goal of cryptanalysis???

The “Weakest Link” in Security

- Cryptography is rarely the weakest link
- Weaker links
  - Implementation of cipher
  - Distribution or protection of keys
Perfectly Secure Ciphers

1. Ciphertext does not reveal any information about which plaintexts are more likely to have produced it
   - i.e., the cipher is robust against chosen ciphertext attacks
   and
2. Plaintext does not reveal any information about which ciphertexts are more likely to be produced
   - i.e., the cipher is robust against chosen plaintext attacks

Computationally Secure Ciphers

1. The cost of breaking the cipher quickly exceeds the value of the encrypted information
   and/or
2. The time required to break the cipher exceeds the useful lifetime of the information
   - Under the assumption there is not a faster / cheaper way to break the cipher, waiting to be discovered

Secret Keys vs. Secret Algorithms

- Security by obscurity
  - We can achieve better security if we keep the algorithms secret
  - Hard to keep secret if used widely
  - Reverse engineering, social engineering

- Publish the algorithms
  - Security of the algorithms depends on the secrecy of the keys
  - Less unknown vulnerability if all the smart (good) people in the world are examine the algorithms
Secret Keys vs. Secret Algorithms

- Commercial world
  - Published
  - Wide review, trust
- Military
  - Keep algorithms secret
  - Avoid giving enemy good ideas
  - Military has access to the public domain knowledge anyway.

Some Early Ciphers

Caesar Cipher

- Replace each letter with the one 3 letters later in the alphabet
  - ex.: plaintext CAT → ciphertext FDW

Trivial to break
Replace each letter by one that is $\delta$ positions later, where $\delta$ is selectable (i.e., $\delta$ is the key)

- example: IBM $\rightarrow$ HAL (for $\delta=25$)

- Also trivial to break with modern computers (only 26 possibilities)

---

**Mono-Alphabetic Ciphers**

- Generalized substitution cipher: an arbitrary (but fixed) mapping of one letter to another
  - 26! ($\approx 4.0 \times 10^{26} \approx 2^{88}$) possibilities
  - The key must specify which permutation; how many bits does that take?

---

**Attacking Mono-Alphabetic Ciphers**

- Broken by statistical analysis of letter, word, and phrase frequencies of the language
- Frequency of single letters in English language, taken from a large corpus of text:


<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8.2%</td>
</tr>
<tr>
<td>B</td>
<td>1.5%</td>
</tr>
<tr>
<td>C</td>
<td>2.8%</td>
</tr>
<tr>
<td>D</td>
<td>4.3%</td>
</tr>
<tr>
<td>E</td>
<td>12.7%</td>
</tr>
<tr>
<td>F</td>
<td>2.2%</td>
</tr>
<tr>
<td>G</td>
<td>2.0%</td>
</tr>
<tr>
<td>H</td>
<td>6.1%</td>
</tr>
<tr>
<td>I</td>
<td>7.0%</td>
</tr>
<tr>
<td>J</td>
<td>0.2%</td>
</tr>
<tr>
<td>K</td>
<td>0.8%</td>
</tr>
<tr>
<td>L</td>
<td>4.0%</td>
</tr>
<tr>
<td>M</td>
<td>2.4%</td>
</tr>
<tr>
<td>N</td>
<td>6.7%</td>
</tr>
<tr>
<td>O</td>
<td>7.5%</td>
</tr>
<tr>
<td>P</td>
<td>1.9%</td>
</tr>
<tr>
<td>Q</td>
<td>0.1%</td>
</tr>
<tr>
<td>R</td>
<td>6.0%</td>
</tr>
<tr>
<td>S</td>
<td>6.3%</td>
</tr>
<tr>
<td>T</td>
<td>9.1%</td>
</tr>
<tr>
<td>U</td>
<td>2.8%</td>
</tr>
<tr>
<td>V</td>
<td>1.0%</td>
</tr>
<tr>
<td>W</td>
<td>2.4%</td>
</tr>
<tr>
<td>X</td>
<td>0.2%</td>
</tr>
<tr>
<td>Y</td>
<td>2.0%</td>
</tr>
<tr>
<td>Z</td>
<td>0.1%</td>
</tr>
</tbody>
</table>
Attacking... (Cont’d)

- If all words equally likely, probability of any one word would be quite low
  - how many words are there in the English language?
- Actual frequencies of some words in English language:

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>6.4%</td>
</tr>
<tr>
<td>of</td>
<td>4.0%</td>
</tr>
<tr>
<td>and</td>
<td>3.2%</td>
</tr>
<tr>
<td>to</td>
<td>2.4%</td>
</tr>
<tr>
<td>a</td>
<td>2.1%</td>
</tr>
<tr>
<td>in</td>
<td>1.8%</td>
</tr>
<tr>
<td>that</td>
<td>1.2%</td>
</tr>
<tr>
<td>is</td>
<td>1.0%</td>
</tr>
<tr>
<td>it</td>
<td>0.9%</td>
</tr>
<tr>
<td>for</td>
<td>0.8%</td>
</tr>
<tr>
<td>as</td>
<td>0.8%</td>
</tr>
</tbody>
</table>

(Tip: Counting Letter Frequencies)

- Program letter, written by TJ O'Connor
- Output for Declaration of Independence:

Vigenere Cipher

- A set of mono-alphabetic substitution rules (shift amounts) is used
  - the key determines what the sequence of rules is
  - also called a poly-alphabetic cipher
- Ex.: key = (3 1 5)
  - i.e., substitute first letter in plaintext by letter+3, second letter by letter+1, third letter by letter+5
  - then repeat this cycle for each 3 letters
Vigenere... (Cont’d)

Ex.: plaintext = “BANDBAD”

plaintext message
B A N D B A D

shift amount
3 1 5 3 1 5 3

ciphertext message
E B S G C F G

Breaking the cipher: look for repeated patterns in the ciphertext

Hill Ciphers

Encrypts \( m \) letters of plaintext at each step
- i.e., plaintext is processed in blocks of size \( m \)
- Encryption of plaintext \( p \) to produce ciphertext \( c \) is accomplished by: \( c = Kp \)
  - the \( m \times m \) matrix \( K \) is the key
  - \( K \)'s determinant must be relatively prime to size of alphabet (26 for our example)
  - decryption is multiplication by inverse: \( p = K^{-1}c \)
  - remember: all arithmetic mod 26

Hill Cipher Example

For \( m = 2 \), let \( K = \begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix} \), \( K^{-1} = \begin{pmatrix} 21 & 2 \\ 3 & 25 \end{pmatrix} \)

Plaintext \( p = \)
\[
\begin{pmatrix} A & B & X & Y & D & G \\ 0 & 1 & 23 & 24 & 3 & 6 \end{pmatrix}
\]

\[
(1\cdot0+2\cdot23) \mod 26 \\
(3\cdot1+5\cdot24) \mod 26
\]

Ciphertext \( c = \)
\[
\begin{pmatrix} 2 & 5 & 19 & 7 & 15 & 13 \\ C & F & T & H & P & N \end{pmatrix}
\]
Hill... (Cont’d)

- Fairly strong for large $m$
- But, vulnerable to chosen plaintext attack
  - choose $m$ plaintexts, generate corresponding ciphertexts
  - form a $m \times m$ matrix $X$ from the plaintexts, and
    - $m \times m$ matrix $Y$ from the ciphertexts
  - can solve directly for $K$ (i.e., $K = YX^{-1}$)

Permutation Ciphers

- The previous codes are all based on substituting one symbol in the alphabet for another symbol in the alphabet
- Permutation cipher: permute (rearrange, transpose) the letters in the message
  - the permutation can be fixed, or can change over the length of the message

Permutation... (Cont’d)

- Permutation cipher ex. #1:
  - Permute each successive block of 5 letters in the message according to position offset $<+1,+3,-2,0,-2>$

 plaintext message

\[
\begin{array}{cccccc}
W & H & Y & O & W & Y \\
H & Y & C & A & N & T \\
I & F & L & Y \\
\end{array}
\]

 ciphertext message

\[
\begin{array}{cccccc}
Y & W & W & O & H & C \\
H & N & A & Y & F & T \\
Y & L & I \\
\end{array}
\]
Permutation... (Cont’d)

- Permutation cipher ex. #2:
  - arrange plaintext in blocks of n columns and m rows
  - then permute columns in a block according to a key K

\[
\begin{array}{cccc}
4 & 3 & 1 & 2 \\
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
\end{array}
\]

Key (perm. of columns) \( m = 3 \)

Plaintext symbol positions

\[
\begin{array}{cccc}
3 & 7 & 11 & 4 \\
8 & 12 & 2 & 6 \\
10 & 1 & 5 & 9 \\
\end{array}
\]

ciphertext sequence (by plaintext position) for one block

A longer example: plaintext = “ATTACK POSTPONED UNTIL TWO AM”

\[
\begin{array}{cccccccc}
4 & 3 & 1 & 2 & 5 & 7 & 6 \\
A & T & T & A & C & K & P \\
O & S & T & P & O & N & E \\
D & U & N & T & I & L & T \\
W & O & A & M & X & Y & Z \\
\end{array}
\]

ciphertext

TTNA APTM TSUO AODW COIX PETZ KNLY

A Perfectly Secure Cipher: One-Time Pads

- According to a theorem by Shannon, a perfectly secure cipher requires:
  - a key length at least as long as the message to be encrypted
  - the key can only be used once (i.e., for each message we need a new key)
  - Very limited use due to need to negotiate and distribute long, random keys for every message
OTP... (Cont’d)

- Idea
  - generate a random bit string (the key) as long as the plaintext, and share with the other communicating party
  - encryption: XOR this key with plaintext to get ciphertext
  - decrypt: XOR same key with ciphertext to get plaintext

plaintext XOR ciphertext XOR plaintext

Key

Why can’t the key be reused?

Some “Key” Issues
Types of Cryptography

- Number of keys
  - Hash functions: no key
  - Secret key cryptography: one key
  - Public key cryptography: two keys - public, private
- The way in which the plaintext is processed
  - Stream cipher: encrypt input message one symbol at a time
  - Block cipher: divide input message into blocks of symbols, and processes the blocks in sequence
    - May require padding

Secret Key Cryptography

- Same key is used for encryption and decryption
- Also known as
  - Symmetric cryptography
  - Conventional cryptography

Secret Key Cryptography (Cont’d)

- Basic technique
  - Product cipher:
    - Multiple applications of interleaved substitutions and permutations
Secret Key Cryptography (Cont’d)

- Ciphertext approximately the same length as plaintext
- Examples
  - Stream Cipher: RC4
  - Block Cipher: DES, IDEA, AES

Applications of Secret Key Cryptography

- Transmitting over an insecure channel
  - Challenge: How to share the key?
- Secure Storage on insecure media
- Authentication
  - Challenge-response
  - To prove the other party knows the secret key
  - Must be secure against chosen plaintext attack
- Integrity check
  - Message Integrity Code (MIC)
    - a.k.a. Message Authentication Code (MAC)

Public Key Cryptography (PKC)

- Invented/published in 1975
- A public/private key pair is used
  - Public key can be publicly known
  - Private key is kept secret by the owner of the key
- Much slower than secret key cryptography
- Also known as
  - Asymmetric cryptography
Another mode: digital signature
- Only the party with the private key can create a digital signature.
- The digital signature is verifiable by anyone who knows the public key.
- The signer cannot deny that he/she has done so.
- The signature is created on a hash value of the message.

Applications of Public Key Cryptography
- Data transmission:
  - Alice encrypts $m_a$ using Bob's public key $e_B$, Bob decrypts $m_a$ using his private key $d_B$.
- Storage:
  - Can create a safety copy: using public key of trusted person.
- Authentication:
  - No need to store secrets, only need public keys.
  - Secret key cryptography: need to share secret key for every person to communicate with.

Applications of PKC (Cont’d)
- Digital signatures
  - Sign hash $H(m)$ with the private key
  - Authorship
  - Integrity
  - Non-repudiation: can't do with secret key cryptography
- Key exchange
  - Establish a common session key between two parties
  - Particularly for encrypting long messages
Hash Algorithms

- Also known as
  - Message digests
  - One-way transformations
  - One-way functions
  - Hash functions
- Length of $H(m)$ much shorter than length of $m$
- Usually fixed lengths: 128 or 160 bits

Hash Algorithms (Cont’d)

- Desirable properties of hash functions
  - Performance: Easy to compute $H(m)$
  - One-way property (Preimage resistance): Given $H(m)$ but not $m$, it’s difficult to find $m$.
  - Weak collision free (Second preimage resistance): Given $m_1$, it’s difficult to find $m_2$ such that $H(m_1) = H(m_2)$.
  - Strong collision free (Collision Resistance): Computationally infeasible to find $m_1, m_2$ such that $H(m_1) = H(m_2)$

Applications of Hash Functions

- Primary application
  - Generate/verify digital signatures

  - Message $m$ → $H(m)$ → Sign → Signature $\text{Sig}(H(m))$
  - Private key

  - Message $m$ → $H(m)$ → Verify → Yes/No
  - Public key

  - Message $m$ → $H(m)$ → Signature $\text{Sig}(H(m))$ → Public key
Applications of Hash Functions (Cont’d)

- Password hashing
  - Doesn’t need to know password to verify it
  - Store $H(password + salt)$ and salt, and compare it with the user-entered password
  - Salt makes dictionary attack more difficult

- Message integrity
  - Agree on a secret key $k$
  - Compute $H(m | k)$ and send with $m$
  - Doesn’t require encryption algorithm, so the technology is exportable

Applications of Hash Functions (Cont’d)

- Message fingerprinting
  - Verify whether some large data structures (e.g., a program) has been modified
  - Keep a copy of the hash
  - At verification time, recompute the hash and compare

  - Hashing program and the hash values must be protected separately from the large data structures

Summary

- Cryptography is a fundamental, and most carefully studied, component of security
  - not usually the “weak link”
- “Perfectly secure” ciphers are possible, but too expensive in practice
- Early ciphers aren’t nearly strong enough
- Key distribution and management is a challenge for any cipher