Overview

- Protocols
- Information Hiding

Protocols

- The point of cryptography is to solve problems
- Problems involve secrecy, authentication, integrity and dishonest people
- A Protocol is a series of steps, involving two or more parties, designed to accomplish a task
- Every step must be executed in turn
- Everyone involved in the protocol must know the protocol and all of the steps to follow in advance
- Everyone involved must agree to follow it
- The protocol must be unambiguous; each step must be well defined
- The protocol must be complete; there must be a specific action for every possible situation
- A cryptographic protocol is a protocol that uses cryptography
- It should not be possible to do more or learn more than what is specified in the protocol

Arbitrated Protocols

- An arbitrator is a disinterested third party trusted to complete a protocol.
  - Disinterested: no vested interest in the protocol and no allegiance to any of the parties
  - Trusted: the parties in the protocol will accept what the arbitrator says is true
- In the real world, lawyer, banks or notary public are often used as arbitrators
- Trent will play the role of the arbitrator

People

- Alice: First participant in all the protocols
- Bob: Second participant in all the protocols
- Carol: Participant in the three- and four-party protocols
- Dave: Participant in the four-party protocols
- Eve: Eavesdropper
- Mallory: Malicious active attacker
- Trent: Trusted arbitrator
- Walter: Warden, he’ll be guarding Alice and Bob in some protocols
- Peggy: Prover
- Victor: Verifier

Problems with arbitrated protocols

- It is easier to find and trust a third party in the real world than on a network
- There is always an extra delay
- The arbitrator must deal with every transaction; he becomes a bottleneck
- Since everyone trust the arbitrator, he becomes a very vulnerable point for attacks
Adjudicated Protocols

• Arbitrated protocols can be subdivided into two lower-layer protocols
  – One non-arbitrated protocol executed every time parties want to complete the protocol
  – The other is an arbitrated protocol executed only when there is a dispute
• This special type of arbitrator is called an adjudicator
• An adjudicator is also disinterested and trusted third party
  – Only called in to determine whether the protocol was executed fairly
• Judges are professional adjudicators

Attacks against Protocols

• Cryptographic attacks can be directed against the
  – Cryptographic algorithms used in protocols
  – Cryptographic techniques and implementations
  – Protocols them self
• When studying protocols we will assume the algorithms and implementations are secure

Adjudicated protocols

• Rely on the parties to be honest
• If someone suspect cheating, evidence exist so that a third party could determine if somebody cheated
• We would also like to know who the cheater are
• Detects cheating instead of preventing it

Attacks against Protocols

• Passive attacks
  – Someone not involved in the protocol tries to eavesdrop on some or all of the protocol
  – Hard to detect, so we try to prevent eavesdropping instead

Self-enforcing Protocols

• The protocol itself guarantees fairness
  – No arbitrator is necessary to complete the protocol
  – No adjudicator is required to resolve disputes
• The protocol is designed so there cannot be disputes
• Cheating can be detected, and the protocol aborted
• Self enforcing protocols does not exist for every situation

Attacks against Protocols

• Active attacks
  – An attacker could try to alter the protocol to his own advantage
  – Pretend to be someone else
  – Alter messages, introduce new messages, delete existing messages, replay old messages
### Attacks against Protocols

- Cheaters
  - Parties involved in the protocol
  - Passive cheaters
    - Follows the protocol but try to obtain more information
  - Active cheaters
    - Disrupt the protocol in progress in order to cheat

### Key Exchange

- A common technique is to encrypt each individual conversation with a separate key
  - The key is then called a session key
- Useful because they only exist and are valid during a limited time

### First Protocol

1. Alice and Bob agree on a cryptosystem
2. Alice and Bob agree on a key
3. Alice takes her plaintext message and encrypts it using the encryption algorithm and the key; creating a ciphertext
4. Alice sends the ciphertext message to Bob
5. Bob decrypts the ciphertext message with the same algorithm and key and reads it

### Key Exchange with Symmetric Cryptography

- Assumes Alice and Bob share a secret key with the Key Distribution Center, the KDC (Trent in our protocol)
1. Alice calls Trent and request a session key to communicate with Bob
2. Trent generates a random session key. He encrypts two copies of it; one with Alice's key and one with Bob's key. He send both copies to Alice
3. Alice decrypts her copy of the session key
4. Alice sends Bob his copy of the session key
5. Bob decrypts his copy of the session key
6. Both Alice and Bob use the session key to communicate with each other
- Relies on the absolute security of Trent
- If Mallory corrupts Trent, the whole network is compromised
- Trent is also a bottleneck

### Key Exchange with Public-Key Cryptography

**Protocol:**
1. Alice gets Bob's public key from the KDC
2. Alice generates a random session key, encrypts it with Bob's public key and sends it to Bob
3. Bob decrypts the key with his private key
4. Both Alice and Bob use the session key to communicate with each other
**Man-in-the-Middle attack**
- Mallory can imitate Bob when talking to Alice and imitate Alice when talking to Bob
  1. Alice sends Bob her public key. Mallory intercepts this key and sends Bob his own public key.
  2. Bob sends Alice his public key. Mallory intercepts this key and sends Alice his own public key.
  3. When Alice sends a message to Bob, encrypted in “Bob’s” public key, Mallory intercepts it. Since this message is really encrypted with his own public key, he decrypts it with his private key, re-encrypts it with Bob’s public key, and sends it on to Bob.
  4. When Bob sends a message to Alice, encrypted in “Alice’s” public key, Mallory intercepts it. Since this message is really encrypted with his own public key, he decrypts it with his private key, re-encrypts it with Alice’s public key, and sends it on to Alice.
- Even if the keys are stored in a database this attack will work, if Mallory can intercept those messages too.

**Authentication**
- When Alice logs into a host computer, how does the host know who she is?
- Passwords are the traditional solution
  - Alice enters her password and the host confirms that it is correct.
- Both Alice and the host knows a secret piece of information and the host requests it from Alice every time she tries to log in.

**Man-in-the-Middle attack**

**Key Exchange with Digital Signatures**
- Digital signatures during a session-key exchange protocol circumvents the man-in-the-middle attack.
- Trent signs both Alice’s and Bob’s public keys.
  - The signed keys include a signed certification of ownership.
- When Alice and Bob receive the keys, they each verify Trent’s signature.

**Authentication using One-Way Functions**
- The host does not need to know the passwords.
- It just have to be able to differentiate valid passwords from invalid passwords.
- This can be done with one-way functions.
- Instead of storing passwords, the host stores one-way function values of the passwords.
  1. Alice sends the host her password.
  2. The host performs a one-way function on the password.
  3. The host compares the result of the one-way function to the value previously stored.
- The list of passwords is useless (?) because the one-way function cannot be reversed to recover the passwords.

**Dictionary Attack**
- A file of passwords encrypted with a one-way function is still vulnerable.
- Mallory compiles a list of the 1,000,000 most common passwords.
- He runs all passwords through the one-way function.
- He compares his computed values to the ones in the password file.
- Salts can make it more difficult:
  - Add a salt (random string) to each password before using the one-way function.
  - Mallory has to generate a new one-way hash for each possible salt value.
  - The salt must be big (Unix usually uses 12 bits, far too little).
**SKEY**

- An authentication program that relies on a one-way function for its security
- Alice enters a random number, \( R \)
- We compute \( f(R), f(f(R)), f(f(f(R))), \) and so on
- \( X[1], X[2], \ldots, X[100] \)
- Alice keeps these numbers securely
- \( X[101] \) is stored in a login database
- The first time Alice wants to log in, she types her name and \( x_{100} \). The computer calculates \( f(x_{100}) \) and compares it to \( x_{101} \). If the match, Alice is authenticated
- The computer replaces \( x_{101} \) with \( x_{100} \) and Alice removes \( x_{100} \) from her list
- The login database is useless to Mallory
- An eavesdropper can't reuse an \( x \) value

**Arbitrated Protocol**

- This protocol uses Trent, who has a trusted timestamping service, and Alice, who wishes to timestamp a document
  1. Alice transmits a copy of the document to Trent
  2. Trent records the date and time he received the document and retains a copy of the document for safekeeping
- Problems:
  - Privacy
  - Trent's database need to be huge
  - An error in transmission or an error in Trent's storage could completely invalidate Alice's claim of a timestamp
  - We have to trust the arbitrator Trent

**Authentication using Public-Key Cryptography**

- Even with salt, the password can be read with anyone with access to the data path
- Attempt with public keys:
  1. The host sends Alice a random string
  2. Alice encrypts the string with her private key and sends it back to the host, along with her name
  3. The host looks up Alice's public key in its database and decrypts the message using that public key
  4. If the decrypted string matches what the host sent Alice in the first place, the host allows Alice access to the system
- Foolish to encrypt arbitrary strings.

**Improved Arbitrated Protocol**

- One-way hash functions and digital signatures help:
  1. Alice produces a one-way hash of the document
  2. Alice transmits the hash to Trent
  3. Trent appends the date and time he received the hash onto the hash and then digitally signs the result
  4. Trent then sends the signed hash with timestamp back to Alice

**Timestamping Services**

- Sometimes you want to prove that a document existed on a certain date
- Haber and Stornetta at Bellcore 1991:
  - The data itself must be timestamped, without any regard to the physical medium on which it resides
  - It must be impossible to change a single bit of the document without that change being apparent
  - It must be impossible to timestamp a document with a date and time different from the present one

**Improved Arbitrated Protocol**

- Problems?
  - Privacy - fixed
  - Trent's database need to be huge - fixed (no storage needed - the signature certifies the date)
  - An error in transmission or an error in Trent's storage could completely invalidate Alice's claim of a timestamp - fixed (Alice checks what Trent returned)
  - We have to trust the arbitrator Trent - Still a problem - Alice and Trent can still make any timestamps they want
Improved Arbitrated Protocol

- We can fix the last problem by linking all timestamps together - the timestamp documents have links to the previous timestamped document
  - If someone challenges we find the owners of timestamps T[i-1] and T[i+1], then T[i-2] and T[i+2], ...
  - Trent and Alice must create a fictitious chain of documents and timestamps both before and after Alice's document, long enough to exhaust the patience of anyone challenging the timestamp

Fair coin flipping

- Alice flips a coin, Bob guesses heads or tails, Alice announces the result
- We want:
  - Alice must flip the coin before Bob guesses
  - Alice must not be able to re-flip the coin after hearing Bob's guess
  - Bob must not be able to know the coin landed before making his guess

Fair coin flipping using One-way functions

- If Alice can find x and x' such that x is even and x' is odd, and y = f(x) = f(x'), then she can cheat every time
- The last significant bit of f(x) must be uncorrelated with x, otherwise Bob can gain an advantage
  - e.g. if f(x) produces odd values 75% of the time

Fair coin flipping using Public-Key cryptography

- This protocol works with either symmetric or public-key cryptography
- We require that the algorithm commute, that is
  - $D_K1(E_{K2}(E_{K1}(M))) = E_{K2}(M)$
- Generally, this is not true for symmetric algorithms but true for some public key algorithms (for example RSA)
- This protocol is self-enforcing

Fair coin flipping using One-way functions

- If Alice and Bob can agree on a one-way function, this protocol is simple:
  1. Alice chooses a random number, x. She computes y = f(x), where f(x) is the one-way function
  2. Alice sends y to Bob
  3. Bob guesses whether x is even or odd and sends his guess to Alice
  4. If Bob's guess is correct, their result of the coin flip is heads, otherwise tails.
  5. Bob confirms that y = f(x)
Secure Elections

- Requirements for an ideal protocol for computerized voting:
  1. Only authorized voters can vote
  2. No one can vote more than once
  3. No one can determine for whom anyone else voted
  4. No one can duplicate anyone else's vote
  5. No one can change anyone else's vote without being discovered
  6. Every voter can make sure that his vote has been taken into account in the final tabulation

Advanced Protocol

- Uses ANDOS - All or nothing disclosure of secrets
- Satisfies all six requirements

Simplistic Voting Protocol 1

1. Each voter encrypts his vote with the public key of a Central Tabulating Facility (CTF)
2. Each voter sends his vote in to the CTF
3. The CTF decrypts the votes, tabulates them, and make the results public
   - The CTF has no idea even if the votes are from eligible voters, if voters vote more than once

Simplistic Voting Protocol 2

1. Each voter signs his vote with his private key
2. Each voter encrypts his signed vote with the CTF's public key
3. Each voter sends his vote to the CTF
4. The CTF decrypts the votes, check the signatures, tabulates the votes, and make the results public
   - Satisfies properties 1 and 2: Only authorized voters can vote and no one can vote more than once
   - But the CTF knows who voted for what

Voting without a Central Facility

- Works only for few people

Advanced Protocol

1. The CTF publishes a list of all legitimate voters
2. Within a specified deadline each voter tells the CTF whether he intends to vote
3. The CTF publishes a list of voters participating in the election
4. Each voter receives an identification number, I, using an ANDOS protocol
5. Each voter generates a public-key/private-key key pair: k,d. If v is the vote, he generates the following message and sends it to the CTF:
   \[ E_k(I,v) \]
   - This message must be sent anonymously
6. The CTF acknowledges receipt of the vote by publishing:
   \[ E_k(I,v) \]
7. Each voter sends the CTF: I, d
8. The CTF decrypts the votes. At the end of the election, it publishes the results of the election and, for each different vote, the list of all \( E_k(I,v) \) values that contained that vote
9. If a voter observes that his vote is not properly counted, he protest by sending the CTF:
   \[ I, E_k(I,v), d \]
10. If a voter wants to change his vote from v to v', he sends the CTF:
    \[ I, E_k(I, v'), d \]
## Information Hiding
- Steganography
- Digital watermarking
  - Fingerprinting
- Methods
- Attacks

## Who is interested?
- Military and intelligence agencies
- Criminals
- Law enforcement and counter intelligence
- Secret communication without encryption
- Media companies

## Classification

<table>
<thead>
<tr>
<th>Covert channels</th>
<th>Steganography</th>
<th>Anonymity</th>
<th>Copyright marking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linguistic steganography</td>
<td>Robust steganography</td>
<td>Fingerprinting</td>
<td>Watermarking</td>
</tr>
<tr>
<td>Invisible watermarking</td>
<td>Visible watermarking</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Why is it important?
- Copy protection have never really worked
  - needs tamper-proof hardware
- Instead we can try to detect copies
  - Tracing original owner
  - Detecting who made the copy
- Sending secret messages without using cryptography or without being detected

## Terminology
- The information to be hidden (watermark, fingerprint, a secret message) is **embedded** in a **cover** object (cd, video, text) giving stego object (marked object).
- The embedding is done with a **key**, a secret variable that is in general known to the object's owner.
- The recovery of the embedded mark may or may not require a key

## Steganography
- General problem:
  - Alice and Bob are in prison and need to communicate with each other to discuss their break out plan
  - All messages have to through the warden Walter
  - If the warden notice any suspicious messages all communication between Alice and Bob will stop
- Lot of historical examples (invisible ink, microfilm, etc.)
Wisdom from cryptography

- Kerckhoffs principle:
  - Assume that the attacker knows the methods
- Secure steganography:
  - An opponent who doesn't know the key cannot obtain any evidence of the communication

Security of steganography systems

- Three parts: detecting, extracting and disabling embedded information
- Requirements for secure steganographic algorithms:
  - Messages are hidden using a public algorithm and a key: the secret key must identify the sender uniquely
  - Only the holder of the correct key can detect, extract, and prove the existence of the hidden message. Nobody else should be able to find any statistical evidence of a message's existence
  - Even if the enemy knows the contents of one hidden message, he should have no chance of detecting others
  - It is computationally infeasible to detect hidden messages

Steganography

- Pure steganography
  - No key is necessary
  - $E: C \times M \rightarrow C$
  - $D: C \rightarrow M$
  - Relies on the secrecy of the method
- Secret key steganography
  - Uses a shared and secret stego-key
  - $E: C \times M \times K \rightarrow C$  
  - $D: C \times K \rightarrow M$
- Public key steganography
  - Two keys: one public and one private

Digital watermarking

- Additional requirement: robustness
- There should be no way of removing the embedded information without rendering the cover object unusable
- Visible watermarks
- Imperceivable watermarks
- Fingerprinting: a unique watermark in each object

Active warden

- It is conjectured that neither public key steganography nor pure steganography is possible with an active (malicious) warden
- A public key needs to be certified - Alice and Bob have no means of certifying the keys when all communication goes through the warden Walter.

Applications

- Watermarking for copyright protection
  - Embed information about the owner of the object
  - Used resolve rightful ownership
  - Requires a very high level of robustness
- Fingerprinting for traitor tracking
  - Each object have unique watermark identifying for example the buyer of a object
  - Copies can be traced
- Fragile watermarks
  - Protects against modification
- Data augmentation
- Automatic monitoring of copyrighted material on the Web/Radio/TV etc.
**Robustness**

- Watermarks should resist any kind of distortion introduced by standard or malicious data processing
- **No** such method has be proposed so far
  - Not clear yet if it is possible

---

**Least Significant Bit tweaking**

- Represent the object as vector of integers
  - For sound: e.g. 16 bit samples
  - For images: e.g. red, blue or green components, luminance or chrominance components
- Change the least significant bit in either all or some integers to represent a 1 or 0 in the mark
- Depending on the amount of embedded information it is quite unperceivable
- Can be used with a key, which describes which bits to change

---

**Robustness**

- IFPI (International Federation for the Phonographic Industry) robustness requirements:
  - The marking mechanism should not affect the sonic quality of the sound recording
  - The marking information should be recoverable after a wide range of filtering and processing operations (D/A and A/D conversion, MPEG compression, adding a second mark with the same system, etc.)
  - There should be no other way to remove or alter the embedded mark without sufficient degradation of the sound quality as to render it unusable

---

**Echo hiding**

- Advanced audio marking systems
- Inserts echos (delay in the 0.5-2 ms range) that cannot be perceived
- Two different echos gives ones and zeros
- Survives compression
- Obvious attack: detect and remove echo

---

**Methods**

- LSB - Least Significant Bit tweaking
- Echo Hiding
- More advanced methods uses other domains: Discrete Cosine Transform (DCT), Fourier transform, Wavelets

---

**Attacks**

- Robustness attacks
- Presentation attacks
- Interpretation attacks
Attacks - Jitter Attack

- Attacks against audio marking systems that tweak low order bits whose locations is specified by a key
- Attack:
  1. Split the signal into chunks of 500 samples
  2. Duplicate or remove on sample from each chunk, giving chunks of 499 or 501 samples
  3. "Almost imperceptible"
  4. Impossible to find the mark - we cannot use the key to find the bits anymore
  - If the mark is in sample $s[a]$, $s[b]$ and $s[c]$ before, they now might be in sample $s[a-1]$, $s[b+3]$, $s[c+2]$

StirMark

- Most image marking schemes survive basic manipulations: rotation, compression, resizing
- Combinations are often not handled though
- StirMark: tool developed for simple robustness testing of image marking systems
- Applies minor distortions
- Defeats most commercial image marking systems

Mosaic Attack

- 'Presentation Attack'
- Designed to defeat systems combining watermarks and web crawlers
- The image is chopped into smaller pieces and put together in the web page in e.g. a table
- Each sub-picture must be small enough such that the (partial) watermark can no longer be detected (for that sub-picture)

StirMark

- Images are slightly stretched, sheared, bent, rotated etc.
Example

- SDMI – Secure Digital Music Initiative
- SDMI compliant devices
  - Portable players, sound cards, etc.
- To play a CD, it must pass some tests
- All legacy CDs and new legally bought CDs must pass the tests

Attacks

- Break the identification technology
- Break the watermark technology
- Disable the test

Goal

- To prevent the following
  - Bob buys an CD, rips the tracks to his computer, compresses them, sends them to Alice. Alice burns them on an CD and tries to play them on a SDMI compliant device
- If a track has been altered (removed from the CD, compressed, etc.) it should not pass the test and refused

Legal protection

- If software methods and/or hardware does not work?
- DMCA - Digital Millenium Copyright Act
- EU copyright directive
  - ..."Member States shall provide adequate legal protection against the circumvention of any effective technological measures, which the person concerned carries out in the knowledge, or with reasonable grounds to know, that he or she is pursuing that objective." ...  
  - "..."technological measures" means any technology, device or component that, in the normal course of its operation, is designed to prevent or restrict acts, in respect of works or other subject-matter..."

Two algorithms

- A watermarking technology
- An identification technology
- If a watermark is present, the tracks are new content

<table>
<thead>
<tr>
<th></th>
<th>Marked</th>
<th>Non marked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed and invalid</td>
<td>Reject</td>
<td>Reject?</td>
</tr>
<tr>
<td>Unsigned</td>
<td>Reject</td>
<td>Accept</td>
</tr>
<tr>
<td>Signed and valid</td>
<td>Accept</td>
<td>Accept?</td>
</tr>
</tbody>
</table>

Summary

- New field computer security - general theories do not exist yet
- It is still not clear if perfect (with regard to robustness) watermarking systems is possible
- Most commercial system is easily defeated
Next time

- Program Security
  - Flaws in programs.
  - Both intentional (malicious code) and accidental (erroneous code) flaws.
  - How to detect flaws, avoid flaws and protect against flaws.