Cryptography Basics

1 Secret Key Encryption

- Convention and Terms
  - Plaintext:
  - Ciphertext:
  - Encryption:
  - Decryption:
  - Cryptanalysis:
- Secret Key Encryption is also called Symmetric Key Encryption: the key used for encryption is the same as the key for decryption.

1.1 Classical Cryptosystems

- Monoalphabetic Substitution Cipher
  - Units of plaintext are *substituted* with ciphertext according to a regular system. The "units" may be single letters, pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.
  - Example:

    ```
    Mapping: A->D, B->E, so on,
    Encryption: HELLO -> KHOOR
    ```

  - Drawbacks: this cipher can be easily broken using frequency analysis. In any language, the frequencies of the characters are different. For example, in English, 'z' appears much less frequent than 't'. Although each character is mapped to another character in the substitution cipher, their frequencies can reveal themselves.
- Polyalphabetic Substitution Cipher: using multiple substitution alphabets.
  - The Enigma machine is more complex but still fundamentally a polyalphabetic substitution cipher.
- The Enigma Machines.
- Transposition Cipher
  - Units of the plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged. By contrast, in a substitution cipher, the units of the plaintext are retained in the same sequence in the ciphertext, but the units themselves are altered.
  - Example

```
HELLO WORLD  -> HLOOL  --> HLOOLELWRD
                ELWRD
```

- One-time pad (see wikipedia)
  - Invented in 1917
  - Used in low bandwidth
  - Used in military
  - The Hotline between US and former Soviet Union use one-time pad.
  - A perfect encryption schemes, in terms of security.

1.2 DES and AES

- DES (Data Encryption Standard) history:
  - Horst Feistel (IBM) created the "Lucifer" block cipher as a result of research hobby.
  - Later, "Lucifer" became a major IBM initiative, and IBM revised it, named it DSD-1.
  - 1974: IBM decided to respond to the call for encryption standard issued by NBS (National Bureau of Standards). This means that IBM would be required to relinquish its patent rights, essentially giving, not selling, the algorithm to the world.
  - Early 1974, NSA offered to work with IBM on DSD-1. NSA's all-star cryptanalysts would analyze DSD-1 and qualify the algorithm. IBM should allow NSA to control the implementation of the crypto system. IBM took the offer.
  - Horst Feistel's Lucifer specified a 128-bit key, but NSA did not like that, and cut it into 64 bits, which is 8 bytes. IBM used one bit of each byte for "parity checks". This reduced the size of the key to 56 bits.
  - 1977, the final algorithm was accepted as a standard, called DES.
- DES Technical Details
  - 56-bit key, 64-bit block, 16 rounds.
  - Block Cipher: 64-bit block
  - Brute-force attack 2^{56}
  - 1998, Electronic Frontier Foundation (EFF) built a "DES cracker" machine with $250,000. It broke a DES key in 56 hours.
  - 1999, `distributed.net` and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes.
  - 3DES: C = E1 [D2 [E1 [P]]]
    - 3DES use 2 keys (some use 3 keys), 112 bits are sufficient.
    - How come 3DES did not become another standard?
    - It is slow: basically, we have to run DES algorithm sequentially three times.
- AES (Advanced Encryption Standard)

  - In January 1997, NIST announced that they wished to choose a successor to DES to be known as AES.
  - In September 1997, NIST ofifcially calls for new algorithms for AES. The algorithms were all to be block ciphers, supporting a block size of 128 bits and key sizes of 128, 192, and 256 bits. Such ciphers were rare at the time of the announcement.
  - In the nine months that followed, fifteen different designs were created and submitted from several different countries.
  - These algorithms were investigated by cryptographers. The investigation focused on security, performance, feasibility, and other factors. During the process, some algorithms were eliminated because of their weakness in security, some were eliminated because of the performance or other factors.
  - In August 1999, NIST announced that they were narrowing the field from fifteen to five. All five algorithms, commonly referred to as "AES finalists", were designed by cryptographers considered well-known and respected in the community.
    * Rijndael (Pronuciation "Rain Doll").
    * IDEA (International Data Encryption Algorithm), used by PGP
    * Blowfish (Bruce Schneier).
    * RC5 (Rivest).
    * CAST-128, used by PGP.
  - These finalists went through a further round of intense analysis and cryptanalysis.
  - On October 2, 2000, NIST announced that Rijndael had been selected as the proposed AES.

- Note: The security of AES and many other well-known encryption algorithms (except the one-time pad) has never been proven. They are considered secure because they have been thoroughly investigated by many cryptographers, and so far nobody could break them.

1.3 Attacking a Cryptosystem

- Ciphertext-Only Attack: attackers try to find the plaintext from a ciphertext.
- Known-Plaintext Attack: the attacker has obtained some (plaintext, ciphertext) pairs, and they use these known pairs to find out the other things that they do not know, such as the key and the unknown plaintexts.
- Chosen-Plaintext Attack: attackers can select any plaintext, and ask the encryption system to produce a ciphertext. The attackers can do this for many times. Attakers then try to use the information to break the encryption scheme.

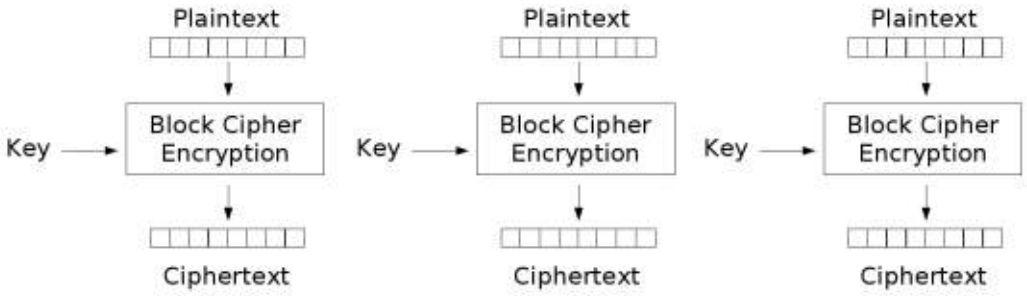
In the real life, all the above three scenarios can happen. Therefore, to become a strong cryptosystem, a cryptosystem should resist all the above three attacks.

- Question: can the classical cryptosystems resist the above attacks?

2 Block Cipher Modes of Operation

Block cipher operates on blocks of fixed length, for example, DES operates on 64-bit blocks, and AES operates on 128-bit blocks. To encrypt a message longer than the block size, the message has to be divided into multiple blocks, so block ciphers can operate on each of them. There are many ways to apply block ciphers on these blocks; These different methods are called *modes of operation*. Details are described in Wikipedia. Figures in this section are from Wikipedia (we only show the encryption part, the decryption part can be easily derived based on the encryption part).

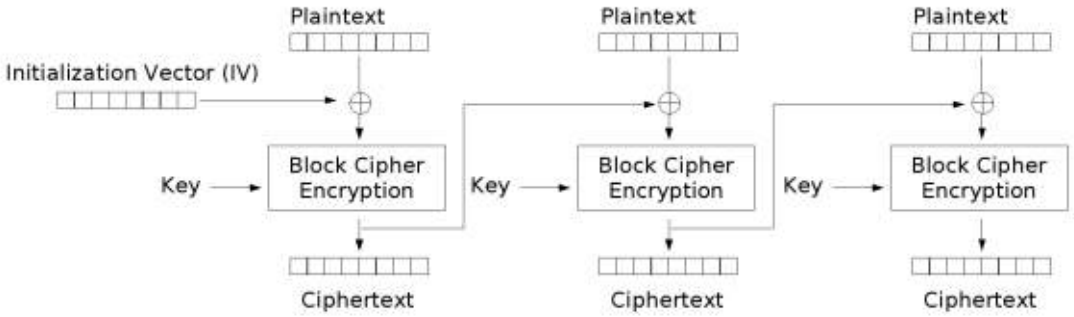
- Electronic Code Book Mode (ECB)



Electronic Codebook (ECB) mode encryption

  - The mode diagram
  - Problem: duplicate blocks and rearrange attack.

- Cipher Block Chaining (CBC)

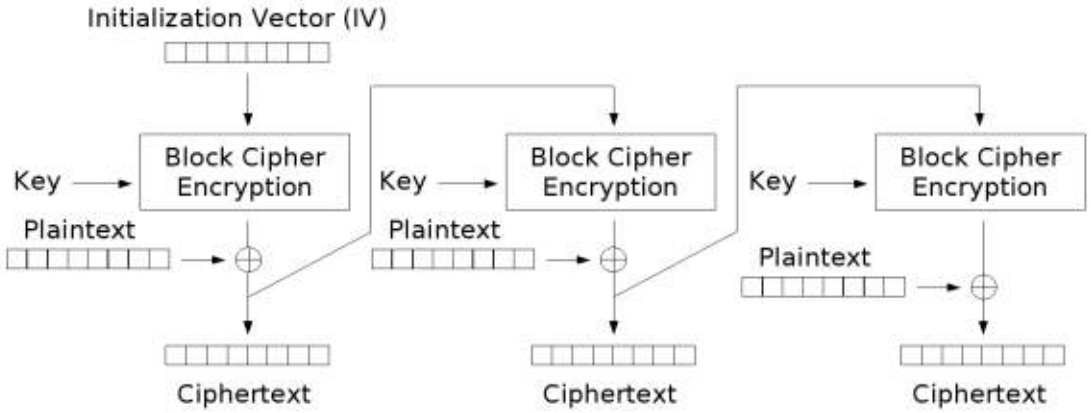


Cipher Block Chaining (CBC) mode encryption

  - IV: Initialization Vector. Does not need to be secret.
  - Even if the same message is sent repeatedly, the ciphertext will be completely different each time due to IV.
  - Has been the most commonly used mode of operation.

  - Parallelization
    - Encryption cannot be parallelized
    - Decryption can be parallelized

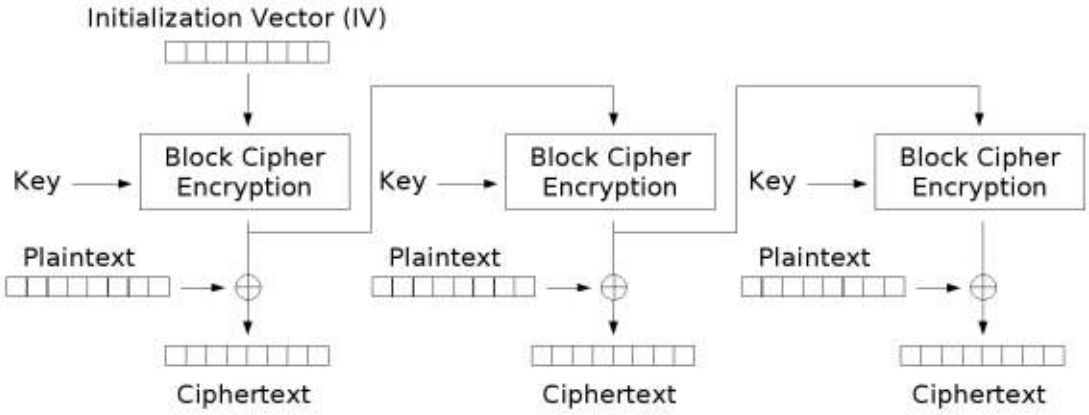
- Cipher feedback (CFB)



Cipher Feedback (CFB) mode encryption

  - Turn a block cipher into a stream cipher
  - Parallelization
    - Encryption cannot be parallelized
    - Decryption can be parallelized

- Output feedback (OFB)

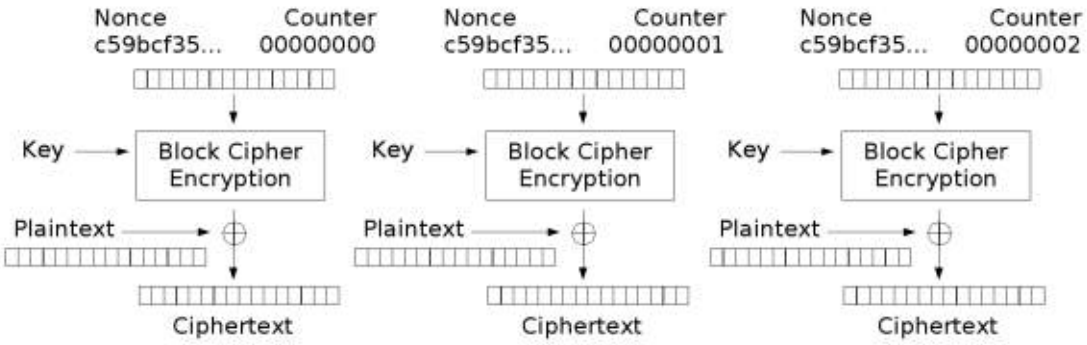


Output Feedback (OFB) mode encryption

  - Also turns a block cipher into a stream cipher
  - Parallelization

  - ∗ Encryption/Decryption cannot be parallelized, but because the plaintext or ciphertext is only used for the final XOR, the block cipher operations may be performed in advance, allowing the final step to be performed in parallel once the plaintext or ciphertext is available.

- Counter (CTR)



Counter (CTR) mode encryption

  - Also turns a block cipher into a stream cipher
  - The counter can be any simple function which produces a sequence which is guaranteed not to repeat for a long time, although an actual counter is the simplest and most popular
  - CTR allows a random access property for decryption

Padding. Because a block cipher works on units of a fixed size, but messages come in a variety of lengths, some modes (mainly CBC) require that the final block be padded before encryption. Several padding schemes exist.

- Add null bytes to the palintext. Care must be taken so the original length of the plaintext can be recovered.
- The original DES method: add a single one bit, followed by enough zero bits to fill out the block; if the message ends on a block boundary, a whole padding block will be added.
- PKCS#5 Standard: each padding octet contains the number of octets that are added in the padding. The following is an example for a 128-bit block cipher that uses the PKCS#5 padding scheme (in the first example, 0x09 is used in the padding, because 9 octets are added in the padding; in the second example, 0x10 is used because 16 octets are added):

```
Original plaintext 1:       0a23bac45092f7
Padded plaintext (PKCS#5): 0a23bac45092f709090909090909090909

Original plaintext 2:       0a23bac45092f793273a7fe9093eaa88
Padded plaintext (PKCS#5): 0a23bac45092f793273a7fe9093eaa08
                           10101010101010101010101010101010
```

- CFB, OFB and CTR modes do not padding. The size of ciphertext is the same as the size of plaintext. This characteristic of stream ciphers makes them suitable for applications that require the encrypted ciphertext data to be the same size as the original plaintext data, and for applications that transmit data in streaming form where it is inconvenient to add padding bytes.

3 One-Way Hash Function

3.1 One-Way Hash Function

- One-way Hash Function
  - Reduce variable-length input to fixed-length (128 or 160 bit) output
  - Notation: M $\Rightarrow$ H(M)
- One Way Property
  - Easy: M $\Rightarrow$ H(M).
  - Computationally Infeasible: H(M) $\Rightarrow$ M.
  - Implications:
    - Even if M_1 and M_2 has just a single bit difference, $H(M_1)$ and $H(M_2)$ will be very different.
    - If you know M and H(M), but after you change a single bit in H(M), you will not be able to find a M' that can generate this modified hash value.
- Collision Free Property: computationally infeasible to find two messages that hash to the same hash value.
  - For m-bit hash, using the brute-force attack, it takes only about $2^{m/2}$ messages, chosen at random, before one would find two with the same value (like the birthday problem).
  - MD5 is broken: it is found not to be collision free!
- Why do we want collision-free property?
  - Example: We can construct M_1 and M_2, such that $h(M_1) = h(M_2)$. The meaning of M_1 and M_2 can be exactly the opposite.
  - M_1 ="Alice owes Kevin \$100". M_2 ="Alice owes Kevin \$1M". Alice signs $h(M_1)$, but not $h(M_2)$. If $h(M_1) = h(M_2)$, Alice will be in trouble.
  - Even if the hash is not collision free, but finding such a meaningful M_1 and M_2 is not easy. However, it might be possible to find $h(M_1, r_1) = h(M_2, r_2)$, where r_1 and r_2 are random numbers.
- Hash Algorithms
  - MD2 (Message Digest) – by Rivest
  - MD3 does exist, but it was superseded by MD4 before it was ever published or used.
  - MD4 (Message Digest) – by Rivest Faster than MD2, but a version of MD4 is found to be weak.
  - MD5 (Message Digest) – by Rivest
    - A little slower than MD4.
    - 128-bit hash
  - SHA (Secure Hash Algorithm)
    - 1993 NIST published SHA
    - 1995 a never published flaw is found in SHA.

    - SHA-1 is proposed: most popular one.
    - SHA-1: 160-bit hash.
    - SHA-2: SHA-256, SHA-384, and SHA-512.
  - MD5, SHA-0, SHA-1 are not collision free
    - Collisions in SHA-0 in 2^{39} operations.
    - Collisions in the full SHA-1 in 2^{52} hash operations, much less than the brute-force attack of 2^{80} operations based on the hash length.
    - The time complexity for finding a collisions in MD5 is 2^{32}.

3.2 MAC: Message Authentication Code

- Message Authentication: making sure that the message is indeed sent by the claimed sender, not by other parties or modified by other parties.
- MAC algorithms
  - Can be constructed from hash functions (e.g. HMAC)
  - Can also be constructed from block cipher algorithms (e.g. OMAC, CBC-MAC and PMAC).
- HMAC: Hashed MAC (or keyed hash)

$$HMC_K(m) = h((K \oplus opad) \ || \ h((K \oplus ipad)||m))$$

  - Need a secret key (SHA, MD5 do not need a secret key)
  - Use one-way Hash function h as a black-box building block. A one-way hash function is conducted by iterating a basic compression function on blocks of data. We denote by B the byte-length of such blocks (e.g. B=64 for MD5 ahd SHA1).
  - ipad = 0x363636...3636 and opad = 0x5c5c5c...5c5c. Their length is B, i.e. the size of hash block. The values of ipad and opad are not critical to the security of the algorithm, but were defined in such a way to have a large Hamming distance from each other and so the inner and outer keys will have fewer bits in common.
  - K is a secret key padded to length B with extra zeros. If K is longer than B byptes, we will use hash(K) as the key.
  - HMAC-MD5, HMAC-SHA1.
  - Need to know the secret key in order to verify.

3.3 Applications of One-Way Hash Function

- **Pseudorandom number generator (PRNG):** One-way hash function can be used for this purpose. This is done by combining a (secret) random seed with a counter and hashing it: $h(seed, c)$, $h(seed, c+1)$, $h(seed, c+2)$, and so on.
- **Stream Cipher:** Use one-way hash function as a pseudorandom number generator to genrate a stream of psuedorandom numbers. XOR the plaintext with this stream of numbers. In decryption, the same stream can be constructed using the counter and the seed (the seed is the encryption key).

- **Password Verification:** To verify passwords, we do not need to store plain-text passwords in a database. We can store the hashes of passwords in the database. This way, even if the whole database is compromised by adversaries, the password information is still safe if a strong one-way hash function was used.
- **Making Commitment:** Alice and Bob plays a simple online game. Each person provides an integer. If the sum of the two integers is odd, Alice wins; otherwise Bob wins. However, whoever gives out the number first will definitely lose. Using one-way hash function, they can commit their numbers first, and then release the numbers to each other. Nobody can change his/her mind after the commitment, unless he/she can find a collision of the hash function.
- **Detecting Changes:** To ensure system security, it is necessary to routinely check whether the important files/configurations are modified. We can use one-way hash to achieve this. This is the main idea behind the Tripwire idea.
- **One-way hash chain**
  - The S/KEY one-time password scheme.
  - Broadcast authentication
- **Merkle Tree**
  - Timestamping a document
    - Publish hash in a magazine or newspaper.
    - One hash per document: expensive.
    - One hash per 1000 documents: cost saving.
    - Using Merkle Tree can achieve such a cost saving.
  - Broadcast authentication in lossy channels
    - Signing each packet is expensive
    - Hashing all of them together and then hash the result cannot tolerate the loss of packet.
    - Using Merkle Tree can solve this problem.

4 Public-Key Cryptography

4.1 History

- First asymmetric key algorithm was invented, secretly, by Clifford Cocks (then a recent mathematics graduate and a new staff member at GCHQ in the UK) early in the 1970s.
- 1976, Diffie and Hellman postulated this system without demonstrating that such algorithms exist.
- 1978, Rivest, Shamir and Adleman all then at MIT invented RSA, which is a reinvention of Cocks scheme.
- Since then, several other asymmetric key algorithms have been developed, but the most widely known remains Cocks/RSA.
- Another algorithm is ElGamal (Taher ElGamal), which relies on the (similar and related) difficulty of the discrete logarithm problem.
- A third is a group of algorithms based on elliptic curves, first discovered by Neal Koblitz in the mid '80s.
- NSA has also claimed to have invented public-key cryptography, in the 1960s; however, there is currently (as of 2004) little supporting evidence for their claims.
- Merkle-Hellman (MH) was one of the earliest public key cryptosystems invented by Ralph Merkle and Martin Hellman in 1978. Although its ideas are elegant, and far simpler than RSA, it has been broken. (Merkle-Hellman Knapsacks).
- Stories behind RSA: Steven Levy's Crypto book

4.2 Diffie-Hellman Key Exchange

- Diffie-Hellman Key Exchange
  - The algorithm was first published by Whitfield Diffie and Martin Hellman in 1976
  - Discrete Logarithms in a finite field is a hard problem: find x where $a^x = b \pmod{n}$
  - The protocol
    1. Alice and Bob agree on a finite cyclic group G of size p and a generating element g in G.
    2. Alice sends g^x mod p to Bob.
    3. Bob sends g^y mod p to Alice.
    4. Alice computes $(g^y)^x$ mod p.
    5. Bob computes $(g^x)^y$ mod p.
    6. Both Alice and Bob get g^{xy} mod p.
  - g can be small.
  - x and y must be large.
- Turn Diffie-Hellman Key Exchange to a Public-Key System
  - Public Key: Alice publishes g, p, and (g^x mod p) as her public key.
  - Private Key: g, p, and x.

  - Encryption:
    1. Bob generates y, and generates a key $K = (g^x)^y \bmod p$.
    2. Bob encrypts M using the key K and a symmetric key encryption method, such as AES.
    3. Bob sends the ciphertext and $g^y \bmod p$ to Alice.
  - Decryption:
    1. Alice generates $K = (g^y)^x \bmod p$.
    2. Alice decrypts the ciphertext using K.

- ElGamal algorithm: this algorithm is similar to the above algorithm, but it does not rely on any symmetric key encryption scheme.

  1. Let $h = g^x \bmod p$.
  2. Public key: (p, g, h)
  3. Private key: x
  4. Encryption: generate a random k, let $c_1 = g^k \bmod p$, $c_2 = m * h^k \bmod p$.
  5. Decryption: $c_2/c_1^x \bmod p$. It should be noted that $c_2/c_1^x = m * h^k/g^{kx} = m \bmod p$.

4.3 RSA Algorithm

- Mathematics background for RSA algorithm
  - Extended Euclidean algorithm: Given x, find y, such that $x \cdot y = 1 \bmod m$. The Extended Euclidean algorithm can efficiently find the solution to this problem.
  - Euler's theorem: For any number a relatively prime to $n = pq$, $a^{(p-1)(q-1)} = 1 \bmod pq$.
    * Why is this very useful?
    * Let $z = k(p-1)(q-1) + r$, we have $a^z = a^{k(p-1)(q-1)} * a^r = a^r \bmod pq$.
    * In other words: If $z = r \bmod (p-1)(q-1)$, then $a^z = a^r \bmod pq$.
    * Special case: If $z = 1 \bmod (p-1)(q-1)$, then $a^z = a \bmod pq$.
    * We can use Euler's theorem to simply $a^z \bmod pq$.

- RSA Algorithm.
  - Let $n = pq$, where p and q are two large primes.
  - Public key (e, n), where e is relative prime to $(p-1)(q-1)$.
  - Private key (d, n), such that $ed = 1 \bmod (p-1)(q-1)$. d can be calculated using the Extended Euclidean algorithm.
  - Encryption: $c = m^e \bmod n$.
  - Decryption: $c^d = (m^e)^d = m^{ed} \bmod n$.

- Security of RSA: depends on the hardness of factoring: factoring $n = p * q$ is hard when n is large.

- RSA Example
  - Let $n = 22 = 2 * 11$.
  - Let $e = 3$, find d, such that $e * d = 1 \bmod 10$. We get $d = 7$.

  - Encrypting $M = 7$: $7^3 = 49 * 7 = 5 * 7 = 13$ mod 22.
  - Decrypting: $13^7 = 13^2 * 13^2 * 13^2 * 13 = (15 * 15) * (15 * 13) = 5 * 19 = 7$ mod 22
  - Sign $M = 9$: $9^7 = 92 * 92 * 92 * 9 = 15$ mod 22.

- RSA Performance Issues:
  - Exponentiating with big numbers: For example, let us compute 123^{32} mod 678. We do not need to do 32 multiplies, instead, we do the following:

$$\begin{aligned} 123^2 &= 15129 = 213 \ mod \ 678, \\ 123^4 &= 213^2 = 621 \ mod \ 678, \\ 123^8 &= 621^2 = 537 \ mod \ 678, \\ 123^{16} &= 537^2 = 219 \ mod \ 678, \\ 123^{32} &= 219^2 = 501 \ mod \ 678. \end{aligned}$$

  - Similarly, we can easily compute 123^{54} mod n. $54 = (1100110)_2 = (((((1) * 2 + 1) * 2) * 2 + 1) * 2 + 1) * 2$. Therefore, $123^{54} = (((((123)^2 * 123)^2)^2 * 123)^2 * 123)^2$. We will do module operations at each step, whenever the intermediate results are larger than n. We will not compute a huge number 123^{54} and then do one module operation at the end. The total cost is 8 multiplies and 8 divides (the divides are caused by the module operations).
  - Performance: if the exponent is 512 bit, the number of multiplication and divides is linear to 512.

- Choice of e in RSA.
  - e does not need to be a large number.
  - In practice, 3 and 65537 are often selected as the value of e.

- Efficiency of RSA: very costly
  - 100 times slower than DES (software)
  - 1000 times slower than DES (hardware)

5 Digital Signature

5.1 Digital Signature Algorithms

- Motivation of digital signature
  - Physical signature
  - Properties: Authenticity, unforgeable, not reusable, unalterable, can't be repudiated.

- RSA Signature Scheme
  - Public key (verifying key): (e, n).
  - Private key (signing key): (d, n).
  - Sign: M^d mod n.

  - Verify: $(M^d)^e \bmod n$. The result should be equal to M if the signature is authentic.

- DSA (Digital Signature Algorithm):
  - 1991, Proposed by NIST as a DSS (Digital Signature Standard)
  - Criticism from RSA and its supporters
  - Developed by NSA
  - Royalty-free
  - DSA is slower than RSA
- Avoid reusing digital signature: Sign with timestamps.
- Sign the hash. In practice, we only sign the hash of the message, not the message itself, because the message may be large, and signing is quite slow for large message.

5.2 Public-Key Certificate

- Associating public keys with identities.
  - The Man-in-the-middle attack
    - How to make sure Bob's Public Key is really Bob's?
    - Binding the identity with the public key
  - Hierarchical certificate authority (eg, X.509),
  - A local trust model (eg, SPKI),
  - A statistical web of trust (eg, PGP and GPG).
- Certificates
  - Consist of: holder's id, holder's public key, CA's signature, expiration date
  - X.509 certificate
    - The certificate can be obtained from a public known database
    - Tree Structure
    - If you trust one, you can verify all of them.

```
A Sample X.509 Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
```

```
            OU=FreeSoft, CN=www.freesoft.org/emailAddress=...
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                    33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                    66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                    70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                    16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                    c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                    8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                    d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                    e8:35:1c:9e:27:52:7e:41:8f
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
        92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
        ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
        d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
        0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
        5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
        8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
        68:9f
```

---

6 Applications of Public-Key Encryption and Digital Signature

- Key Exchange using public keys
  - Goal: Alice and Bob want to agree upon a key K, which can be used as session key.
  - Session key: only exists for the duration of the communication.
  - Alice sends her public key to Bob, and Bob sends his to Alice.
- Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL).
  - Based on PKI.
  - Provide security for communications over networks such as the Internet.
  - Widely used in applications like web browsing, electronic mail, instant messaging and voice-over-IP (VoIP).
  - A prominent use of TLS is for securing World Wide Web traffic carried by HTTP to form HTTPS.
  - `OpenSSL` is an open source implementation of the SSL and TLS protocols. The core library (written in the C programming language) implements the basic cryptographic functions and provides various utility functions.

7 Implementing Cryptography Correctly is Not Easy: Case Studies

7.1 Diebold Electronic Voting System: Case Study

Background: Municipalities and states throughout the U.S. are adopting paperless electronic voting systems to replace outdated punch-card and mechanical voting systems. A number of different vendors have designed such kind of e-voting systems. A team of researchers at Johns Hopkins University conducted a security analysis on one of the e-voting systems, the Diebold system. The report pointed out a number of places where cryptography is implemented incorrectly. We will use this report as our case study to understand what might go wrong when integrating cryptography in a real-world system. We refer to this report as the *JHU Report* in this lecture note.

- **Key management**

  JHU Report: In the Diebold code we analyzed, both the keys for the smartcard and the keys used to encrypt the votes were static entries in the source code. This means that the same keys are used on every voting device. Thus, an attacker who was able to compromise a single voting device would have access to the keys for all other voting devices running the same software.

  ```
  #define DESKEY ((des_key*)"F2654hD4")
  ```

  From the CVS logs, we see this particular key has been used without change since December 1998, when the CVS tree for AccuVote-TS version 3 began, and we assume that the key was in use much before 14 that. Although Jones reports that the vendor may have been aware of the key management problems in their code since at least 1997 [16, 17], our findings show that the design flaw was never addressed.

- **Encryption algorithm**

  JHU Report: A second set of problems has to do with the way that the Diebold code encrypts the votes and audit logs. The files that hold the votes are encrypted using the Data Encryption Standard (DES) algorithm in CBC mode. There are problems with the use of both DES and the CBC mode, as we describe below.

  In their response to "allegation 44", Diebold states that "there are stronger forms of compression than DES, but the authors' implication that the keys can be recovered 'in a short time' is deliberately misleading." We assume that Diebold meant to claim that there are stronger encryption algorithms available, as DES is not a compression algorithm.

- **Integrity**

  JHU Report: Instead of using such a MAC, the Diebold code uses a non-cryptographic checksum called a CRC to detect whether a file has been tampered with. This is completely insecure as is discussed on page 15 of our paper. The use of CRCs instead of MACs has long been documented in the security literature as a very serious mistake.

  In Diebold system, before being encrypted, a 16-bit cyclic redundancy check (CRC) of the plaintext data is computed. This CRC is then stored along with the ciphertext in the file and verified whenever the data is decrypted and read. This process in handled by the ReadRecord and WriteRecord functions in TSElection/RecordFile.cpp. Since the CRC is an unkeyed, public function, it does not provide any meaningful integrity protection for the data. In fact, by storing it in an unencrypted form, the purpose of encrypting the data in the first place (leaking no information about the contents of the

plaintext) is undermined. Standard industry practice would be to first encrypt the data to be stored and then to compute a keyed cryptographic checksum (such as HMAC-SHA1) of the ciphertext. This cryptographic checksum could then be used to detect any tampering with the plaintext. Note also that each entry has a timestamp, which can be used to detect reordering, although sequence numbers should also be added to detect record deletion.

- **Mode**

  JHU Report: We note that "DES is being used in CBC mode which requires an initialization vector to ensure its security." We go on to show that the Diebold code does not provide the necessary initialization vectors. A detailed explanation of this problem is highly technical; we refer the interested reader to A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation.

  Second, DES is being used in CBC mode which requires a random initialization vector to ensure its security. The implementation here always uses zero for its IV. This is illustrated by the call to DesCBCEncrypt in TSElection/RecordFile.cpp; since the second to last argument is NULL, DesCBCEncrypt will use the all-zero IV. To correctly implement CBC mode, a source of strong random numbers must be used to generate a fresh IV for each encryption. Suitably strong random numbers can be derived from many different sources, ranging from custom hardware to accumulated observations of user behavior.

- **Random number**

  JHU Report: While the voter's identity is not stored with the votes, each vote is given a serial number ... generated by a linear congruential random number generator ... seeded with static information.

  Diebold Response: There is no need for "security" here. The only intent of this code is to pseudo-randomize the order of ballots for purposes of display and reporting, as required in some states.

  Jones (Doug Jone from the University of Iowa also responded to Diebold's reponse): Diebold is wrong. There is need for security here. If the sequence of pseudo-random numbers is known, and the sequence in which voters actually entered the booth has been recorded (as a poll-watcher can easily do), then we can recover any particular voter's ballot from the report of individual ballots. This allows an insider working at election central to check this report (I'd use a pocket camera to take photos of the report), in cooperation with a poll watcher, to confirm whether the paid voters have earned their pay by voting the required way. Vote buying schemes that rely on insiders at the vote count cooperating with poll-watchers date back many years, and therefore, strong randomization schemes are justified here! I've worked as a poll watcher, I know that perfect records are hard to keep, but I also know that I can correct my records if I can talk a few voters into signing their ballots with pre-selected write-in votes or funny patterns of yes-no votes on the judicial retention ballot.

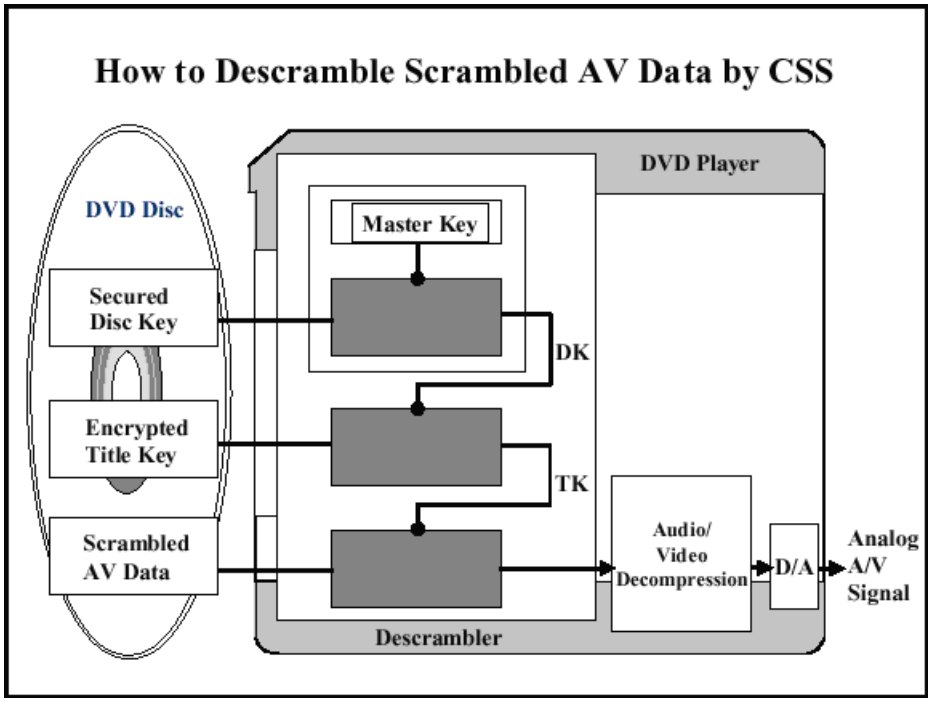
- **Smartcards**

  JHU Report: Upon reviewing the Diebold code, we observed that the smartcards do not perform any cryptographic operations. This, in and of itself, is an immediate red flag. One of the biggest advantages of smartcards over classic magnetic-stripe cards is the smartcards ability to perform cryptographic operations internally, and with physically protected keys. Because of a lack of cryptography, there is no secure authentication of the smartcard to the voting terminal. This means that nothing prevents an attacker from using his or her own homebrew smartcard in a voting terminal.

  One might naturally wonder how easy it would be for an attacker to make such a homebrew smartcard. First, we note that user-programmable smartcards and smartcard readers are available commercially

over the Internet in small quantities and at reasonable prices. Second, an attacker who knows the protocol spoken between voting terminals and legitimate smartcards could easily implement a homebrew card that speaks the same protocol.

7.2 DVD Protection: Case Study



- The process:
  1. Each DVD player has a master key, which is unique to the DVD player manufacturer. This key is called the *player key*.
  2. The palyer reads an encrypted *disk key* from the DVD, and uses the *player key* to decrypt the disk key. Since there are many player keys out there (each manufacture has one), the DVD must contain a copy of the encrypted disk key for each player key.
  3. The player reads the encrypted title key for the file to be played. The DVD will likely contain multiple files, each with its own title key. The player uses the descrypted disk key to decrypt the title key.
  4. The player used the title key to descrypt the content.
- Encryption algorithm: Content Scrambling System (CSS).
  - **Its security depends on its secrecy:** in 1999, Jon Johansen (with another two anonymous people) disassembled a software DVD player to uncover the descrambling algorithm. They then wrote and released a C code called DeCSS.
  - **The encryption key is only 40-bit and not all possible 40-bit numbers can be keys:** A high-end home computer in 1999 running optimized code could brute-force it within 24 hours, and modern computers can now brute-force it in a few seconds or less.
- Software player has its own unlock key. This is where the problem is.

- The key must be stored in the memory.
- The contents must appear in the memory unencrypted.
- Users can reverse engineering the code.
- Users can debug the code and find the keys in the memory.
- Making reverse engineering a crime is not going to be a solution. Unfortunately, this is where the DVD industry is going.

7.3 Mistakes in Encryption: Not using cryptography when it is needed

- The Wall Street Journal (December 17, 2009): **Insurgents Hack U.S. Drones.**

  > **Militants in Iraq have used $26 off-the-shelf software to intercept live video feeds from U.S. Predator drones**, potentially providing them with information they need to evade or monitor U.S. military operations.
  > Senior defense and intelligence officials said Iranian-backed insurgents intercepted the video feeds by taking advantage of an unprotected communications link in some of the remotely flown planes' systems. Shiite fighters in Iraq used software programs such as SkyGrabber – available for as little as $25.95 on the Internet – to regularly capture drone video feeds, according to a person familiar with reports on the matter.
  > U.S. officials say there is no evidence that militants were able to take control of the drones or otherwise interfere with their flights. Still, the intercepts could give America's enemies battlefield advantages by removing the element of surprise from certain missions and making it easier for insurgents to determine which roads and buildings are under U.S. surveillance.
  > .....
  > The potential drone vulnerability lies in an unencrypted downlink between the unmanned craft and ground control. **The U.S. government has known about the flaw since the U.S. campaign in Bosnia in the 1990s, current and former officials said. But the Pentagon assumed local adversaries wouldn't know how to exploit it**, the officials said."
  > .....
  > Officials stepped up efforts to prevent insurgents from intercepting video feeds after the July incident. **The difficulty, officials said, is that adding encryption to a network that is more than a decade old involves more than placing a new piece of equipment on individual drones.** Instead, many components of the network linking the drones to their operators in the U.S., Afghanistan or Pakistan have to be upgraded to handle the changes.
  > .....
  > Today, the Air Force is buying hundreds of Reaper drones, a newer model, **whose video feeds could be intercepted in much the same way** as with the Predators, according to people familiar with the matter. A Reaper costs between $10 million and $12 million each and is faster and better armed than the Predator.

7.4 Mistake in Encrytion: Inventing your own encryption algorithm and keep it secret

- Good encryption algorithms such as DES, AES, and Blowfish take many years to develop by smart minds who specialize on cryptography, and then they were scrutinized by many other smart minds. The reason why we see these names is because nobody has broken them so far. All the bad ones have already been eliminated.

- The selection process for AES: `http://en.wikipedia.org/wiki/Advanced_Encryption_Standard_process`
- If you ever want to invent your own encryption algorithm for your software within a few days or even a few months, think about what those algorithms have gone through. If this does not stop you, then look at the following cases:
  - Case 1: DVD encryption `http://www.schneier.com/essay-193.html`.
  - Case 2: WEP: is a deprecated algorithm to secure IEEE 802.11 wireless networks. Beginning in 2001, several serious weaknesses were identified by cryptanalysts with the result that today a WEP connection can be cracked with readily available software within minutes.
  - Case 3: The GSM encryption algorithm was broken. `http://www.gsmarena.com/the_gsm_encryption_algorithm_was_broken-news-1347.php`.
  - Case 4: 3G GSM encryption cracked in less than two hours `http://www.engadget.com/2010/01/15/3g-gsm-encryption-cracked-in-less-than-two-hours/`
  - Case 5: Philip Zimmermann, Beware of Snake Oil. `http://www.philzimmermann.com/EN/essays/SnakeOil.html`.
- Keeping your algorithm secret is not going to help much: reverse engineering.
  - Why Security-Through-Obscurity Won't Work. `http://slashdot.org/features/980720/0819202.shtml`
  - Principle

    Principle: Security of encryption should be based on the secrecy of the keys, not the secrecy of the algorithm.
- If you still want to try, then you are either a cryptographer who simply wants to develop a better encryption algorithm, or you are simply crazy.

7.5 Mistake in Encryption: Bad Key Management

- Hardcode the keys in the program, e.g. Diebold.
- Principles