Information Security and Cryptography for Communications and Network

Agenda

- Classical Cryptography
- Shannon's Theory
- The Data Encryption Standard (DES)
- The RSA System and Factoring
- Other Public-key Cryptography
- Signature Schemes

Agenda (2)

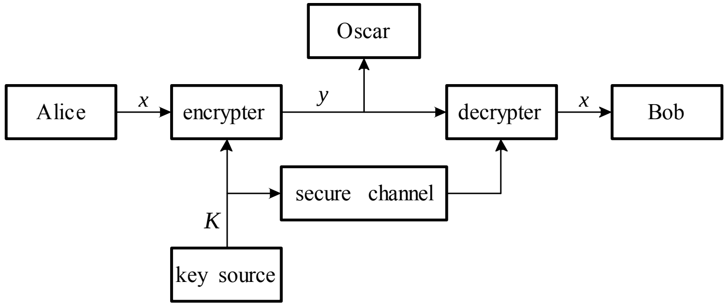
- Hash Functions
- Key Distribution and Key Agreement
- Identification Schemes
- Authentication Codes
- Secret Sharing Schemes
- Pseudo-random Number Generation
- Zero-knowledge Proofs
- Power Analysis

Cryptosystem

A cryptosystem is a five-tuple (***P***, ***C***, ***K***, ***E***, ***D***), where the following conditions are satisfied:

1. ***P*** is a finite set of possible plaintexts
2. ***C*** is a finite set of possible cipher-texts
3. ***K***, the key-space, is a finite set of possible keys

4. For each $K \in \mathcal{K}$, there is an encryption rule $e_K \in \mathcal{E}$ and a corresponding decryption rule $d_K \in \mathcal{D}$. Each e_K: $\boldsymbol{P} \rightarrow \boldsymbol{C}$ and d_K: $\boldsymbol{C} \rightarrow \boldsymbol{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext $x \in \mathcal{P}$.



The Communication Channel

Let $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$. For $0 \le K \le 25$, define

$$e_K(x) = x + K \bmod 26$$

and

$$d_K(y) = y - K \bmod 26$$

$(x, y \in \mathbb{Z}_{26})$.

Shift Cipher

Let $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$. $\boldsymbol{\mathcal{K}}$ consists of all possible permutations of the 26 symbols $0, 1, \ldots, 25$. For each permutation $\pi \in \mathcal{K}$, define

$$e_\pi(x) = \pi(x),$$

and define

$$d_\pi(y) = \pi^{-1}(y),$$

where π^{-1} is the inverse permutation to π.

Substitution Cipher

Shannon's Theory

- Computational Security (RSA, etc.)
- Unconditional Security (based on Shannon Information Theory)

Suppose **X** and **Y** are random variables. We denote the probability that **X** takes on the value x by $p(x)$, and the probability that **Y** takes on the value y by $p(y)$. The joint probability $p(x, y)$ is the probability that **X** takes on the value x and **Y** takes on the value y.

The conditional probability $p(x|y)$ denotes the probability that **X** takes on the value x given that **Y** takes on the value y. The random variables **X** and **Y** are said to be independent if $p(x, y) = p(x)\, p(y)$ for all possible values x of **X** and y of **Y**.

Joint probability can be related to conditional probability by the formula

$$p(x, y) = p(x|y)\, p(y).$$

Interchanging x and y, we have that

$$p(x, y) = p(y|x)\, p(x).$$

From these two expressions, we immediately obtain the following result, which is known as Bayes' Theorem.

Bayes' Theorem

If $p(y) > 0$, then

$$p(x|y) = \frac{p(x)\, p(y|x)}{p(y)}.$$

Spurious Keys and Unicity Distance

Let (***P***, ***C***, ***K***, ***E***, ***D***) be a cryptosystem. Then

$$H(\mathbf{K}|\mathbf{C}) = H(\mathbf{K}) + H(\mathbf{P}) - H(\mathbf{C}).$$

First, observe that $H(\mathbf{K}, \mathbf{P}, \mathbf{C}) = H(\mathbf{C}|\mathbf{K}, \mathbf{P}) + H(\mathbf{K}, \mathbf{P})$.

Now, the key and plaintext determine the ciphertext uniquely, since $y = e_K(x)$.

This implies that $H(\mathbf{C}|\mathbf{K}, \mathbf{P}) = 0$. Hence, $H(\mathbf{K}, \mathbf{P}, \mathbf{C}) = H(\mathbf{K}, \mathbf{P})$. But **K** and **P** are independent, so $H(\mathbf{K}, \mathbf{P}) = H(\mathbf{K}) + H(\mathbf{P})$. Hence,

$$H(\mathbf{K}, \mathbf{P}, \mathbf{C}) = H(\mathbf{K}, \mathbf{P}) = H(\mathbf{K}) + H(\mathbf{P}).$$

Entropy of a natural language

Suppose L is a natural language.

The entropy of L is defined to be the quantity

$$H_L = \lim_{n \to \infty} \frac{H(\mathbf{P}^n)}{n}$$

and the redundancy of L is defined to be

$$R_L = 1 - \frac{H_L}{\log_2 |P|}$$

H_L measures the entropy per letter of the language L. A random language would have entropy $\log_2|\boldsymbol{P}|$.

So the quantity R_L measures the fraction of ``excess characters," which we think of as redundancy.

Unicity distance

The unicity distance of a cryptosystem is defined to be the value of n, denoted by n_0, at which the expected number of spurious keys becomes zero; i.e., the average amount of ciphertext required for an opponent to be able to uniquely compute the key, given enough computing time.

$$n_0 \approx \frac{\log_2 |K|}{R_L \log_2 |P|}$$

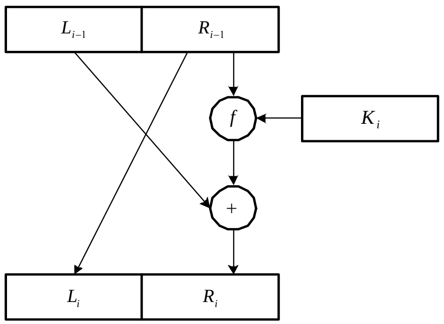
DES

1. Given a plaintext x, a bit-string x_0 is constructed by permuting the bits of x according to a (fixed) initial permutation IP. We write $x_0 = \mathrm{IP}(x) = L_0R_0$, where L_0 comprises the first 32 bits of x_0 and R_0 the last 32 bits.
2. 16 iterations of a certain function are then computed. We compute L_iR_i, $1 \le i \le 16$, according to the following rule:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

where $\oplus$ denotes the exclusive-or of two bit-strings. f is a function that we will describe later, and K_1, K_2, …, K_{16} are each bit-strings of length 48 computed as a function of the key K. (Actually, each K_i is a permuted selection of bits from K.) $K_1, K_2, \ldots, K_{16}$ comprises the *key schedule*.
One round of encryption is depicted in Figure 3.1

3. Apply the inverse permutation IP^{-1} to the bit-string $R_{16}\ L_{16}$, obtaining the cipher-text y.
That is, $y = \mathrm{IP}^{-1}(R_{16}\ L_{16})$. Note the inverted order of L_{16} and R_{16}.



One round of DES encryption

Public-key Cryptography

- RSA: Difficulty of factoring large integers
- Knapsack: Difficulty of the subset sum problem
- McEliece: Difficulty of decoding a linear code
- ElGamal: Difficulty of the discrete logarithm problem for finite fields
- Elliptic Curve: Work in the domain of elliptic curves rather than finite fields

1. $z = 1$
2. for $i = \ell - 1$ down to 0 do
3. $z = z^2 \bmod n$
4. if $b_i = 1$ then
$$z = z \times x \bmod n$$

The square-and-multiply algorithm to compute $x^b \bmod n$

Let $n = pq$, where p and q are primes. Let $\mathbf{P} = \mathbf{C} = Z_n$, and define

$$K = \{(n, p, q, a, b) : n = pq,\ p, q \text{ prime}, ab \equiv 1 (\bmod \phi(n))\}$$

For $K = (n, p, q, a, b)$, define

$$e_K(x) = x^b \bmod n$$

and

$$d_K(y) = y^a \bmod n$$

$(x, y \in Z_n)$ The values n and b are public, and the values p, q, a are secret.

RSA Cryptosystem

1. Bob generates two large primes, p and q
2. Bob computes $n = pq$ and $\phi(n) = (p-1)(q-1)$
3. Bob chooses a random $b (1 < b < \phi(n))$ such that $\gcd(b, \phi(n)) = 1$
4. Bob computes $a = b^{-1} \bmod \phi(n)$ using the Euclidean algorithm
5. Bob publishes n and b in a directory as his public key.

Setting up RSA

ElGamal Cryptosystem and Discrete Logs

Problem Instance

$I = (p, \alpha, \beta)$, where p is prime, $\alpha \in Z_p$ is a primitive element, and $\beta \in Z_p^*$.

Objective

Find the unique integer a, $0 \le a \le p-2$ such that

$$\alpha^a \equiv \beta (\bmod p)$$

We will denote this integer a by $\log_\alpha \beta$.

Let p be a prime such that the discrete log problem in Z_p is intractable, and let $\alpha \in Z_p{}^*$ be a primitive element.
Let $\mathcal{P} = Z_p{}^*, \mathcal{C} = Z_p{}^* \times Z_p{}^*$, and define

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a (\text{mod } p)\}$$

The values p, α and β are public, and a is secret.
For $K = (p, \alpha, a, \beta)$, and for a (secret) random number $k \in Z_{p-1}$, define

$$e_K(x, k) = (y_1, y_2)$$

where

$$y_1 = \alpha^k \text{ mod } p$$

and

$$y_2 = x\beta^k \text{ mod } p$$

For $y_1, y_2 \in Z_p{}^*$, define

$$d_K(y_1, y_2) = y_2(y_1{}^a)^{-1} \text{ mod } p$$

Let G be a generating matrix for an $[n, k, d]$ Goppa code $\mathbf{C}$, where $n = 2^m$, $d = 2t + 1$ and $k = n - mt$. Let S be a matrix that is invertible over Z_2, let P be $n \times n$ an permutation matrix, and let $G' = SGP$. Let $\boldsymbol{P} = (Z_2)^k$, $\boldsymbol{C} = (Z_2)^n$, and let

$$\mathcal{K} = \{(G, S, P, G')\}$$

where G, S, P, and G' are constructed as described above.
G' is public, and G, S, and P are secret.
For $K = (G, S, P, G')$, define $e_K(\mathbf{x}, \mathbf{e}) = \mathbf{x}G' + \mathbf{e}$

McEliece Cryptosystem

where $\mathbf{e} \in (Z_2)^n$ is a random vector of weight t.
Bob decrypts a ciphertext $\mathbf{y} \in (Z_2)^n$ by means of the following operations:

1. Compute $\mathbf{y}_1 = \mathbf{y}P^{-1}$.
2. Decode $\mathbf{y}_1$, obtaining $\mathbf{y}_1 = \mathbf{x}_1 + \mathbf{e}_1$, where $\mathbf{x}_1 \in \mathbf{C}$.
3. Compute $\mathbf{x}_0 \in (\mathbf{Z}_2)^k$ such that $\mathbf{x}_0 G = \mathbf{x}_1$.
4. Compute $\mathbf{x} = \mathbf{x}_0 S^{-1}$.

Signature Schemes

A signature scheme is a five-tuple (***P***, ***A***, ***K***, ***S***, ***V***), where the following conditions are satisfied:

1. ***P*** is a finite set of possible messages

2. ***A*** is a finite set of possible signatures

3. ***K***, the key-space, is a finite set of possible keys

4. For each $K \in \mathcal{K}$, there is a signing algorithm $sig_K \in \mathcal{S}$ and a corresponding verification algorithm $ver_K \in \mathcal{V}$. Each sig_K: $\boldsymbol{P} \rightarrow \boldsymbol{A}$ and $ver_K : \mathcal{P} \times \mathcal{A} \rightarrow \{\text{true, false}\}$ are functions such that the following equation is satisfied for every message $x \in \mathcal{P}$ and for every signature $y \in \mathcal{A}$:

$$ver(x, y) = \begin{cases} \text{true} & if \quad y = sig(x) \\ \text{false} & if \quad y \neq sig(x) \end{cases}$$

Let $n = pq$, where p and q are primes. Let $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$, and define

$$\mathcal{K} = \{(n, p, q, a, b) : n = pq, p, q \text{ prime}, ab \equiv 1 \pmod{\phi(n)}\}.$$

The values n and b are public, and the values p, q, a are secret.

For $K = (n, p, q, a, b)$, define

$$sig_K(x) = x^a \bmod n$$

and

$$ver_K(x, y) = \text{true} \Leftrightarrow x \equiv y^b \pmod{n}$$

$(x, y \in \mathbb{Z}_n)$.

RSA Signature Scheme

Let p be a prime such that the discrete log problem in $\mathbb{Z}_p$ is intractable, and let $\alpha \in \mathbb{Z}_p{}^*$ be a primitive element. Let $\mathcal{P} = \mathbb{Z}_p{}^*$, $\mathcal{A} = \mathbb{Z}_p{}^* \times \mathbb{Z}_{p-1}$, and define

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, α and β are public, and a is secret.

For $K = (p, \alpha, a, \beta)$, and for a (secret) random number $k \in \mathbb{Z}_{p-1}{}^*$, define

$$sig_K(x, k) = (\gamma, \delta),$$

where

$$\gamma = \alpha^k \bmod p$$

and

$$\delta = (x - a\gamma)k^{-1} \bmod (p-1).$$

For $x, \gamma \in \mathbb{Z}_p{}^*$ and $\delta \in \mathbb{Z}_{p-1}$, define

$$ver_K(x, \gamma, \delta) = \text{true} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}.$$

ElGamal Signature Scheme

Let p be a 512-bit prime such that the discrete log problem in $\mathbb{Z}_p$ is intractible, and let q be a 160-bit prime that divides $p-1$. Let $\alpha \in \mathbb{Z}_p{}^*$ be a qth root of 1 modulo p. Let $\mathcal{P} = \mathbb{Z}_q{}^*$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$, and define

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, q, α and β are public, and a is secret.

For $K = (p, q, \alpha, a, \beta)$, and for a (secret) random number k, $1 \leq k \leq q-1$, define

$$sig_K(x, k) = (\gamma, \delta),$$

where

$$\gamma = (\alpha^k \bmod p) \bmod q$$

and

$$\delta = (x + a\gamma)k^{-1} \bmod q.$$

For $x \in \mathbb{Z}_q{}^*$ and $\gamma, \delta \in \mathbb{Z}_q$, verification is done by performing the following computations:

$$e_1 = x\delta^{-1} \bmod q$$
$$e_2 = \gamma\delta^{-1} \bmod q$$
$$ver_K(x, \gamma, \delta) = \text{true} \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma.$$

DSS (Digital Signature Standard)

Let $p = 2q + 1$ be a prime such that q is prime and the discrete log problem in $\mathbb{Z}_p$ is intractible. Let $\alpha \in \mathbb{Z}_p{}^*$ be an element of order q. Let $1 \leq a \leq q-1$ and define $\beta = \alpha^a \bmod p$. Let G denote the multiplicative subgroup of $\mathbb{Z}_p{}^*$ of order q (G consists of the quadratic residues modulo p). Let $\mathcal{P} = \mathcal{A} = G$, and define

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, α and β are public, and a is secret.

For $K = (p, \alpha, a, \beta)$ and $x \in G$, define

$$y = sig_K(x) = x^a \bmod p.$$

For $x, y \in G$, verification is done by executing the following protocol:

1. Alice chooses e_1, e_2 at random, $e_1, e_2 \in \mathbb{Z}_q{}^*$.
2. Alice computes $c = y^{e_1}\beta^{e_2} \bmod p$ and sends it to Bob.
3. Bob computes $d = c^{a^{-1} \bmod q} \bmod p$ and sends it to Alice.
4. Alice accepts y as a valid signature if and only if

$$d \equiv x^{e_1}\alpha^{e_2} \pmod{p}.$$

Undeniable Signature Scheme

Hash Functions

message	x	arbitrary length
	$\downarrow$	
message digest	$z = h(x)$	160 bits
	$\downarrow$	
signature	$y = sig_K(z)$	320 bits

Signing a message digest

Suppose p is a large prime and $q = (p-1)/2$ is also prime. Let α and β be two primitive elements of $\mathbb{Z}_p$. The value $\log_\alpha \beta$ is not public, and we assume that it is computationally infeasible to compute its value.
The hash function

$$h : \{0, \ldots, q-1\} \times \{0, \ldots, q-1\} \rightarrow \mathbb{Z}_p \backslash \{0\}$$

is defined as follows:

$$h(x_1, x_2) = \alpha^{x_1}\beta^{x_2} \bmod p.$$

Chaum-van Heijst-Pfitzmann Hash Function

1. $A = 67452301$ (hex)
   $B = efcdab89$ (hex)
   $C = 98badcfe$ (hex)
   $D = 10325476$ (hex)
2. **for** $i = 0$ **to** $N/16 - 1$ **do**
3. **for** $j = 0$ **to** 15 **do**
   $X[j] = M[16i + j]$
4. $AA = A$
   $BB = B$
   $CC = C$
   $DD = D$
5. **Round1**
6. **Round2**
7. **Round3**
8. $A = A + AA$
   $B = B + BB$
   $C = C + CC$
   $D = D + DD$

The MD4 Hash Function

1. $A = (A + f(B,C,D) + X[0]) \lll 3$
2. $D = (D + f(A,B,C) + X[1]) \lll 7$
3. $C = (C + f(D,A,B) + X[2]) \lll 11$
4. $B = (B + f(C,D,A) + X[3]) \lll 19$
5. $A = (A + f(B,C,D) + X[4]) \lll 3$
6. $D = (D + f(A,B,C) + X[5]) \lll 7$
7. $C = (C + f(D,A,B) + X[6]) \lll 11$
8. $B = (B + f(C,D,A) + X[7]) \lll 19$
9. $A = (A + f(B,C,D) + X[8]) \lll 3$
10. $D = (D + f(A,B,C) + X[9]) \lll 7$
11. $C = (C + f(D,A,B) + X[10]) \lll 11$
12. $B = (B + f(C,D,A) + X[11]) \lll 19$
13. $A = (A + f(B,C,D) + X[12]) \lll 3$
14. $D = (D + f(A,B,C) + X[13]) \lll 7$
15. $C = (C + f(D,A,B) + X[14]) \lll 11$
16. $B = (B + f(C,D,A) + X[15]) \lll 19$

Round 1

1. $A = (A + g(B,C,D) + X[0] + 5A827999) \lll 3$
2. $D = (D + g(A,B,C) + X[4] + 5A827999) \lll 5$
3. $C = (C + g(D,A,B) + X[8] + 5A827999) \lll 9$
4. $B = (B + g(C,D,A) + X[12] + 5A827999) \lll 13$
5. $A = (A + g(B,C,D) + X[1] + 5A827999) \lll 3$
6. $D = (D + g(A,B,C) + X[5] + 5A827999) \lll 5$
7. $C = (C + g(D,A,B) + X[9] + 5A827999) \lll 9$
8. $B = (B + g(C,D,A) + X[13] + 5A827999) \lll 13$
9. $A = (A + g(B,C,D) + X[2] + 5A827999) \lll 3$
10. $D = (D + g(A,B,C) + X[6] + 5A827999) \lll 5$
11. $C = (C + g(D,A,B) + X[10] + 5A827999) \lll 9$
12. $B = (B + g(C,D,A) + X[14] + 5A827999) \lll 13$
13. $A = (A + g(B,C,D) + X[3] + 5A827999) \lll 3$
14. $D = (D + g(A,B,C) + X[7] + 5A827999) \lll 5$
15. $C = (C + g(D,A,B) + X[11] + 5A827999) \lll 9$
16. $B = (B + g(C,D,A) + X[15] + 5A827999) \lll 13$

Round 2

1. $A = (A + h(B,C,D) + X[0] + 6ED9EBA1) \lll 3$
2. $D = (D + h(A,B,C) + X[8] + 6ED9EBA1) \lll 9$
3. $C = (C + h(D,A,B) + X[4] + 6ED9EBA1) \lll 11$
4. $B = (B + h(C,D,A) + X[12] + 6ED9EBA1) \lll 15$
5. $A = (A + h(B,C,D) + X[2] + 6ED9EBA1) \lll 3$
6. $D = (D + h(A,B,C) + X[10] + 6ED9EBA1) \lll 9$
7. $C = (C + h(D,A,B) + X[6] + 6ED9EBA1) \lll 11$
8. $B = (B + h(C,D,A) + X[14] + 6ED9EBA1) \lll 15$
9. $A = (A + h(B,C,D) + X[1] + 6ED9EBA1) \lll 3$
10. $D = (D + h(A,B,C) + X[9] + 6ED9EBA1) \lll 9$
11. $C = (C + h(D,A,B) + X[5] + 6ED9EBA1) \lll 11$
12. $B = (B + h(C,D,A) + X[13] + 6ED9EBA1) \lll 15$
13. $A = (A + h(B,C,D) + X[3] + 6ED9EBA1) \lll 3$
14. $D = (D + h(A,B,C) + X[11] + 6ED9EBA1) \lll 9$
15. $C = (C + h(D,A,B) + X[7] + 6ED9EBA1) \lll 11$
16. $B = (B + h(C,D,A) + X[15] + 6ED9EBA1) \lll 15$

Round 3

Time-stamping

1. Bob computes $z = h(x)$
2. Bob computes $z' = h(z \| pub)$
3. Bob computes $y = sig_K(z')$
4. Bob publishes (z, pub, y) in the next day's newspaper.

Key Pre-distribution

1. A prime p and a primitive element $\alpha \in \mathbb{Z}_p{}^*$ are made public.
2. V computes

$$K_{U,V} = \alpha^{a_U a_V} \bmod p = b_U{}^{a_V} \bmod p,$$

using the public value b_U from U's certificate, together with his own secret value a_V.

3. U computes

$$K_{U,V} = \alpha^{a_U a_V} \bmod p = b_V{}^{a_U} \bmod p,$$

using the public value b_V from V's certificate, together with her own secret value a_U.

Identification Schemes

1. Bob chooses a *challenge*, x, which is a random 64-bit string. Bob sends x to Alice.
2. Alice computes

$$y = e_K(x)$$

and sends it to Bob.

3. Bob computes

$$y' = e_K(x)$$

and verifies that $y' = y$.

Challenge-and-response protocol

Authentication Codes

An authentication code is a four-tuple ($\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{K}$, $\boldsymbol{E}$), where the following conditions are satisfied:

1. $\boldsymbol{S}$ is a finite set of possible source states
2. $\boldsymbol{A}$ is a finite set of possible authentication tags
3. $\boldsymbol{K}$, the keyspace, is a finite set of possible keys
4. For each $K \in \mathcal{K}$, there is an authentication rule e_K: $\boldsymbol{S} \rightarrow \boldsymbol{A}$.

Secret Sharing Schemes

Let t, w be positive integers, $t \le w$.
A (t, w)-threshold scheme is a method of sharing a key K among a set of w participants (denoted by $\boldsymbol{P}$), in such a way that any t participants can compute the value of K, but no group of $t-1$ participants can do so.

Initialization Phase

1. D chooses w distinct, non-zero elements of $\mathbb{Z}_p$, denoted x_i, $1 \le i \le w$ (this is where we require $p \ge w+1$). For $1 \le i \le w$, D gives the value x_i to P_i. The values x_i are public.

Share Distribution

2. Suppose D wants to share a key $K \in \mathbb{Z}_p$. D secretly chooses (independently at random) $t-1$ elements of $\mathbb{Z}_p$, $a_1, \ldots, a_{t-1}$.
3. For $1 \le i \le w$, D computes $y_i = a(x_i)$, where

$$a(x) = K + \sum_{j=1}^{t-1} a_j x^j \bmod p.$$

4. For $1 \le i \le w$, D gives the share y_i to P_i.

Shamir (t, w)-threshold scheme

Pseudo-random Number Generation

Let k, ℓ be positive integers such that $\ell \ge k+1$ (where ℓ is a specified polynomial function of k).
A (k, ℓ)-pseudo-random bit generator (more briefly, a (k, ℓ)-PRBG) is a function $f: (\mathrm{Z}_2)^k \to (\mathrm{Z}_2)^\ell$ that can be computed in polynomial time (as a function of k). The input $s_0 \in (\mathrm{Z}_2)^k$ is called the seed, and the output $f(s_0) \in (\mathrm{Z}_2)^\ell$ is called a pseudo-random bit-string.

Let $M \ge 2$ be an integer, and let $1 \le a, b \le M-1$. Define $k = \lceil \log_2 M \rceil$ and let $k+1 \le \ell \le M-1$.
For a seed s_0, where $0 \le s_0 \le M-1$, define

$$s_i = (as_{i-1} + b) \bmod M$$

for $1 \le i \le \ell$, and then define

$$f(s_0) = (z_1, z_2, \ldots, z_\ell),$$

where

$$z_i = s_i \bmod 2.$$

$1 \le i \le \ell$. Then f is a (k, ℓ)-Linear Congruential Generator.

Linear Congruential Generator

Zero-knowledge Proofs

- Completeness
  If x is a yes-instance of the decision problem, then Vic will always accept Peggy's proof.

- Soundness
  If x is a no-instance of, then the probability that Vic accepts the proof is very small.

Input: an integer n with unknown factorization $n = pq$, where p and q are prime, and $x \in QR(n)$

1. Repeat the following steps $\log_2 n$ times:
2. Peggy chooses a random $v \in Z_n^*$ and computes

$$y = v^2 \bmod n.$$

   Peggy sends y to Vic.
3. Vic chooses a random integer $i = 0$ or 1 and sends it to Peggy.

4. Peggy computes

$$z = u^i v \bmod n,$$

   where u is a square root of x, and sends z to Vic.
5. Vic checks to see if

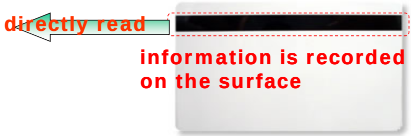
$$z^2 \equiv x^i y (\bmod n).$$

6. Vic accepts Peggy's proof if the computation of step 5 is verified in each of the $\log_2 n$ rounds.

A perfect zero-knowledge interactive proof system for Quadratic Residues

Magnetic stripe card vs Smart Card

- Magnetic stripe card： significant information can be read from the surface of the stripe



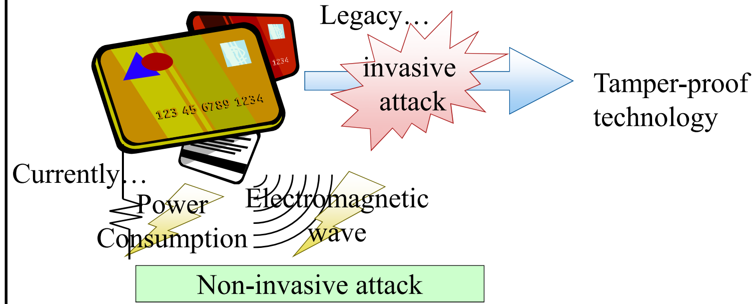
Easy to forge

- Smart card: significant information is stored in the IC chip

Hard to forge

Smartcard is high-security token
with encryption communication

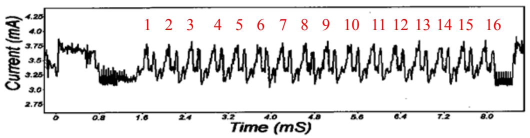
Attacks against smart card



Side-channel attack (SCA)

Protection against SCA is required

An example of the power consumption of smart card

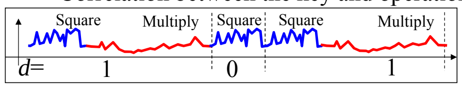


Power consumption of 16 rounds DES on a smartcard
"Paul Kocher, Joshua Jaffe, and Benjamin Jun "Differential Power Analysis", Advances in Cryptography-CRYPTO'99", pp.388-397.

"Power analysis" is a powerful attack against smart card

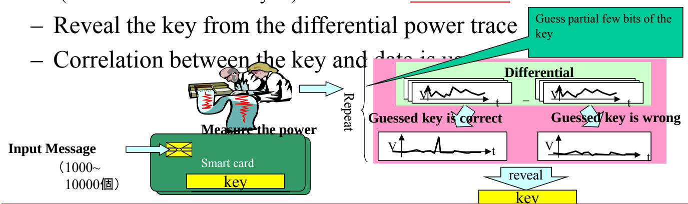
Power analysis

- SPA(Simple Power Analysis): Observe the internal operation processing
  - Reveal the key from single power trace
  - Correlation between the key and operation is used



input: cipher text c,
private key $d=d[n-1]||..||d[0]$
$T=1$
for $i=n-1$ down to 0
$T = T^2 \pmod N$ /* Square */
if $d[i]=1$ $T=T \times c \pmod N$ /* Multiply */
return $T= c^d \pmod N$

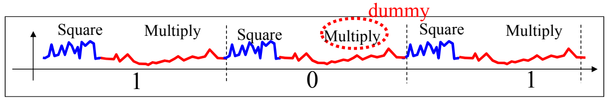
- DPA(Differential Power Analysis): Observe the internal data
  - Reveal the key from the differential power trace
  - Correlation between the key and



Protection must be secure against SPA and DPA in both

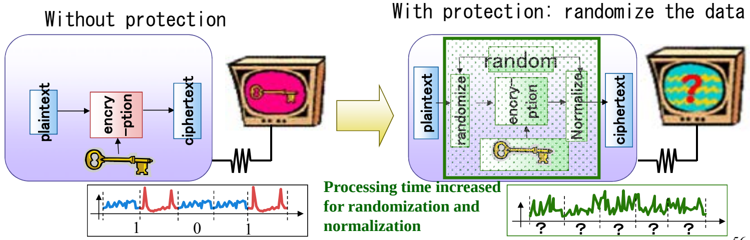
Protection against power analysis

- Protect SPA: Perform the constant operation pattern



Processing time increased +33% for dummy operation

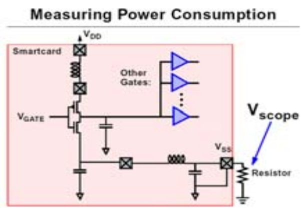
- Protect DPA: Randomize the internal data to hide the correlation



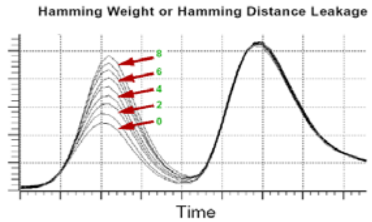
Problem: processing time overhead is increased with protection

Data hamming weight and power consumption

■Set up



■Result



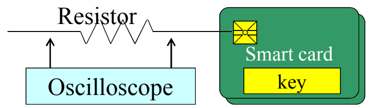
Power consumption grows in proportion with the hamming weight of the data (for certain IC chips)

From the paper of T.S.Messerges http://www.iccip.csl.uiuc.edu/conf/ceps/2000/messerges.pdf

Protection against DPA

- Reduce the signal
  - Represent the data without hamming weight difference e.g. 0→01, 1→1
  - Circuit size is increased
- Increase the noise
  - Add the noise generator circuit.
  - Protection is deactivated by increasing the number of the power consumption data
- Duplicate the data
  - Duplicate the intermediate data M into two random data M_1 and M_2 satisfying $M=M_1 \oplus M_2$
  - Processing time/circuit size is increased
- Update date the cryptographic key with certain period
  - If the key before is updated enough number of the power consumption data is collected, the attack is avoided.

Power analysis



- Reveal the cryptographic key stored in the smart card by observing the power consumption(Kocher, 1998)
- Power consumption shows internal operation and data value in the smart card, which are related with the key
- Simple and powerful attack
  - Just add a resistor to Vcc of IC chip
  - Instrument is low-cost (Digital oscilloscope)

This attack is possible even when the implemented cryptographic algorithm is mathematically secure
→Extra security protection mechanism must be implemented