CS491/591 Intro. to Computer Security

Fall 2012

Lecture on Cryptographic Primitives

Industrial-Strength Crypto

- Shared key (symmetric) methods
- Private key (asymmetric) methods
- (Hash functions)

Shared Key Cryptography

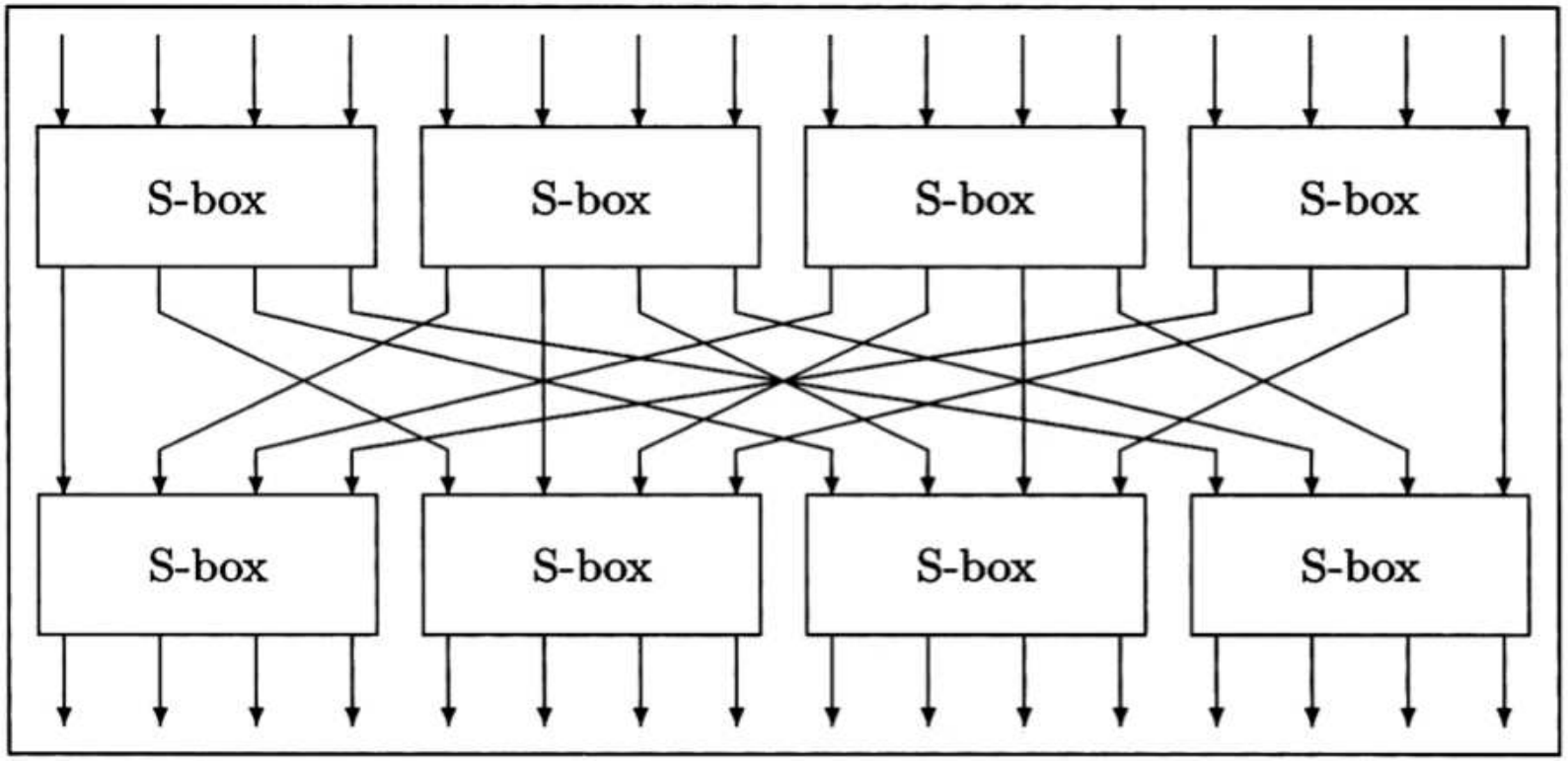
Sender and receiver share a key.

- Secret between the two of them.
- n participants require n^2 keys.
- How do they get the keys?

Typically these are **block ciphers**

- Encrypt blocks of plaintext into blocks of ciphertext
- Use combination of substutition and transposition in form of SP-network.
- Networks needs to be wide enough, have enough rounds, and have good S-box internal design.

SP-network Architecture



(from Ross Anderson, Security Engineering)

DES (Data Encryption Standard)

Standardized in 1976.

Security analyzed by NSA.

- Algorithm is public, but full design **rationale** is not.
- There were persistent worries about “back doors.”

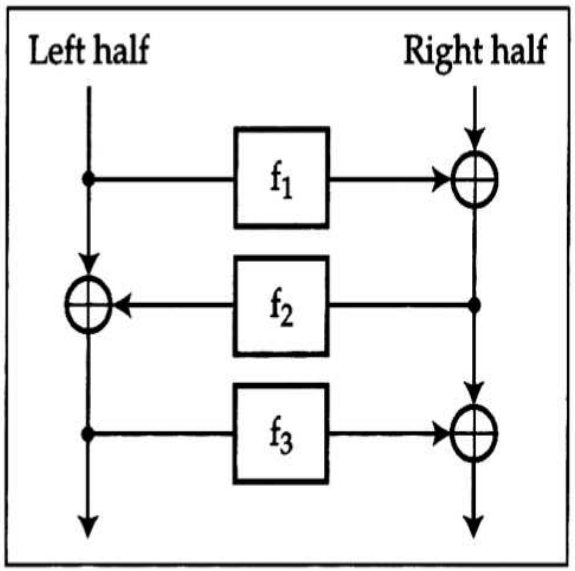
Widely used in banking, government, and embedded applications.

Block size = 64 bits
Key length = 56 bits
Structured as 16-round **Feistel cipher**

Uses simple operators on (up to) 64 bit values.

Relatively simple and fast to implement in software and hardware.

Feistel Structure



(from Ross Anderson, Security Engineering)

FEISTEL

Important feature of Feistel: decrypt just by applying functions f_j in reverse order.

Proof: Have

$$L_j = R_{j-1}$$
$$R_j = L_{j-1} \oplus f_j(R_{j-1})$$

Thus

$$R_{j-1} = L_j$$
$$L_{j-1} \oplus f_j(R_{j-1}) = R_j$$
$$L_{j-1} \oplus f_j(R_{j-1}) \oplus f_j(R_{j-1}) = R_j \oplus f_j(R_{j-1})$$
$$L_{j-1} = R_j \oplus f_j(R_{j-1})$$
$$L_{j-1} = R_j \oplus f_j(L_j)$$

Consequently the f_j don't need to be invertible; makes design easier.

What is in DES Feistel round?

- expansion, addition of key material, processing with ”S boxes”, permutation
- box is a 6-to-4 substitution table
- design is black art

The kinds of characteristics we want:

- No output should be close to a linear funcion of the inputs.
- If inputs differ in exactly one bit, output must differ in at least 2 bits.
- Etc.

In fact, DES S-boxes appear to work very well, but we don’t really know why!

But DES is not very secure because key length is so short.

- By 1999, a distributed attack could break it in 22 hours.

Still used today, but typically using a triple repitition to get larger effective key length (just a double repetition doesn’t help).

AES (Advanced Encryption Standard)

Result of public NIST competition.

Uses Rijndael algorithm, invented by a pair of Dutch cryptographers.

Adopted in 2002.

Used broadly in industry, ecommerce, government...

Blocksize = 128 bits
Keylength = 128, 192 or 256 bits
Rounds = 10, 12, or 14

Fast on stock hardware, and newer X86 processors include custom instructions.

What is in a Rijndael round?

Each round consists of four steps applied to 4x4 square of bytes:

- xor in a key (specific to each round, derived from user key by a formula)
- apply a substitution using an S-box to each byte
- shift each row by a different amount
- mix contents of columns using a combination of shifts and xors.

Note: all these steps (and sbox) are invertible.

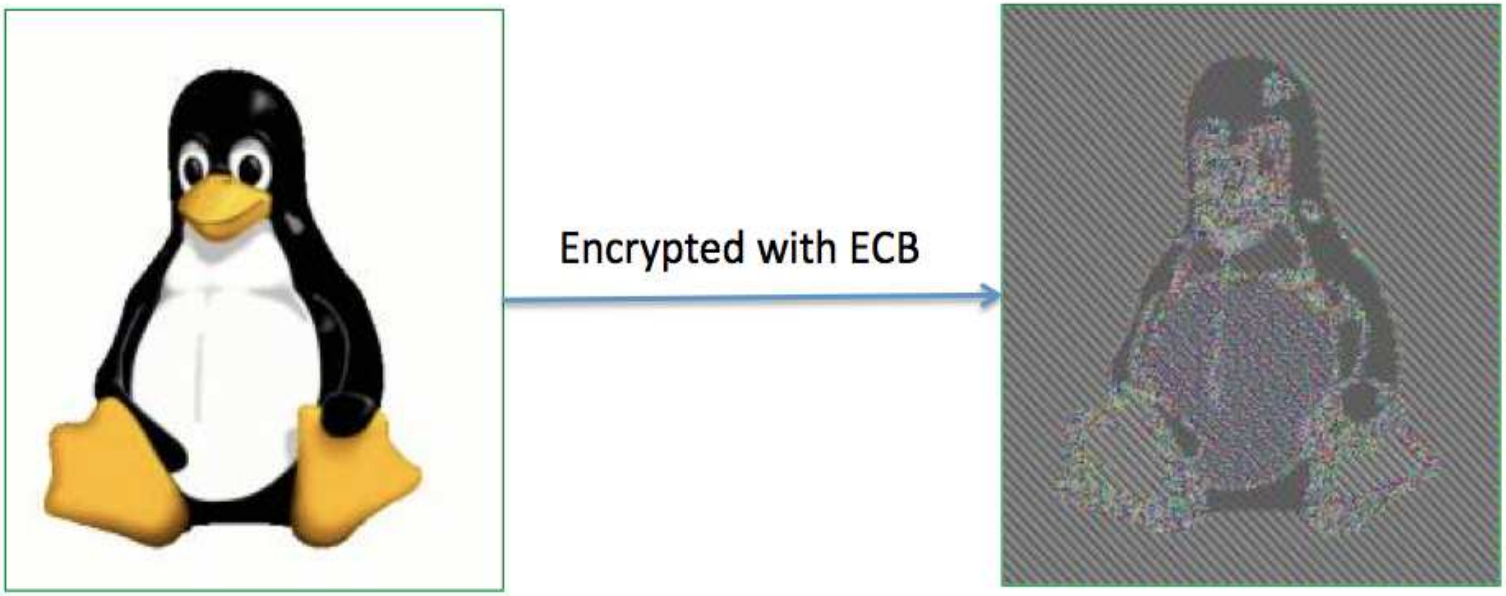
Best short-cut attack on 128-bit version that recovers key (chosen ciphertext) is $O(2^{126})$, almost as slow as brute force $O(2^{128})$.

Happily, Rijndael can be extended to larger key sizes.

Electronic Code Book

Important issue with block ciphers: what to do on longer messages?

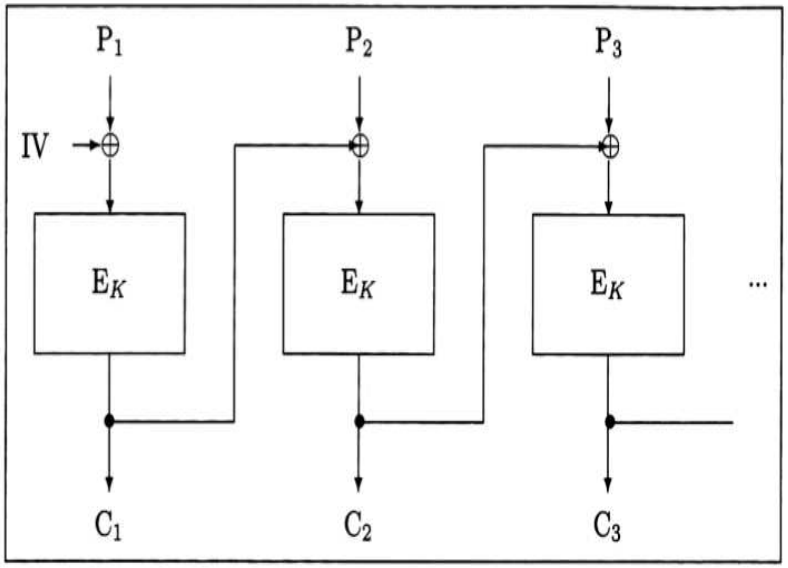
- ECB is simplest approach: each n-bit block is independently using the same key
- not very secure: just a substitution cipher with alphabet of n-bit strings



Images courtesy of
http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

Cipher Block Chaining

- Each block of plaintext is xor'ed with preceding block of ciphertext before encryption:



(from Ross Anderson, Security Engineering)

- secure against passive (no injection) attacks
- not so secure against chosen plaintext attacks

PUBLIC KEY CRYPTOGRAPHY

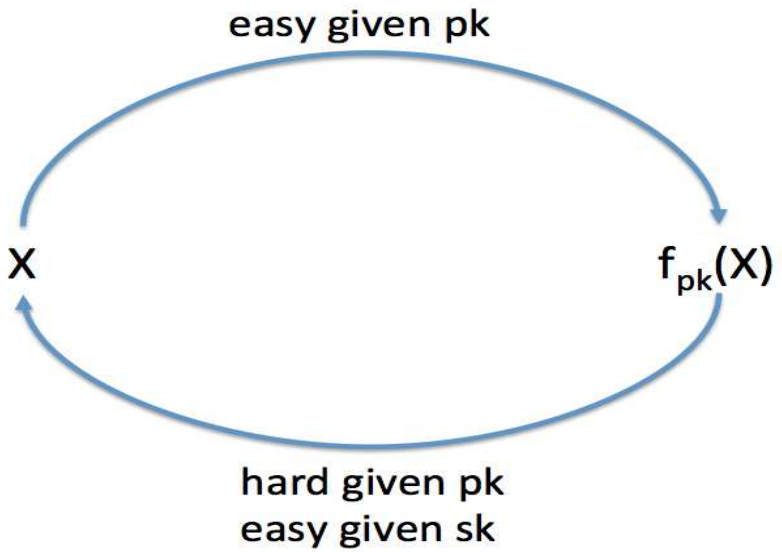
Uses two keys, one public and one secret.

$D(sk, E(pk, M)) = M$

Receiver publishes their public key.

n participants need only n (pairs of) keys

Security is based on notion of **trapdoor** function:



(from Tom Ristenpart)

Existence of these functions is based on mathematical computation problems that are believed to be hard.

RSA (RIVEST/SHAMIR/ADLEMAN) ALGORITHM

Proposed in 1979.

- Designers won 2002 Turing Award
- Very similar work was developed by Britain's GCHQ ca. 1970 — but never published (or used).

Widely used in protocols, especially to establish shared keys for cheaper protocols.

1000X slower than DES and harder to implement

Basic Idea:

- Treat plaintext as a large binary number.
- Encrypt by modular exponentiation (repeated multiplication modulo a number).
- Security is based on assumed difficulty of factoring large numbers. Largest number known to have been factored by general-purpose algorithm is 768 bits. RSA typically uses 1024-2048 bits.

Number Theory Preliminaries

Recall definition of mod operator:
$a \ mod \ b$ = remainder ("residue") when a is divided by b

Have:

- $(a + b) \ mod \ n = ((a \ mod \ n) + (b \ mod \ n)) mod \ n$
- $ab \ mod \ n = ((a \ mod \ n)(b \ mod \ n)) \ mod \ n$

Define "congruence $mod \ n$" as follows:

- $a \equiv b (mod \ n)$ iff $a = b + kn$ for some integer k

Will make use of the idea of multiplicative inverse modulo n:
a^{-1} is the number in $\{0, \ldots, n-1\}$ such that $a * a^{-1} \equiv 1 (mod \ n)$.

The multiplicative inverse exists whenever a and n are relatively prime (have no common factors other than 1). Examples:

- $2^{-1} = 3 (mod \ 5)$ because $2 * 3 \equiv 1 (mod \ 5)$.
- $4^{-1} = 4 (mod \ 5)$ because $4 * 4 \equiv 1 (mod \ 5)$.
- $3^{-1} = 7 (mod \ 10)$ because $3 * 7 \equiv 1 (mod \ 10)$.

RSA KEY GENERATION

- Choose large distinct primes p and q
- Set $n = p * q$
- Choose a random encryption exponent e such that e and $(p-1) * (q-1)$ are relatively prime
- Derive decryption exponent d as $d = e^{-1} (mod\ ((p-1) * (q-1)))$. (This exists because e and $(p-1) * (q-1)$ are relatively prime).
- Public key $pk = (e, n)$
- Secret key $sk = (d, n)$
- Now **forget** p and q; we don't need them anymore and we must not accidentally disclose them!

RSA Encryption and Decryption

Given message $m < n$

- (break up m into parts smaller than n if necessary)

$E((e, n), m) = m^e \ mod \ n$

$D((d, n), c) = c^d \ mod \ n$

Note that these are not cheap operations (even with some useful shortcuts).

EXAMPLE RSA

Choose $p = 47$,$q = 71$

- $n = p * q = 3337$
- $(p - 1) * (q - 1) = 3220$
- Choose e relatively prime with 3220: e.g. $e = 79$
- Public key is (79, 3337)
- Find $d = 79^{-1}\ mod\ 3220 = 1019$
- Private key is (1019, 3337)

To encrypt m = 688232687966683

- Break into chunks $<$ 3337 : 688 232 687 966 683
- Encrypt: E((79, 3337), 688) = $688^{79}\ mod\ 3337$ = 1570
- Decrypt: D((1019, 3337), 1570) = $1570^{1019}\ mod\ 3337$ = 688

A LITTLE NUMBER THEORY

A (fairly) elementary fact in number theory (Euler's generalization of Fermat's "little" theorem') says that:

If m and n are relatively prime, then $m^{\phi(n)} \equiv 1(mod\ n)$.

• where "Euler's totient function" $\phi(n)$ = number of positive integers $\leq n$ that are relatively prime to n.

[Proof omitted.]

Corollary: Suppose m and n are relatively prime and $ed \equiv 1(mod\ \phi(n))$ for some $e, d \geq 1$. Then we can write $ed = 1 + h\phi(n)$ for some nonnegative integer h. So:

$$m^{ed} \equiv m^{1+h\phi(n)} \equiv m(m^{\phi}(n))^{h} \equiv m(mod\ n)$$

WHY DOES RSA WORK?

In the RSA setting, we have $n = pq$ for p, q prime. It is then easy to see [proof omitted] that $\phi(n) = ((p-1) * (q-1))$, and so $ed \equiv 1(mod\ \phi(n))$ (by definition of d).

Hence the corollary from the previous page applies:

$$m^{ed} \equiv m^{1+h\phi(n)} \equiv m(m^{\phi}(n))^h \equiv m(mod\ n)$$

So now suppose $c = m^e\ mod\ n$. Then:

$$\begin{aligned} & c^d\ mod\ n & \\ = & (m^e\ mod\ n)^d\ mod\ n & (\text{defn}) \\ = & m^{ed}\ mod\ n & (\text{arith}) \\ = & m\ mod\ n & (\text{by corollary}^*) \\ = & m & (\text{because } m < n) \end{aligned}$$

* a slightly different argument is needed if m and n are not relatively prime (though this is highly unlikely in practice!)

OTHER PUBLIC KEY SYSTEMS

Other results in number theory provide similar sources of (presumably) hard problems.

Discrete Logarithms

For prime p, the set $\mathbb{Z}_p = \{0, 1, ...p-1\}$ with operations $mod\ p$ form a **finite field** with usual laws of arithmetic.

A "**primitive root** g modulo a prime p" is a number in $\mathbb{Z}_p$ such that $\{g^0\ mod\ p, g^1\ mod\ p, \ldots, g^{p-1}\ mod\ p\} = \mathbb{Z}_p$. (We say g "generates" the group $\mathbb{Z}_p$.)

For any primitive root g of a large prime p, the function

$$f(x) = g^x\ mod\ p$$

is effectively a "one-way" function — that is, there is no computationally efficient way to invert it (as far as we know).

DIFFIE-HELLMAN KEY EXCHANGE

Protocol for obtaining a shared secret using public communication.

- Security based on difficulty of discrete logarithm problem.
- Used as basis for ElGamel public key protocol.

Method:

- Alice and Bob publicly agree on a large prime p and primitive root g.
- Alice chooses a secret value A; Bob chooses a secret value B.
- Alice send Bob $g^A \ mod \ p$ and Bob sends Alice $g^B \ mod \ p$.
- The shared secret is $g^{AB} \ mod \ p$.
- Alice and Bob can each easily calculate this by exponentiating (mod p) the value they receive by their secret value.
- Eve the evil eavesdropper cannot do this easily (as far as we know). In particular, Eve cannot extract A from $g^A \ mod \ p$ without solving the discrete logarithm problem.