Module 2: Cryptography II

Nitesh Saxena

Overview of the Module

1.1 Block Cipher Modes of Encryption
1.2 Other Ciphers
1.3 Public Key Crypto Overview
1.4 Math Background
1.5 Public Key Encryption (RSA)
1.6 RSA Security
1.7 Digital Signatures

Module 2, Lecture 1

Block Cipher Encryption Modes

Block Cipher Encryption modes

- Electronic Code Book (ECB)
- Cipher Block Chain (CBC)
  - Most popular one
- Others (we will not cover)
  - Cipher Feed Back (CFB)
  - Output Feed Back (OFB)

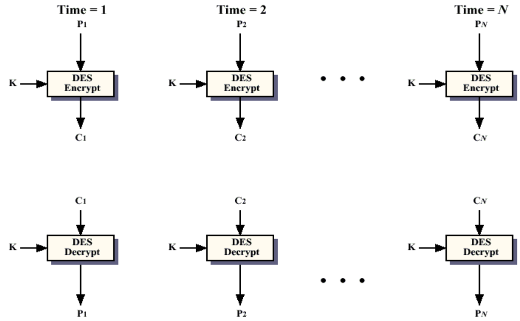
Analysis

We will analyze both mode in terms of:

- Security
- Computational Efficiency (parallelizing encryption/decryption)
- Transmission Errors

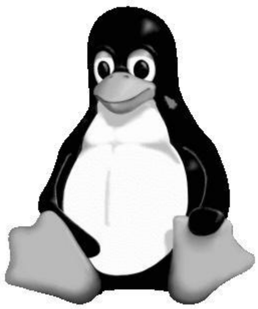
Electronic Code Book (ECB) Mode

- Although DES encrypts 64 bits (a block) at a time, it can encrypt a long message (file) in Electronic Code Book (ECB) mode.

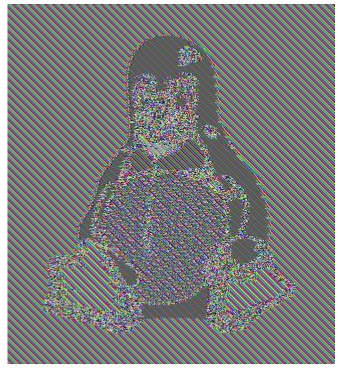


- Deterministic -- If same key is used then identical plaintext blocks map to identical ciphertext

Example – why ECB is bad?



Tux

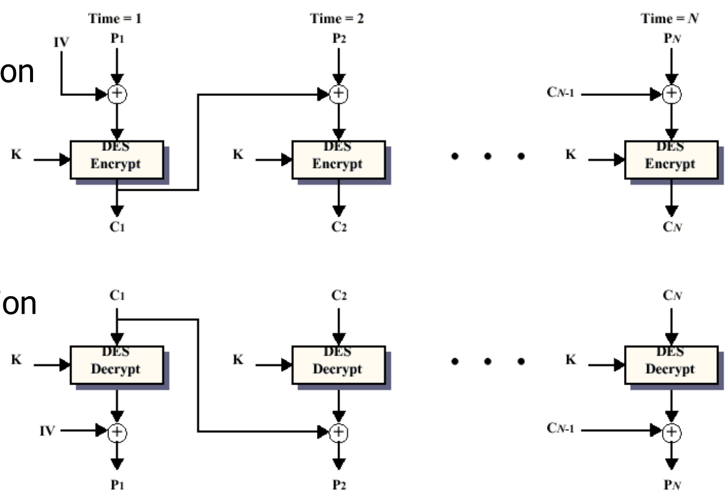


Tux encrypted with AES in ECB mode

Cipher Block Chain (CBC) Mode

encryption

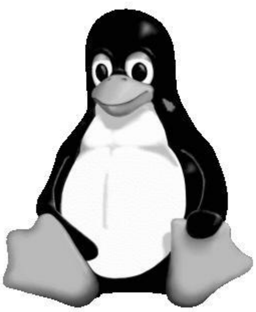
decryption



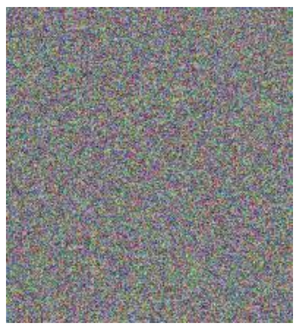
CBC Traits

- Randomized encryption
- IV – Initialization vector serves as the randomness for first block computation; the ciphertext of the previous block serves as the randomness for the current block computation
- IV is a random value
- IV is **no secret**; it is sent along with the ciphertext blocks (it is part of the ciphertext)

Example – why CBC is good?



Tux



Tux encrypted with AES in CBC mode

CBC – More Properties

- What happens if *k-th* cipher block C_K gets corrupted in transmission.
  - With ECB – Only decrypted P_K is affected.
  - With CBC?
    - Only blocks P_K and P_{K+1} are affected!!

Security of Block Cipher Modes

- ECB is not even secure against eavesdroppers (ciphertext only and known plaintext attacks)
- CBC is secure against CPA attacks (assuming 3-DES or AES is used in each block computation); automatically secure against eavesdropping attacks
- However, **not secure against CCA**. Why?
  - Intuitively, this is because the ciphertext can be "massaged" in a meaningful way

How to achieve CCA security?

- Prevent any massaging of the ciphertext
- Intuitively, this can be achieved by using integrity protection mechanisms (such as MACs – message authentication codes), which we will study later
- The ciphertext is generated using CBC and a MAC is generated on this ciphertext
- Both ciphertext and the MAC is sent off
- The other party decrypts only if MAC is valid

Module 2, Lecture 2

Other Ciphers

Advanced Encryption Standard (AES)

- National Institute of Science and Technology
  - DES is an aging standard that no longer addresses today's needs for strong encryption
  - Triple-DES: Endorsed by NIST as today's defacto standard
- AES: The Advanced Encryption Standard
  - Finalized in 2001
  - Goal – To define Federal Information Processing Standard (FIPS) by selecting a new powerful encryption algorithm suitable for encrypting government documents
  - AES candidate algorithms were required to be:
    - Symmetric-key, supporting 128, 192, and 256 bit keys
    - Royalty-Free
    - Unclassified (i.e. public domain)
    - Available for worldwide export

AES

- AES Round-3 Finalist Algorithms:
  - MARS
    - Candidate offering from IBM
  - RC6
    - Developed by Ron Rivest of RSA Labs, creator of the widely used RC4 algorithm
  - Twofish
    - From Counterpane Internet Security, Inc.
  - Serpent
    - Designed by Ross Anderson, Eli Biham and Lars Knudsen
  - Rijndael: the winner!
    - Designed by Joan Daemen and Vincent Rijmen

Other Symmetric Ciphers and their applications

- IDEA (used in PGP)
- Blowfish (password hashing in OpenBSD)
- RC4 (used in WEP), RC5
- SAFER (used in Bluetooth)

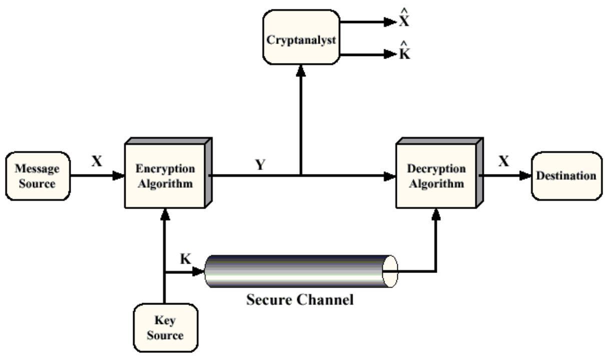
Module 2, Lecture 3

Public Key Crypto Overview

Recall: Private Key/Public Key Cryptography

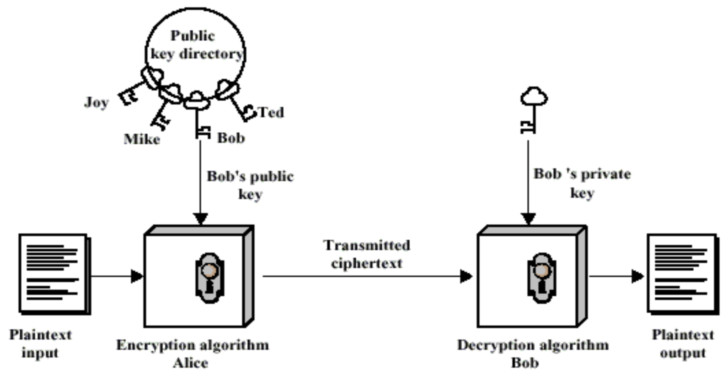
- **Private Key**: Sender and receiver share a common (private) key
  - Encryption and Decryption is done using the private key
  - Also called conventional/shared-key/single-key/symmetric-key cryptography
- **Public Key**: Every user has a private key and a public key
  - Encryption is done using the public key and Decryption using private key
  - Also called two-key/asymmetric-key cryptography

Private key cryptography revisited.



- Good: Quite efficient
- Bad: Key distribution and management is a serious problem

Public key cryptography model



- Good: Key management problem potentially simpler
- Bad: Much slower than private key crypto (we'll see later!)

Public Key Encryption

- Two keys:
  - public encryption key *e*
  - private decryption key *d*
- Encryption easy when *e* is known
- Decryption easy when *d* is known
- Decryption hard when *d* is not known
- We'll study such public key encryption schemes; first we need some mathematical background (next lecture).

Public Key Encryption: Security Notions

- Very similar to what we studied for private key encryption
  - What's the difference?
    - Adversary has access to public key
    - Adversary can create encryptions on its own

Module 2, Lecture 4

Math Background

Group: Definition

(G,.) (where G is a set and . : GxG→G) is said to be a group if following properties are satisfied:

1. *Closure* : for any a, b ε G, a.b ε G
2. *Associativity :* for any a, b, c ε G, a.(b.c)=(a.b).c
3. *Identity* : there is an identity element such that a.e = e.a = a, for any a ε G
4. *Inverse* : there exists an element a^{-1} for every a in G, such that $a.a^{-1} = a^{-1}.a = e$

Groups: Examples

- Set of all integers with respect to addition -- (Z,+)
- Set of all integers with respect to multiplication (Z,*) – not a group
- Set of all real numbers with respect to multiplication (R,*)
- Set of all integers modulo m with respect to modulo addition (Z_m, “modular addition”)

Multiplicative inverses in Z_m

- 1 is the multiplicative identity in Z_m

$$x * 1 \equiv x (\bmod m) \equiv 1 * x (\bmod m)$$

- Multiplicative inverse ($x*x^{-1}=1 \bmod m$)
  - SOME, but not ALL elements have unique multiplicative inverse.
  - In Z_9 : 3*0=0, 3*1=3, 3*2=6, 3*3=0, 3*4=3, 3*5=6, …, so 3 does not have a multiplicative inverse (mod 9)
  - On the other hand, 4*2=8, 4*3=3, 4*4=7, 4*5=2, 4*6=6, 4*7=1, so 4^{-1}=7, (mod 9)

Which numbers have inverses?

- In Z_m, x has a multiplicative inverse if and only if x and m are relatively prime or gcd(x,m)=1
  - E.g., 4 in Z_9
- Efficient algorithm to compute inverses
  - Extended Euclidian Algorithm

Modular Exponentiation

- Usual approach to computing x^c mod n is inefficient when c is large.
- Efficient algorithm: Square and Multiply

Euler's totient function

- Given positive integer *n*, Euler's totient function $\Phi(n)$ is the number of positive numbers less than n that are relatively prime to n
- Fact: If *p* is prime then
  - {1,2,3,…,*p*-1} are relatively prime to *p*.

Euler's totient function

- Fact: If *p* and *q* are prime and *n=pq* then

$$\Phi(n) = (p-1)(q-1)$$

- Each number that is not divisible by *p* or by *q* is relatively prime to *pq*.
  - E.g. p = 5, q = 7: {1,2,3,4,-,6,-,8,9,-,11,12,13,-,-,16,17,18,19,-,-,22,23,24,-,26,27,-,29,-,31,32,33,34,-}
  - pq – p - (q-1) = (p-1)*(q-1)

Euler's Theorem and Fermat's Theorem

- If *a* is relatively prime to n then

$$a^{\Phi(n)} \equiv 1 \bmod n$$

- If *a* is relatively prime to p then

a^{p-1} = 1 mod p

Euler's Theorem and Fermat's Theorem

EG: Compute 9^{100} **mod** 17:

p =17, so p-1 = 16. 100 = 6·16+4. Therefore, $9^{100}=9^{6\cdot 16+4}=(9^{16})^6(9)^4$. So mod 17 we have 9^{100} $\equiv (9^{16})^6(9)^4 \pmod{17} \equiv (1)^6(9)^4 \pmod{17}$
$\equiv (81)^2 \pmod{17} \equiv \mathbf{16}$

Further Reading

- Chapter 4 of Stallings
- Chapter 2.4 of HAC

Module 2, Lecture 5

The RSA Cryptosystem (Encryption)

RSA Math Setting

- In RSA, we work in a multiplicative group Z_n*, i.e., a group of all numbers between 0 and n-1 that are relatively prime to n.
- Here, n is a product of two prime numbers p and q
  - i.e., n itself is composite
- The size of Z_n* is $\Phi(n)$ = (p-1)(q-1)
- Computation is done modulo n

"Textbook" RSA: KeyGen

- Alice wants people to be able to send her encrypted messages.
- She chooses two (large) prime numbers, *p* and *q* and computes *n=pq* and $\Phi(n)$. ["large" = 1024 bits +]
- She chooses a number *e* such that *e* is relatively prime to $\Phi(n)$ and computes *d*, the inverse of *e* in $Z_{\Phi(n)}$, i.e., ed =1 mod $\Phi(n)$
- She publicizes the pair *(e,n)* as her public key. (e is called RSA exponent, n is called RSA modulus). She keeps *d* secret and destroys *p*, *q*, and $\Phi(n)$
- Plaintext and ciphertext messages are elements of Z_n and *e* is the encryption key.

RSA: Encryption

- Bob wants to send a message *x* (an element of Z_n^*) to Alice.
- He looks up her encryption key, *(e,n)*, in a directory.
- The encrypted message is
$$y = E(x) = x^e \bmod n$$
- Bob sends *y* to Alice.

RSA: Decryption

- To decrypt the message

$$y = E(x) = x^e \bmod n$$

she's received from Bob, Alice computes

$$D(y) = y^d \bmod n$$

Claim: *D(y) = x*

RSA: why does it all work

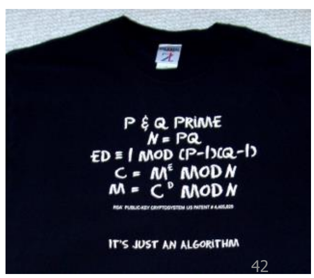
- Need to show
  - D[E[x]] = x
  - E[x] and D[y] can be computed efficiently if keys are known
  - E^{-1}[y] cannot be computed efficiently without knowledge of the (private) decryption key d.
- Also, it should be possible to select keys reasonably efficiently
  - This does not have to be done too often, so efficiency requirements are less stringent.

E and D are Inverses

$$
\begin{aligned}
D(y) &= y^d \bmod n \\
&\equiv (x^e \bmod n)^d \bmod n \\
&\equiv (x^e)^d \bmod n \\
&\equiv x^{ed} \bmod n \\
&\equiv x^{t\Phi(n)+1} \bmod n \qquad \text{Because } ed \equiv 1 \bmod \Phi(n) \\
&\equiv (x^{\Phi(n)})^t x \bmod n \\
&\equiv 1^t x \bmod n \equiv x \bmod n \qquad \text{From Euler's Theorem}
\end{aligned}
$$

Tiny RSA example.

- Let p = 7, q = 11. Then n = 77 and
  $\Phi(n) = 60$
- Choose e = 13. Then d = 13^{-1} mod 60 = 37.
- Let message = 2.
- E(2) = 2^{13} mod 77 = 30.
- D(30) = 30^{37} mod 77=2



Slightly Larger RSA example.

- Let p = 47, q = 71. Then n = 3337 and
  $$\Phi(pq) = 46 * 70 = 3220$$
- Choose e = 79. Then d = 79^{-1} mod 3220 = 1019.
- Let message = 688232… Break it into 3 digit blocks to encrypt.
- E(688) = 688^{79} mod 3337 = 1570.
  E(232) = 232^{79} mod 3337 = 2756
- D(1570) = 1570^{1019} mod 3337 = 688.
  D(2756) = 2756^{1019} mod 3337 = 232.

Module 2, Lecture 6

RSA Security

Security of RSA: RSA assumption

- Suppose Eve intercepts the encrypted message *y* that Bob has sent to Alice.
- Eve can look up *(e,n)* in the public directory (just as Bob did when he encrypted the message)
- If Eve can compute $d = e^{-1} \bmod \Phi(n)$ then he can use $D(y) = y^d \bmod n = x$ to recover the plaintext *x*.
- If Oscar can compute $\Phi(n)$, he can compute *d* (the same way Alice did).

Security of RSA: factoring

- Oscar knows that *n* is the product of two primes
- If he can factor *n*, he can compute $\Phi(n)$
- But factoring large numbers is *very difficult*:
  - Grade school method takes $O(\sqrt{n})$ divisions.
  - Prohibitive for large n, such as 160 bits
  - Better factorization algorithms exist, but they are still too slow for large n
  - Lower bound for factorization is an open problem

How big should n be?

- Today we need n to be at least 1024-bits
  - This is equivalent to security provided by 80-bit long keys in private-key crypto
- No other attack on RSA function known
  - Except some side channel attacks, based on timing, power analysis, etc. But, these exploit certain physical charactesistics, not a theoretical weakness in the cryptosystem!

Efficiency (even with large n)

During key generation:

- Select large primes
  - Primes are dense so choose randomly.
  - Probabilistic primality testing methods known. Work in logarithmic time.
- Compute multiplicative inverses
  - Efficient algorithm (Extended Euclidean algorithm) exists

During encryption and decryption

- Requires modular multiplication (use Square and Multiply)

RSA in Practice

- Textbook RSA is insecure
  - Since it is deterministic
- In practice, we use a “randomized” version of RSA, called RSA-OAEP
  - Use PKCS#1 standard for RSA encryption

  https://www.rfc-editor.org/rfc/rfc8017

  Interested in details of OAEP: refer to:

  https://iacr.org/archive/crypto2001/21390259.pdf

Further Reading

- Stallings Chapter 11
- HAC Chapter 9

Module 2, Lecture 7

Digital Signatures

Goals

- Authentication
- Integrity
- Non-repudiation

Public Key Signatures

- Signer has public key, private key pair
- Signer signs using its private key
- Verifier verifies using public key of the signer

Security Notion/Model for Signatures

- Existential Forgery under (adaptively) chosen message attack (CMA)
  - Adversary (adaptively) chooses messages m_i of its choice
  - Obtains the signature s_i on each m_i
  - Outputs any message m (≠ mi) and a signature s on m

RSA Signatures

- Key Generation: same as in encryption
- Sign(m): $s = m^d \bmod N$
- Verify(m,s): ($s^e == m \bmod N$)

- The above text-book version is insecure; why?
- In practice, we use a randomized version of RSA (implemented in PKCS#1)
  - Hash the message and then sign the hash