Cryptography

ITS335: IT Security

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 20 December 2015

its335y15s2l02, Steve/Courses/2015/s2/its335/lectures/crypto.tex, r4287

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

Key Management

Digital Signatures

Random Numbers

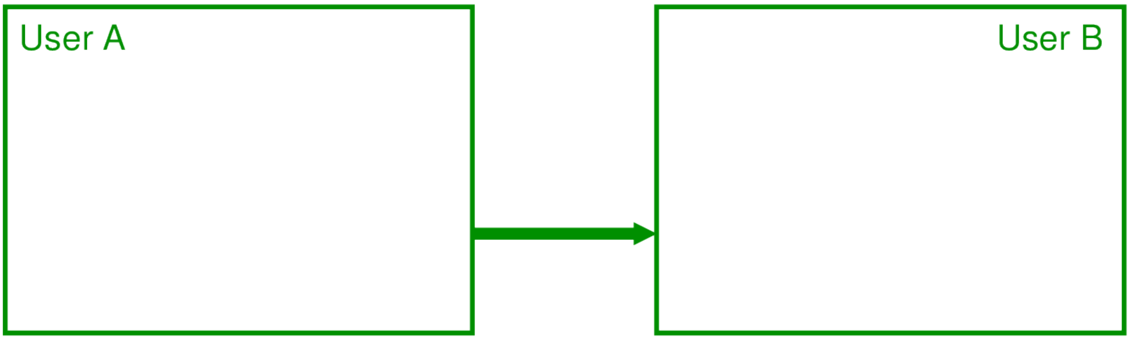
Summary

Encryption for Confidentiality

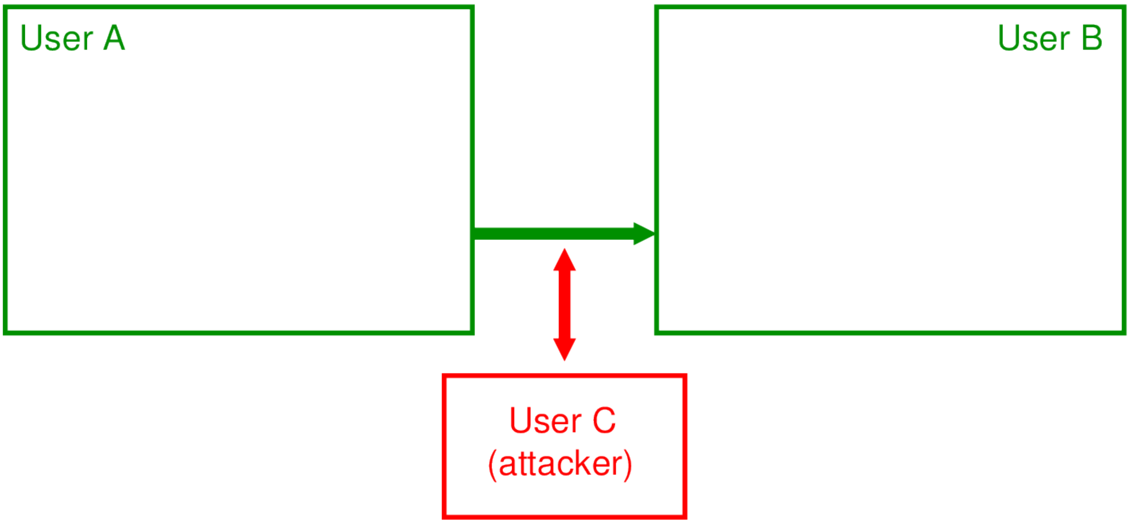
- Aim: assure confidential information not made available to unauthorised individuals (data confidentiality)
- How: encrypt the original data; anyone can see the encrypted data, but only authorised individuals can decrypt to see the original data
- Used for both sending data across network and storing data on a computer system

---

Model of Encryption for Confidentiality



Model of Encryption for Confidentiality

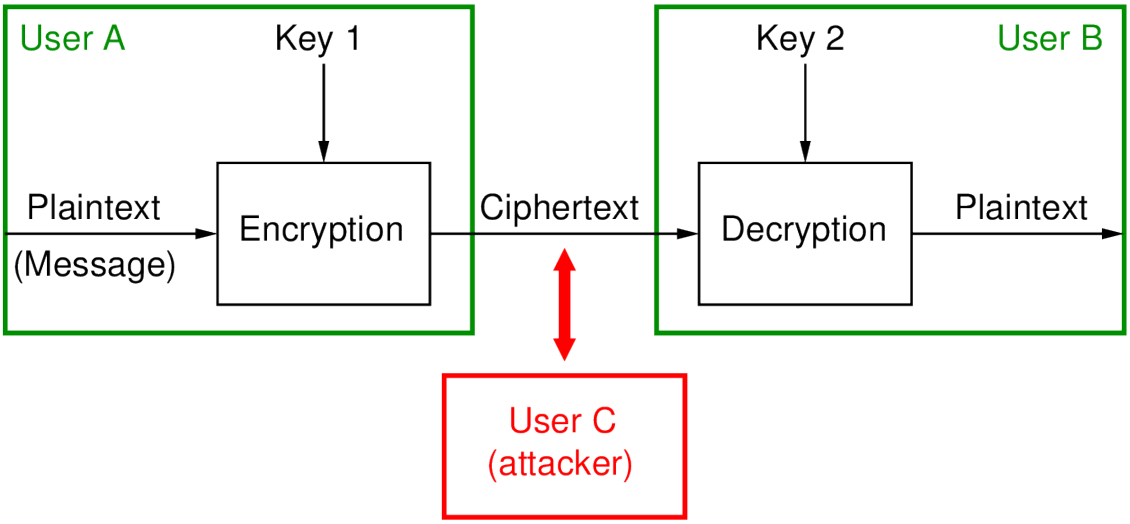


User A

User B

User C
(attacker)

Model of Encryption for Confidentiality



User A

Key 1

Plaintext
(Message)

Encryption

Ciphertext

Key 2

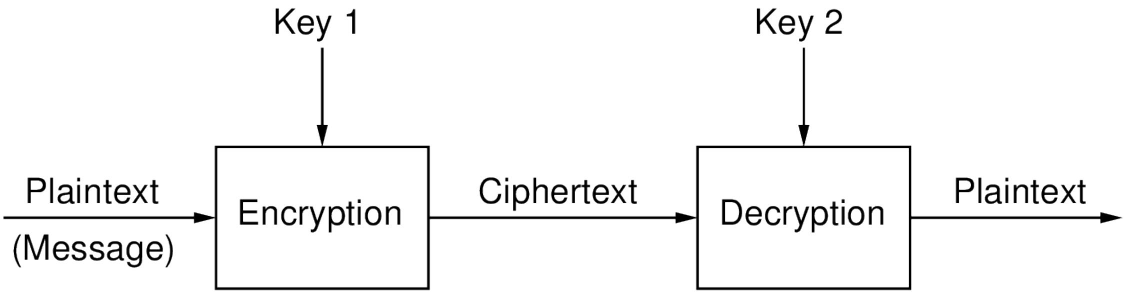
User B

Decryption

Plaintext

User C
(attacker)

Model of Encryption for Confidentiality



Key 1

Key 2

Plaintext
(Message)

Encryption

Ciphertext

Decryption

Plaintext

Terminology

Plaintext	original message
Ciphertext	encrypted or coded message
Encryption	convert from plaintext to ciphertext (enciphering)
Decryption	restore the plaintext from ciphertext (deciphering)
Key	information used in cipher known only to sender/receiver
Cipher	a particular algorithm (cryptographic system)
Cryptography	study of algorithms used for encryption
Cryptanalysis	study of techniques for decryption without knowledge of plaintext
Cryptology	areas of cryptography and cryptanalysis

Requirements and Assumptions

Requirements for secure use of symmetric encryption:

1. Strong encryption algorithm: Given the algorithm and ciphertext, an attacker cannot obtain key or plaintext
2. Sender/receiver know secret key (and keep it secret)

Assumptions:

- Cipher is known
- Secure channel to distribute keys

Characterising Cryptographic Systems

Operations used for encryption:

Substitution replace one element in plaintext with another

Transposition re-arrange elements

Product systems multiple stages of substitutions and transpositions

Number of keys used:

Symmetric sender/receiver use same key (single-key, secret-key, shared-key, conventional)

Public-key sender/receiver use different keys (asymmetric)

Processing of plaintext:

Block cipher process one block of elements at a time

Stream cipher process input elements continuously

Example Substitution Cipher: Caesar Cipher

Encrypt Shift plaintext letters K positions to right (wrapping where necessary)

Example Transposition Cipher: Rail-Fence

Encrypt Plaintext letters written in diagonals over K rows; ciphertext obtained by reading row-by-row

Example Product System

Encrypt Repeat following steps n times:

1. Apply Vigenere cipher with $K_{n,1}$
2. Apply Rail-fence cipher with $K_{n,2}$

Attacks

Goal of the Attacker

- Discover the plaintext (good)
- Discover the key (better)

Assumed Attacker Knowledge

- Ciphertext
- Algorithm
- Other pairs of (plaintext, ciphertext) using same key

Attack Methods

Brute-force attack Try every possible key on ciphertext

Cryptanalysis Exploit characteristics of algorithm to deduce plaintext or key

Assumption: attacker can recognise correct plaintext

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

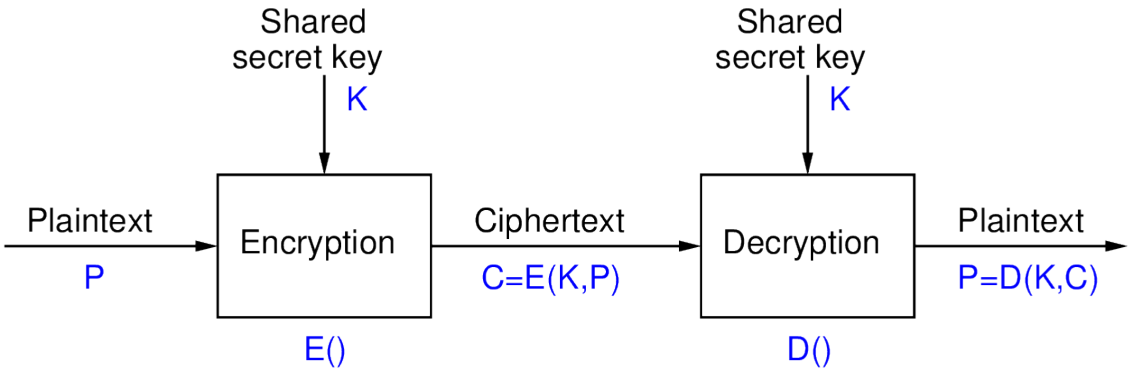
Key Management

Digital Signatures

Random Numbers

Summary

Symmetric Key Encryption for Confidentiality



Requirements

- Strong encryption algorithm: given algorithm, ciphertext and known pairs of (plaintext, ciphertext), attacker should be unable to find plaintext or key
- Shared secret keys: sender and receiver both have shared a secret key; no-one else knows the key

Block vs Stream Ciphers

Block Ciphers

- Encrypt block of plaintext at a time, typically 64 or 128 bits
- Slow algorithms/implementations
- Can re-use keys

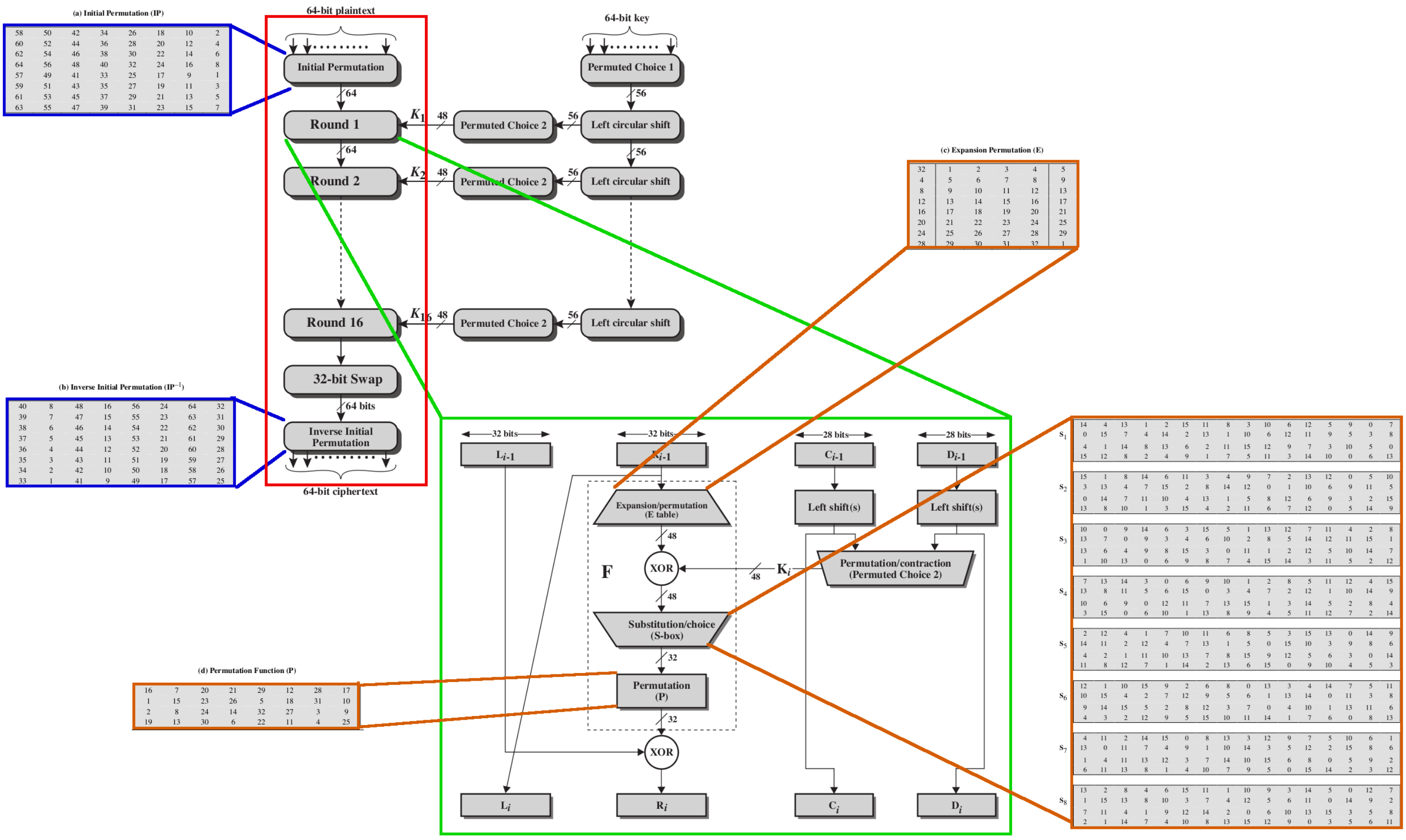
Stream Ciphers

- Encrypt 1 byte of plaintext at a time
- Encryption performed by XOR plaintext with keystream (created by pseudo-random number generator)
- Fast algorithms/implementations
- Cannot re-use keys

Data Encryption Standard (DES)

- Designed by IBM and NSA; standardised by NIST in 1977 as FIPS-46
  - 1999: NIST recommended Triple-DES; DES only for legacy systems
  - 2005: FIPS-46 standard withdrawn
- Block size: 64 bits
- Key length: 56 bits (64 bits, but 8 are parity)
- Initial and final permutations, then 16 rounds, each involving permutations and substitutions
- Feistel structure
- Decryption is almost identical to encryption $\rightarrow$ single implementation for both algorithms
- Key size is insecure; algorithm considered secure
- Status: not recommended

DES Encryption Operations



(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

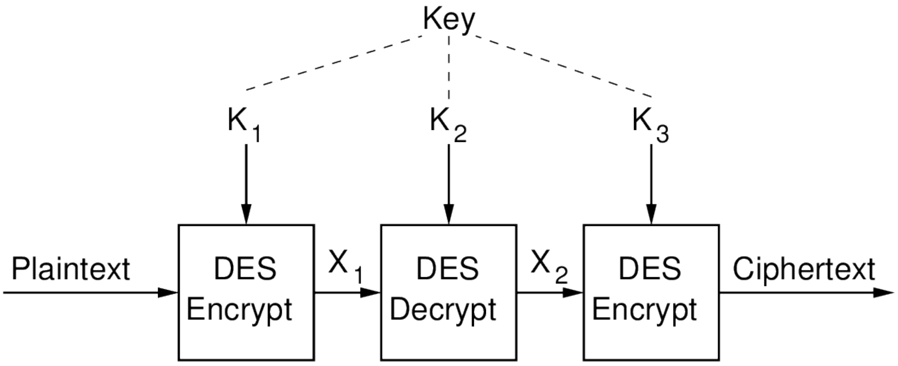
(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Triple-DES (3DES)

- Standardised by ANSI/NIST in 1998/99
- Applies DES three times: Encrypt, Decrypt, Encrypt
- Block size: 64 bits
- Key length: 168 bits (options for 112 and 56 bits)
- Three times slower than DES
- Status: banks still use in many applications; available as an option in many products



Advanced Encryption Standard (AES)

- NIST held competition to select algorithm to replace DES/3DES in 1997
  - Won by Rijndael algorithm by Rijmen and Daemen
  - 2001: Standardised as FIPS-197
- Block size: 128
- Key length: 128, 192, 256 bits
- Substitution-permutation network
- Status: used in many products, e.g. WiFi (WPA), full disk encryption (BitLocker, FileVault2, dm-crypt, LUKS), Internet security (HTTPS), ...

Other Symmetric Encryption Algorithms

- Blowfish (Schneier, 1993): 64 bit blocks/32–448 bit keys; Feistel structure
- Twofish (Schneier et al, 1998): 128/128, 192, 256; Feistel structure
- Serpent (Anderson et al, 1998): 128/128, 192, 256; Substitution-permutation network
- Camellia (Mitsubishi/NTT, 2000): 128/128, 192, 256; Feistel structure
- IDEA (Lai and Massey, 1991): 64/128
- CAST-128 (Adams and Tavares, 1996): 64/40–128; Feistel structure
- CAST-256 (Adams and Tavares, 1998): 128/up to 256; Feistel structure
- RC5 (Rivest, 1994): 32, 64 or 128/up to 2040; Feistel-like structure
- RC6 (Rivest et al, 1998): 128/128, 192, 256; Feistel structure

Assumptions: Symmetric Key Encryption

- The same secret key, K, is used for encryption, E(), and decryption, D(). The secret is shared between two entities, i.e. K_{AB}.
- Encrypting plaintext, P, with a key, produces ciphertext C, e.g. $C = \mathrm{E}(K_{AB}, P)$.
- Decrypting ciphertext with the correct key will produce the original plaintext. The decrypter will be able to recognise that the plaintext is correct (and therefore the key is correct). E.g. $P = \mathrm{D}(K_{AB}, C)$.
- Decrypting ciphertext using the incorrect key will *not* produce the original plaintext. The decrypter will be able to recognise that the key is wrong, i.e. the decryption will produce unrecognisable output.

Attacks on Block Ciphers

Brute Force Attack

- Approach: try all keys in key space
- Metric: number of operations (time)
- k bit key requires 2^k operations
- Depends on key length and computer speed

Cryptanalysis

- Approach: Find weaknesses in algorithms
- Methods: Linear cryptanalysis, differential cryptanalysis, meet-in-the-middle attack, side-channel attacks ...
- Metrics:
  - Number of operations
  - Amount of memory
  - Number of known plaintexts/ciphertexts

Brute Force Attacks on Block Ciphers

Key	Key	Worst case time at speed:		
length	space	10^9/sec	10^{12}/sec	10^{15}/sec
32	2^{32}	4 sec	4 ms	4 us
56	2^{56}	833 days	20 hrs	72 sec
64	2^{64}	584 yrs	213 days	5 sec
128	2^{128}	10^{22} yrs	10^{19} yrs	10^{16} yrs
192	2^{192}	10^{41} yrs	10^{38} yrs	10^{35} yrs
256	2^{256}	10^{60} yrs	10^{57} yrs	10^{54} yrs
26!	2^{88}	10^{10} yrs	10^{7} yrs	10^{4} yrs

Age of Earth: 4×10^9 years
Age of Universe: 1.3×10^{10} years

How Fast/Expensive is a Brute Force Attack Today?

Cryptanalysis on Block Ciphers

Cipher	Method	Key	Required resources:		
		space	Time	Memory	Known data
DES	Brute force	2^{56}	2^{56}	-	-
3DES	MITM	2^{168}	2^{111}	2^{56}	2^{2}
3DES	Lucks	2^{168}	2^{113}	2^{88}	2^{32}
AES 128	Biclique	2^{128}	$2^{126.1}$	2^{8}	2^{88}
AES 256	Biclique	2^{256}	$2^{254.4}$	2^{8}	2^{40}

- Known data: chosen pairs of (plaintext, ciphertext)
- MITM: Meet-in-the-middle
- Lucks: S. Lucks, Attacking Triple Encryption, in *Fast Software Encryption*, Springer, 1998
- Biclique: Bogdanov, Khovratovich and Rechberger, Biclique Cryptanalysis of the Full AES, in *ASIACRYPT2011*, Springer, 2011

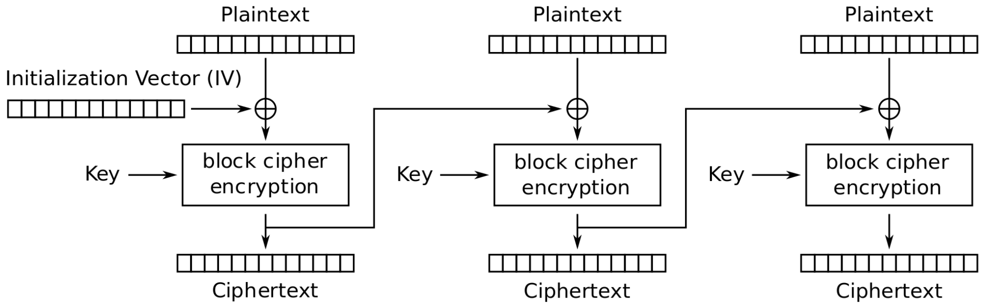
Assumptions: Knowledge of Attacker

- All algorithms used in cryptography, e.g. encryption/decryption algorithms, hash functions, are public.
- An attacker knows which algorithm is being used, and any public parameters of the algorithm.
- An attacker can intercept any message sent across a network.
- An attacker does not know secret values (e.g. symmetric secret key K_{AB} or private key PR_A).
- Brute force attacks requiring greater than 2^{80} operations are impossible.

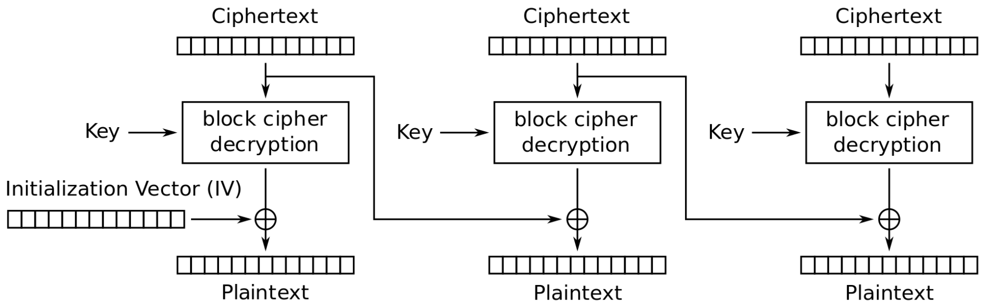
Using Block Ciphers on Real Data

- Block ciphers typical operate on 64 or 128 bit blocks
- Modes of operation are used to apply ciphers on multiple blocks
  - Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback Mode (CFB), Output Feedback Mode (OFB), Counter (CTR), XTS-AES
- Trade-offs: security, parallelism, error propagation
- Often require Initialisation Vector (IV)
- CFB, OFB and CTR can turn block cipher into stream cipher

Mode of Operation Example: CBC



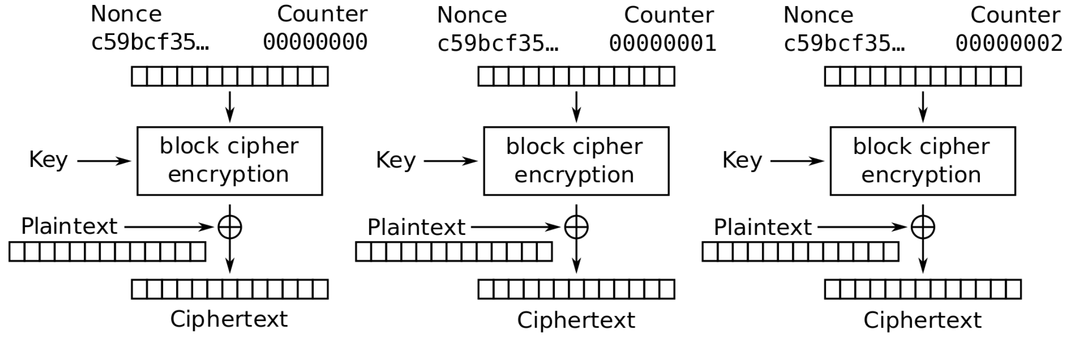
Cipher Block Chaining (CBC) mode encryption



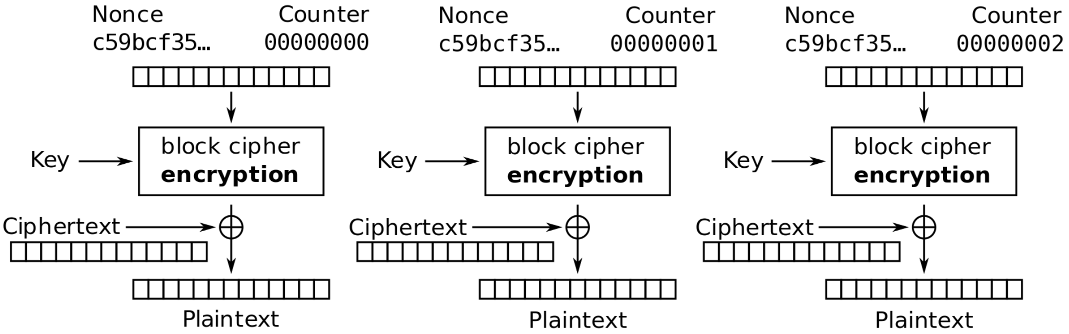
Cipher Block Chaining (CBC) mode decryption

Credit: WhiteTimberwolf, Wikimedia Commons, Public Domain

Mode of Operation Example: CTR



Counter (CTR) mode encryption

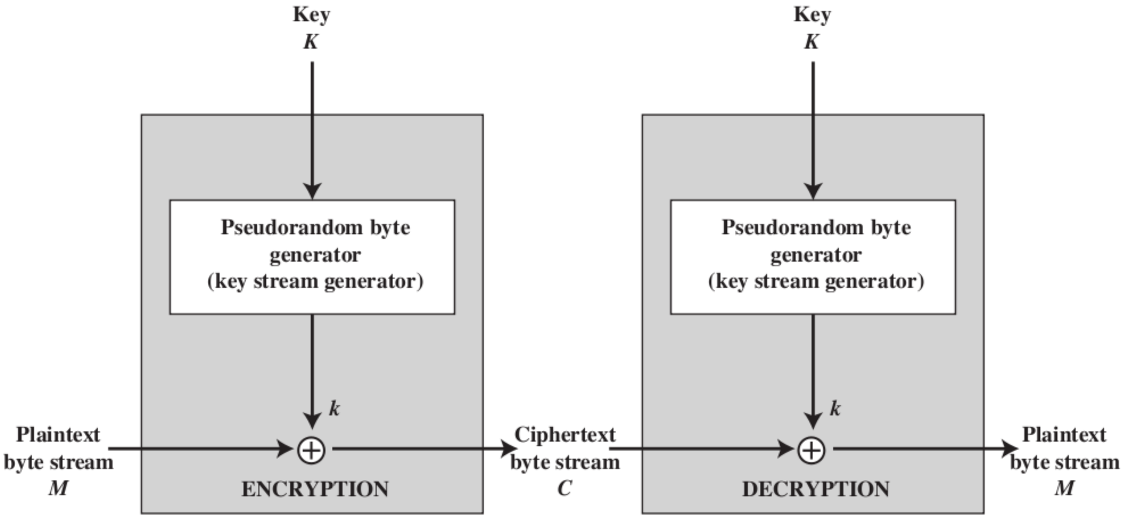


Counter (CTR) mode decryption

Credit: WhiteTimberwolf, Wikimedia Commons, Public Domain

Stream Ciphers

- Encrypt one byte at a time by XOR with pseudo-random byte (keystream)
- Generally faster implementations than block ciphers
- Keystream should not repeat (large period); use different key or nonce when re-using cipher



Credit: Figure 7.5 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

Example Stream Cipher: RC4

- Designed by Ron Rivest in 1987
- Used in secure web browsing and wireless LANs
- Can use variable size key: 8 to 2048 bits
- Several theoretical limitations of RC4

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

Key Management

Digital Signatures

Random Numbers

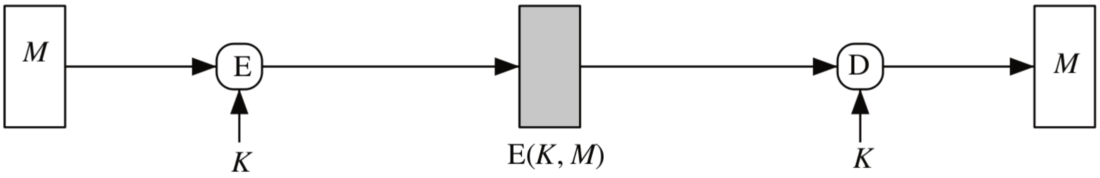
Summary

Authentication

- Receiver wants to verify:
  1. Contents of the message have not been modified (*data authentication*)
  2. Source of message is who they claim to be (*source authentication*)
- Different approaches available:
  - Symmetric Key Encryption
  - Message Authentication Codes
  - Hash Functions
  - Public Key Encryption (see Digital Signatures)

Authentication using Symmetric Key Encryption

- Assumption: decryption using wrong key or modified ciphertext will produce unintelligible output
- Symmetric key encryption can provide: data authentication and source authentication (as well as confidentiality)



Credit: Figure 12.1(a) in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

- However, typically authentication is performed separately to encryption for confidentiality
  - Avoid overhead of using encryption when not needed

Authentication using Message Authentication Codes

- Append small, fixed-size block of data to message: cryptographic checksum or MAC

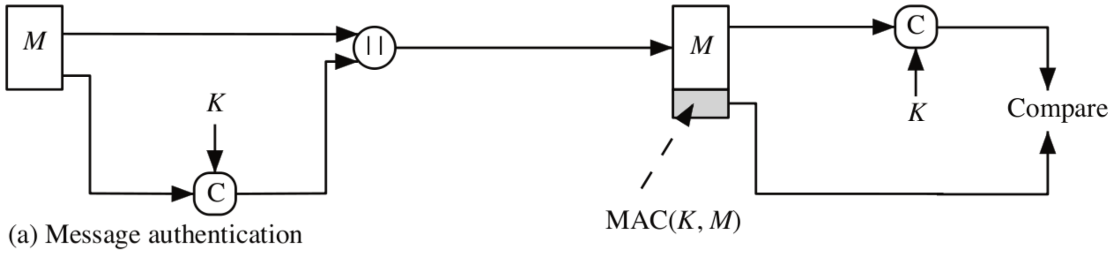
$$\mathrm{MAC} = \mathrm{F}(K, M)$$

M = input message
F = MAC function
K = shared secret key of k bits
MAC = message authentication code (or tag) of n bits

- MAC function also called *keyed hash function*
- MAC function similar to encryption, but does not need to be reversible
  - Easier to design stronger MAC functions than encryption functions

---

Authentication using Message Authentication Codes



(a) Message authentication

Credit: Figure 12.4(a) in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

MAC Algorithms

- Data Authentication Algorithm (DAA): based on DES; considered insecure
- Cipher-Based Message Authentication Code (CMAC): mode of operation used with Triple-DES and AES
- OMAC, PMAC, UMAC, VMAC, ...
- HMAC: MAC function derived from cryptographic hash functions
  - MD5/SHA are fast in software (compared to block ciphers)
  - Libraries for hash functions widely available
  - Security of HMAC depends on security of hash function used

---

MAC Attacks

Security Requirement

- Key is secret and difficult to find from pairs of (M, MAC)
- Given pairs of (M, MAC), difficult to find the MAC of another message

Brute Force Attacks on MACs

- Option 1: Try all possible keys for one or more pairs of (MAC, M); effort $\approx 2^k$
- Option 2: Try many values of M to find correct MAC; effort $\approx 2^n$
- Effort to break MAC: $\min(2^k, 2^n)$

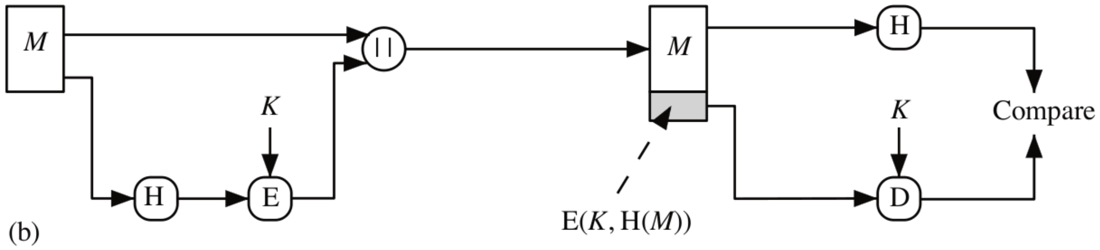
Assumptions: Authentication with Symmetric Key and MACs

- An entity receiving ciphertext that successfully decrypts with symmetric secret key K_{AB} knows that the original message has not been modified and that it originated at one of the owners of the secret key (i.e. A or B).
- An entity receiving a message with attached MAC that successfully verifies, knows that the message has not been modified and originated at one of the owners of the MAC secret key.

Authentication using Hash Functions

- Hash function H: variable-length block of data M input; fixed-size hash value $h = \mathrm{H}(M)$ output
- Applying H to large set of inputs should produce evenly distributed and random looking outputs
- Cryptographic hash function: computationally infeasible to find:
  1. M that maps to known h (one-way property)
  2. M_1 and M_2 that produce same h (collision-free property)
- Append hash value to message; receiver verifies if message changed

Example of Authentication with Hash functions



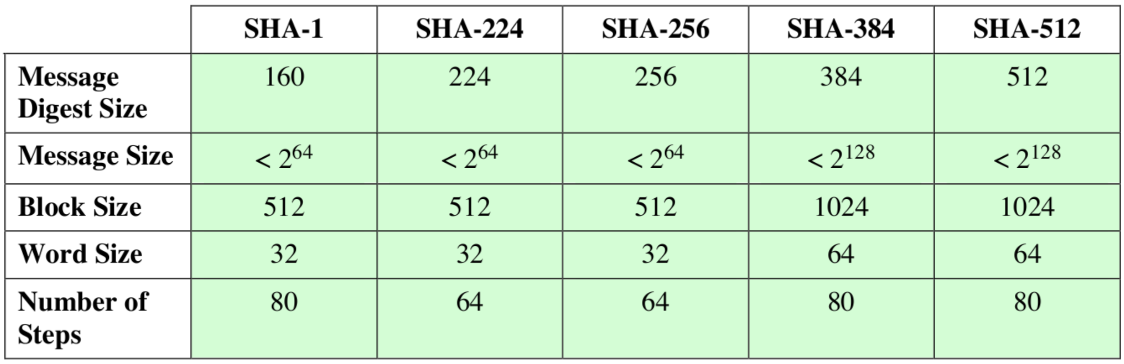
Credit: Figure 11.2(b) in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

Hash Algorithms: MD5

- Message Digest algorithm 5, developed by Ron Rivest in 1991
- Standardised by IETF in RFC 1321
- Generates 128-bit hash
- Was commonly used by applications, passwords, file integrity; no longer recommended
- Collision and other attacks possible; tools publicly available to attack MD5

Hash Algorithms: SHA

- Secure Hash Algorithm, developed by NIST
- Standardised by NIST in FIPS 180 in 1993
- Improvements over time: SHA-0, SHA-1, SHA-2, SHA-3
- SHA-1 (and SHA-0) are considered insecure; no longer recommended
- SHA-2 considered secure
- SHA-3 in begin standardised by NIST



	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Credit: Table 11.3 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

Hash Attacks

Security Requirement

Preimage resistant: For any given h, computationally infeasible to find y such that $\mathrm{H}(y) = h$ (*one-way property*)

Second preimage resistant: For any given x, computationally infeasible to find $y \neq x$ with $\mathrm{H}(y) = \mathrm{H}(x)$ (*weak collision resistant*)

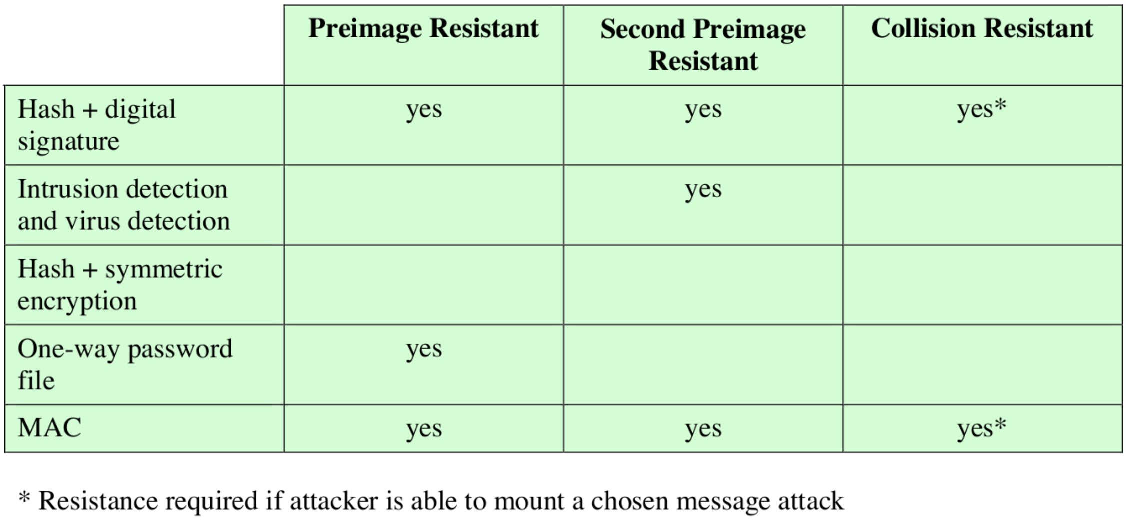
Collision resistant: Computationally infeasible to find any pair (x, y) such that $\mathrm{H}(x) = \mathrm{H}(y)$ (*strong collision resistant*)

Brute Force Attacks

- Depend on hash value length of n bits
- Preimage and second preimage resistant: 2^n
- Collision resistant: $2^{n/2}$

Required Properties when using Hash Functions

Not all applications of hash functions require all properties



	Preimage Resistant	**Second Preimage Resistant**	**Collision Resistant**
Hash + digital signature	yes	yes	yes*
Intrusion detection and virus detection		yes	
Hash + symmetric encryption			
One-way password file	yes		
MAC	yes	yes	yes*

* Resistance required if attacker is able to mount a chosen message attack

Credit: Table 11.2 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

How Fast/Expensive is a MD5 Collision Attack Today?

Assumptions: Hash Functions

- A cryptographic hash function, $\mathrm{H}()$, takes a variable sized input message, M, and produces a fixed size, small output hash, h, i.e. $h = \mathrm{H}(M)$.
- Given a hash value, h, it is impossible to find the original message M.
- Given a hash value, h, it is impossible to find another message M' that also has a hash value of h.
- It is impossible to find two messages, M and M', that have the same hash value.

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

Key Management

Digital Signatures

Random Numbers

Summary

Birth of Public-Key Cryptosystems

- Beginning to 1960's: permutations and substitutions (Caesar, rotor machines, DES, . . .)
- 1960's: NSA secretly discovered public-key cryptography
- 1970: first known (secret) report on public-key cryptography by CESG, UK
- 1976: Diffie and Hellman public introduction to public-key cryptography
  - Avoid reliance on third-parties for key distribution
  - Allow digital signatures
- 1978: Rivest, Shamir and Adlemen created RSA

Principles of Public-Key Cryptosystems

- Symmetric algorithms used same secret key for encryption and decryption
- Asymmetric algorithms in public-key cryptography use one key for encryption and different but related key for decryption
- Characteristics of asymmetric algorithms:
  - Require: Computationally infeasible to determine decryption key given only algorithm and encryption key
  - Optional: Either of two related keys can be used for encryption, with other used for decryption

Public and Private Keys

Public Key

- For secrecy: used in encryption
- For authentication: used in decryption
- Available to anyone

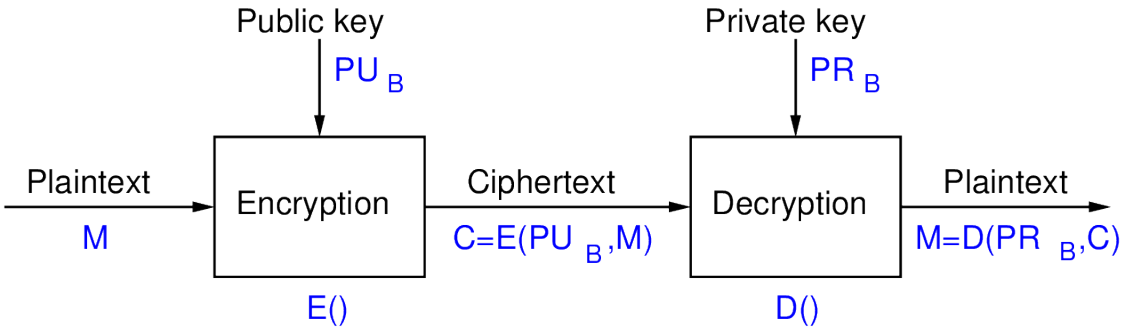
Private Key

- For secrecy: used in decryption
- For authentication: used in decryption
- Secret, known only by owner

Public-Private Key Pair

- User *A* has pair of related keys, public and private: (PU_A, PR_A)

Confidentiality with Public Key Crypto



Public key

PU_B

Private key

PR_B

Plaintext

M

Encryption

E()

Ciphertext

$C=E(PU_B,M)$

Decryption

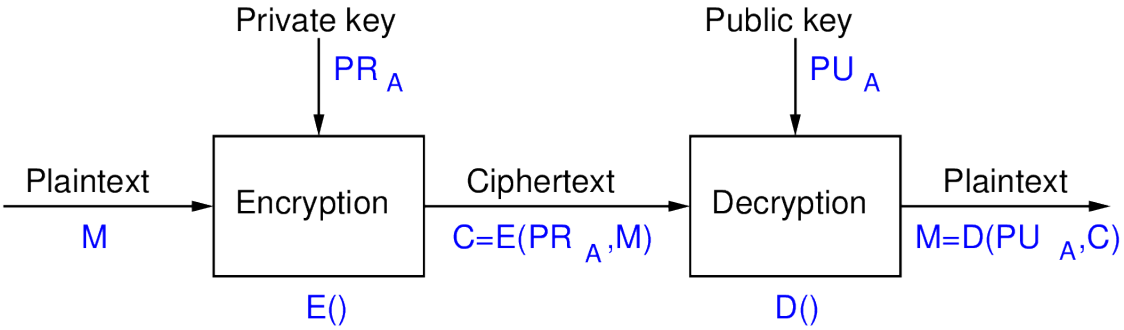
D()

Plaintext

$M=D(PR_B,C)$

- Encrypt using receivers public key
- Decrypt using receivers private key
- Only the person with private key can successful decrypt

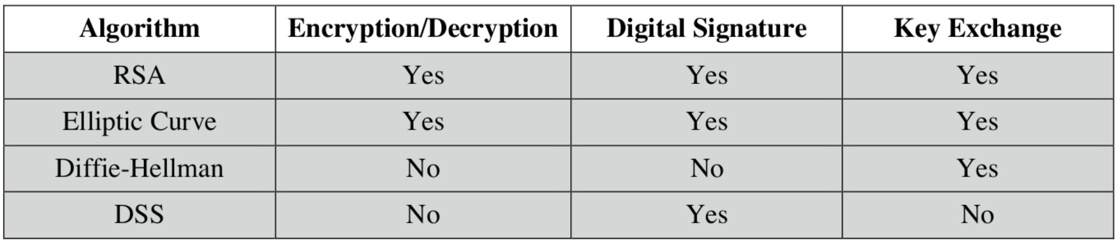
Authentication with Public Key Crypto



- Encrypt using senders private key
- Decrypt using senders public key
- Only the person with private key could have encrypted

Applications of Public Key Cryptosystems

- Secrecy, encryption/decryption of messages
- Digital signature, *sign* message with private key
- Key exchange, share secret session keys



Algorithm	**Encryption/Decryption**	**Digital Signature**	**Key Exchange**
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Credit: Table 9.3 in Stallings, *Cryptography and Network Security*, 5th Ed., Pearson 2011

Requirements of Public-Key Cryptography

1. Computationally easy for B to generate pair (PU_b,PR_b)
2. Computationally easy for A, knowing PU_b and message M, to generate ciphertext:

$$C = \mathrm{E}(PU_b, M)$$

3. Computationally easy for B to decrypt ciphertext using PR_b:

$$M = \mathrm{D}(PR_b, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

4. Computationally infeasible for attacker, knowing PU_b and C, to determine PR_b
5. Computationally infeasible for attacker, knowing PU_b and C, to determine M
6. (Optional) Two keys can be applied in either order:

$$M = \mathrm{D}[PU_b, \mathrm{E}(PR_b, M)] = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

Requirements of Public-Key Cryptography

6 requirements lead to need for trap-door one-way function

- Every function value has unique inverse
- Calculation of function is easy
- Calculation of inverse is infeasible, unless certain information is known

$$\begin{array}{ll} Y = f_k(X) & \text{easy, if } k \text{ and } Y \text{ are known} \\ X = f_k^{-1}(Y) & \text{easy, if } k \text{ and } Y \text{ are known} \\ X = f_k^{-1}(Y) & \text{infeasible, if } Y \text{ is known but } k \text{ is not} \end{array}$$

- What is easy? What is infeasible?
    - Computational complexity of algorithm gives an indication
    - Easy if can be solved in polynomial time as function of input

Public-Key Cryptanalysis

Brute Force Attacks

- Use large key to avoid brute force attacks
- Public key algorithms less efficient with larger keys
- Public-key cryptography mainly used for key management and signatures

Compute Private Key from Public Key

- No known feasible methods using standard computing

Probable-Message Attack

- Encrypt all possible M' using PU_b—for the C' that matches C, attacker knows M
- Only feasible of M is short
- Solution for short messages: append random bits to make it longer

---

Example Public-Key Algorithm: RSA

Key Generation

1. Choose primes p and q, and calculate $n = pq$
2. Select e: $gcd(\phi(n), e) = 1, 1 < e < \phi(n)$
3. Find $d \equiv e^{-1} \pmod{\phi(n)}$

n and e are public; p, q and d are private

Encryption

Encryption of plaintext M, where $M < n$:

$$C = M^e \bmod n$$

Decryption

Decryption of ciphertext C:

$$M = C^d \bmod n$$

Example Public-Key Algorithm: RSA

- Created by Ron Rivest, Adi Shamir and Len Adleman in 1978
- Security of RSA
  1. Brute force attack on d
  2. Factor n into its two prime factors
  3. Determine $\phi(n)$ directly, without determining p or q
  4. Determine d directly, without determining $\phi(n)$
- Factoring is considered the easiest. Some records by length of n:
  - 1991: 330 bits (100 digits)
  - 2003: 576 bits (174 digits)
  - 2005: 640 bits (193 digits)
  - 2009: 768 bit (232 digits), 10^{20} operations, 2000 years on single core 2.2 GHz computer
- Typical length of n: 1024 bits, 2048 bits, 4096 bits

Performance of Public Key Cryptography

- Public key crypto algorithms typically much slower than symmetric key algorithms

Assumptions: Public Key Encryption

- There is a pair of keys, public (PU) and private (PR). One key from the pair is used for encryption, the other is used for decryption. Each entity has their own pair, e.g. (PU_A, PR_A).
- Encrypting a plaintext message, M, with a key, produces ciphertext C, e.g. $C = \mathrm{E}(PU_A, M)$.
- Decrypting ciphertext with the correct key will produce the original plaintext. The decrypter will be able to recognise that the plaintext is correct (and therefore the key is correct). E.g. $M = \mathrm{D}(PR_A, C)$.
- Decrypting ciphertext using the incorrect key will *not* produce the original plaintext. The decrypter will be able to recognise that the key is wrong, i.e. the decryption will produce unrecognisable output.

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

Key Management

Digital Signatures

Random Numbers

Summary

Key Management

Challenges

- How to share a secret key?
- How to obtain someone else's public key?
- When to change keys?

Assumptions and Principles

- Many users wish to communicate securely across network
- Attacker can intercept any location in network
- Manual interactions between users are undesirable (e.g. physical exchange of keys)
- More times a key is used, greater chance for attacker to discover the key

---

Exchanging Secret Keys

Option 1: Manual Exchange of All Keys

- All users exchange secret keys with all other users manually (e.g. face-to-face)
- Inconvenient

Option 2: Manual Exchange of Master Keys

- All users exchange master key with trusted, central entity (e.g. Key Distribution Centre)
- Session keys automatically exchanged between users via KDC
- Security and performance bottleneck at KDC

Exchanging Secret Keys

Option 3: Public Key Cryptography to Exchange Secrets

- Use public-key cryptography to securely and automatically exchange secret keys
- Example 1: user A encrypts secret with user B's public key; sends to B
- Example 2: Diffie-Hellman secret key exchange

Distributing Public Keys

- By design, public keys are made public
- Issue: how to ensure public key of A actually belongs to A (and not someone pretending to be A)
- Approaches for public key distribution
  1. Public announcement (web page, email, newspaper)
  2. Publish in electronic directory (which manually authenticates users)
  3. Public key authority:
     - Users manually publish key at authority, and gain authorities public key
     - Users automatically request other users public keys from authority
  4. Public key certificates
     - Users manually register with authority
     - Authority issues certificates to users: users public key signed by authority
     - Users automatically exchange certificates

Key Hierarchy and Lifetimes

- Master keys used to securely exchange session keys
- Session keys used to securely exchange data
- Change session keys automatically and regularly
- Change master keys manually and seldom
- Session key lifetime:
  - Shorter lifetime is more secure; but increases overhead of exchanges
  - Connection-oriented protocols (e.g. TCP): new session key for each connection
  - Connection-less protocols (e.g. UDP/IP): change after fixed period or certain number of packets sent

---

Assumptions: Key Management

- A secret key can be exchanged between two entities without other entities learning its value.
- Any entity can obtain the correct public key of any other entity.

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

Key Management

Digital Signatures

Random Numbers

Summary

Digital Signatures

- Aim of a signature: prove to anyone that a message originated at (or is approved by) a particular user
- Symmetric key cryptography
  - Two users, A and B, share a secret key K
  - Receiver of message (user A) can verify that message came from the other user (B)
  - User C *cannot* prove that the message came from B (it may also have came from A)
- Public key cryptography can provide signature: only one user has the private key

Digital Signature Operations (Concept)

Signing

- User signs a message by encrypting with own private key

$$S = E(PR_A, M)$$

- User attaches signature to message

Verification

- User verifies a message by decrypting signature with signer's public key

$$M' = D(PU_A, S)$$

- User then compares received message M with decrypted M'; if identical, signature is verified

---

Digital Signature Operations (Practice)

No need to encrypt entire message; encrypt hash of message

Signing

- User signs a message by encrypting hash of message with own private key

$$S = E(PR_A, H(M))$$

- User attaches signature to message

Verification

- User verifies a message by decrypting signature with signer's public key

$$h = D(PU_A, S)$$

- User then compares hash of received message, $H(M)$, with decrypted h; if identical, signature is verified

Digital Signature Algorithms

- RSA
- Digital Signature Algorithm (DSA): FIPS-186
- ECDSA: DSA with elliptic curve cryptography
- ElGamal signature scheme: DSA is enhancement of ElGamal
- Bilinear pairing based signatures, e.g. BLS
- Different hash algorithms can be used; e.g. SHA2
    - Preimage resistant, second preimage resistant, collision resistant

---

Assumptions: Digital Signatures

- A digital signature of a message M is the hash of that message encrypted with the signers private key, i.e. $S = \mathrm{E}(PR, \mathrm{H}(M))$
- An entity receiving a message with an attached digital signature knows that that message originated by the signer of the message.

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

Key Management

Digital Signatures

Random Numbers

Summary

Random Numbers

Examples of Random Numbers in Cryptography

- Generate keys in public key algorithms
- Keystream in stream ciphers
- Generate session keys for symmetric ciphers
- Authentication and key distribution protocols to prevent replays

Requirements of Sequence of Random Numbers

- Randomness, e.g. selecting large prime numbers for RSA involves selecting random numbers and checking that they are not composite
  - Uniform distribution
  - Independence
- Unpredictability, e.g. protocols relay on unpredictable values so attacker cannot generate fake/replay messages

Random vs Pseudo-random

Pseudo-random Number Generators (PRNG)

- Algorithms used to generate random numbers
- Algorithms are deterministic, therefore numbers produced are not statistically unpredictable or independent
- However good algorithms produce sequences of number that pass many randomness tests

True Random Number Generators

- Use non-deterministic source to produce randomness
  - Measure ionising radiation events, leaky capacitors, thermal noise from resistor, disk reads of hard disks, . . .
- Require hardware for measurements

Pseudo Random Number Generators

Characteristics

- Seed: initial state of algorithm
- Period: length of sequence produced before repeating

Examples

- Linear congruential generators
- Linear feedback shift registers
- Blum Blum Shub
- Mersenne Twister
- Stream and block ciphers

Assumptions: Random Numbers

- Pseudo-random number generators (PRNG) can generate effectively true random numbers.

Contents

Encryption for Confidentiality

Symmetric Key Encryption

Authentication and Hash Functions

Public Key Encryption

Key Management

Digital Signatures

Random Numbers

Summary

Key Points

- Symmetric key encryption used for file and network data confidentiality (AES, 3DES)
- Public key encryption used for key exchange and source authentication (RSA, ECC, DH, certificates)
- MAC functions used for data and source authentication (HMAC)
- Hash function used for data authentication (MD5, SHA)
- Public key crypto combined with hash functions for digital signatures
- Random numbers used in many security algorithms and protocols

Common Principles used in Security

- *Experience*: Algorithms that have been used over a long period are less likely to have security flaws than newer algorithms.
- *Performance*: Symmetric key algorithms are significantly faster than public key algorithms.
- *Performance*: The time to complete a cryptographic operation is linearly proportional with the input data size.
- *Key Distribution*: Keys should be distributed using automatic means.
- *Key Re-use*: The more times a key is used, the greater the chance of an attacker discovering that key.
- *Multi-layer Security*: Using multiple overlapping security mechanisms can increase the security of a system.

Security Issues

- Key management and distribution: difficult to confirm that public key belongs to claimed entity
- Implementation: flaws in software implementations can weaken otherwise secure algorithms
- Algorithm design: difficult to prove security of algorithms; where the design decisions well motivated, public?

Areas To Explore

- Elliptic Curve Cryptography
- Steganography
- Quantum Cryptography