Encryption and Security Tutorial

Peter Gutmann

University of Auckland

http://www.cs.auckland.ac.nz/~pgut001

Security Requirements

Confidentiality

- Protection from disclosure to unauthorised persons

Integrity

- Maintaining data consistency

Authentication

- Assurance of identity of person or originator of data

Non-repudiation

- Originator of communications can’t deny it later

Security Requirements (ctd)

Availability

- Legitimate users have access when they need it

Access control

- Unauthorised users are kept out

These are often combined

- User authentication used for access control purposes
- Non-repudiation combined with authentication

Security Threats

Information disclosure/information leakage

Integrity violation

Masquerading

Denial of service

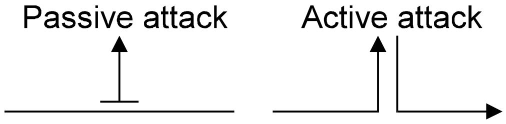
Illegitimate use

Generic threat: Backdoors, trojan horses, insider attacks

Most Internet security problems are access control or authentication ones

- Denial of service is also popular, but mostly an annoyance

Attack Types



Passive attack can only observe communications or data

Active attack can actively modify communications or data

- Often difficult to perform, but very powerful
  - Mail forgery/modification
  - TCP/IP spoofing/session hijacking

Security Services

From the OSI definition:

- Access control: Protects against unauthorised use
- Authentication: Provides assurance of someone's identity
- Confidentiality: Protects against disclosure to unauthorised identities
- Integrity: Protects from unauthorised data alteration
- Non-repudiation: Protects against originator of communications later denying it

Security Mechanisms

Three basic building blocks are used:

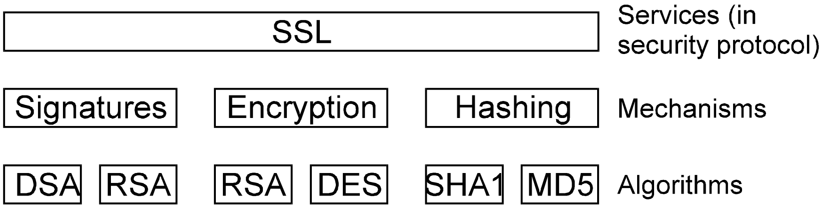
- Encryption is used to provide confidentiality, can provide authentication and integrity protection
- Digital signatures are used to provide authentication, integrity protection, and non-repudiation
- Checksums/hash algorithms are used to provide integrity protection, can provide authentication

One or more security mechanisms are combined to provide a security service

---

Services, Mechanisms, Algorithms

A typical security protocol provides one or more services



SSL — Services (in security protocol)

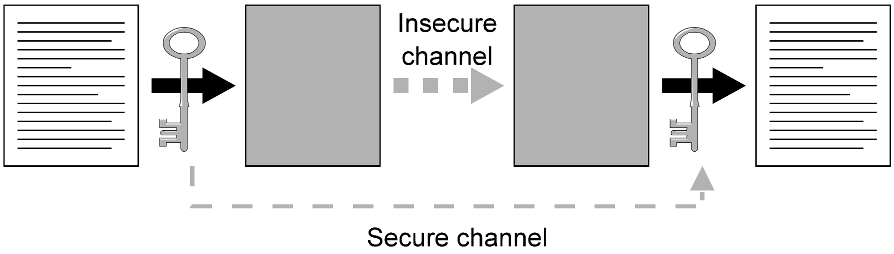
Signatures, Encryption, Hashing — Mechanisms

DSA, RSA, RSA, DES, SHA1, MD5 — Algorithms

- Services are built from mechanisms
- Mechanisms are implemented using algorithms

Conventional Encryption

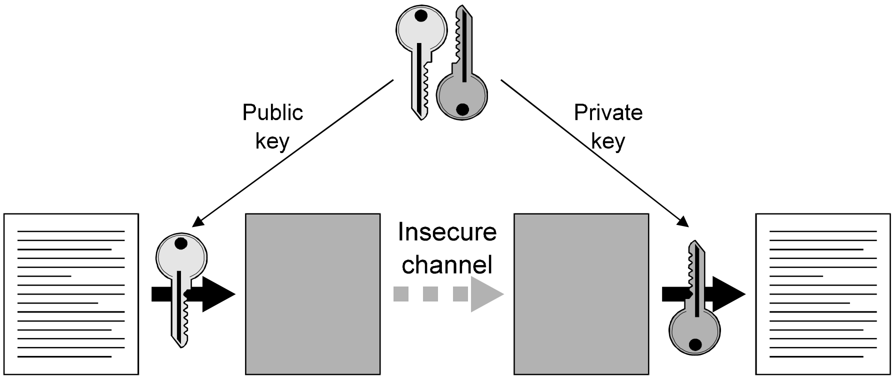
Uses a shared key



Problem of communicating a large message in secret reduced to communicating a small key in secret

Public-key Encryption

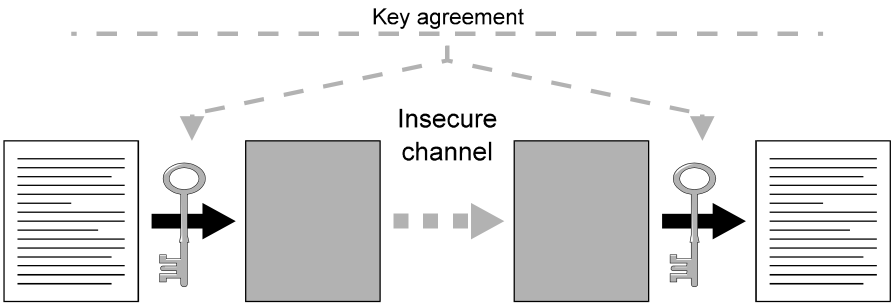
Uses matched public/private key pairs



Anyone can encrypt with the public key, only one person can decrypt with the private key

Key Agreement

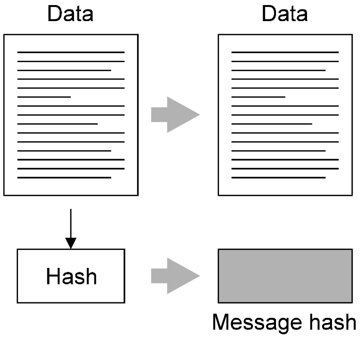
Allows two parties to agree on a shared key



Provides part of the required secure channel for exchanging a conventional encryption key

Hash Functions

Creates a unique “fingerprint” for a message

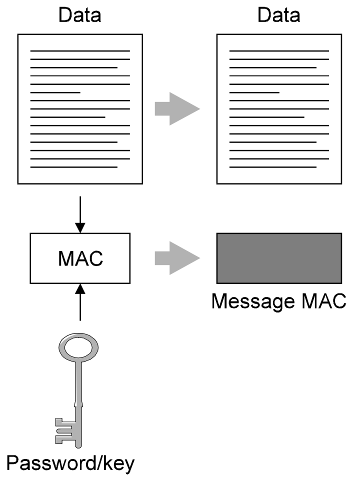


Anyone can alter the data and calculate a new hash value

- Hash has to be protected in some way

MAC's

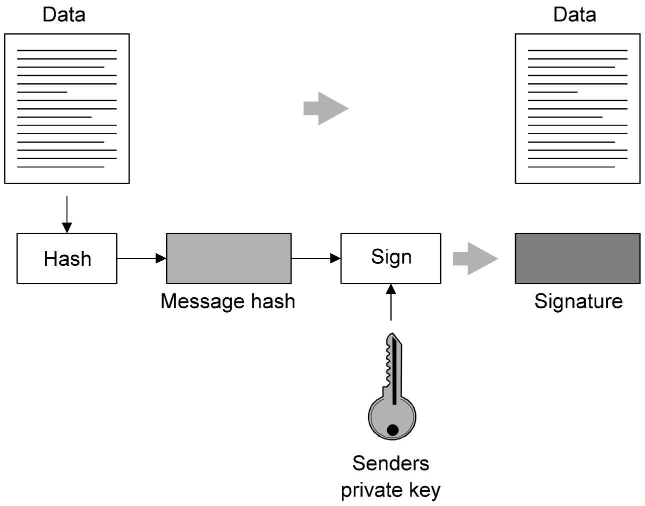
Message Authentication Code, adds a password/key to a hash



Only the password holder(s) can generate the MAC

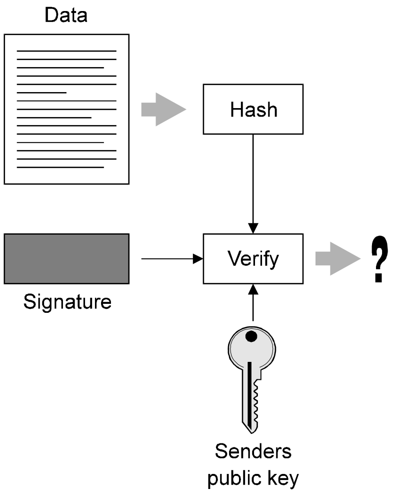
Digital Signatures

Combines a hash with a digital signature algorithm



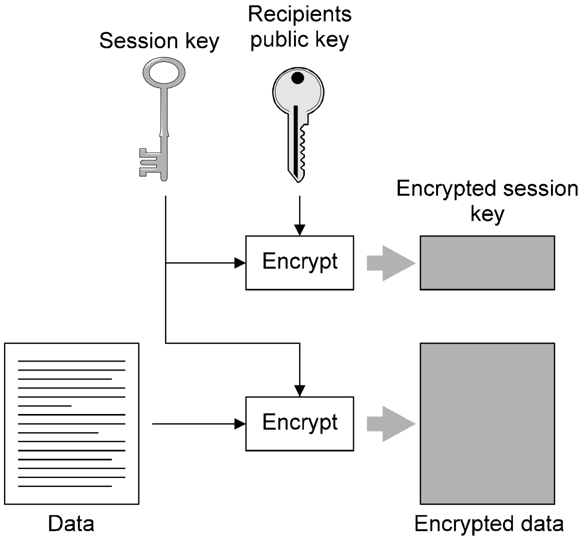
Digital Signatures (ctd)

Signature checking:

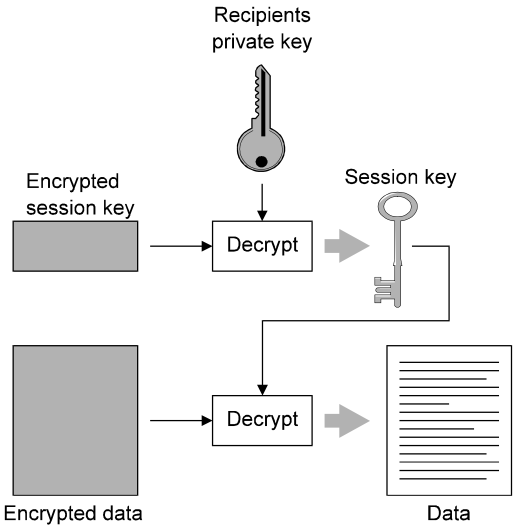


Message/Data Encryption

Combines conventional and public-key encryption

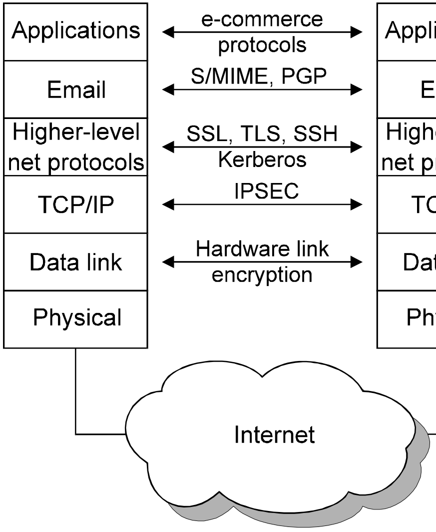


Message/data Encryption (ctd)



Public-key encryption provides a secure channel to exchange conventional encryption keys

Security Protocol Layers



The further down you go, the more transparent it is

The further up you go, the easier it is to deploy

Encryption and Authentication Algorithms and Technology

> Cryptography is nothing more than a mathematical framework for discussing the implications of various paranoid delusions
>
> - Don Alvarez

Historical Ciphers

Nonstandard hieroglyphics, 1900BC

Atbash cipher (Old Testament, reversed Hebrew alphabet, 600BC)

Caesar cipher:

letter = letter + 3

‘fish’ → ‘ilvk’

rot13: Add 13/swap alphabet halves

- Usenet convention used to hide possibly offensive jokes
- Applying it twice restores original text

Substitution Ciphers

Simple substitution cipher:

a = p, b = m, c = f, ...

Break via letter frequency analysis

Polyalphabetic substitution cipher

1. a = p, b = m, c = f, ...
2. a = l, b = t, c = a, ...
3. a = f, b = x, c = p, ...

Break by decomposing into individual alphabets, then solve as simple substitution

One-time Pad (1917)

Message	s	e	c	r	e	t
	18	5	3	17	5	19
OTP	+15	8	1	12	19	5
	7	13	4	3	24	24
	g	m	d	c	x	x

OTP is unbreakable *provided*

- Pad is never reused (VENONA)
- Unpredictable random numbers are used (physical sources, eg radioactive decay)

One-time Pad (ctd)

Used by

- Russian spies
- The Washington-Moscow “hot line”
- CIA covert operations

Many snake oil algorithms claim unbreakability by claiming to be a OTP

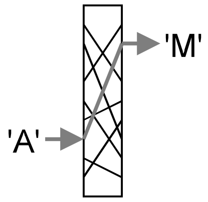
- Pseudo-OTP’s give pseudo-security

Cipher machines attempted to create approximations to OTP’s, first mechanically, then electronically

---

Cipher Machines (~1920)

1. Basic component = wired rotor



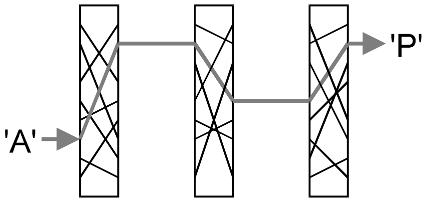
- Simple substitution

2. Step the rotor after each letter

- Polyalphabetic substitution, period = 26

Cipher Machines (ctd)

3. Chain multiple rotors



Each steps the next one when a full turn is complete

Cipher Machines (ctd)

Two rotors, period = 26 × 26

= 676

Three rotors, period = 26 × 26 × 26

= 17,576

Rotor sizes are chosen to be relatively prime to give maximum-length sequence

Key = rotor wiring

= rotor start position

Cipher Machines (ctd)

Famous rotor machines

- US: Converter M-209
- UK: TYPEX
- Japan: Red, Purple
- Germany: Enigma

Many books on Enigma

- Kahn, Siezing the Enigma
- Levin, Ultra Goes to War
- Welchman, The Hut Six Story
- Winterbothm, The Ultra Secret

"It would have been secure if used properly"

Use of predictable openings:

- "Mein Fuehrer! ..."
- "Nothing to report"

Use of the same key over an extended period

Encryption of the same message with old (compromised) and new keys

Device treated as a magic black box, a mistake still made today

Inventors believed it was infallible, "   "   "   "   "

Cipher Machines (ctd)

Various kludges made to try to improve security — none worked

Enigmas were sold to friendly nations after the war

Improved rotor machines were used into the 70’s and 80’s

Further reading:

- Kahn, The Codebreakers
- Cryptologia, quarterly journal

Stream Ciphers

Binary pad (keystream), use XOR instead of addition

Plaintext = original, unencrypted data
Ciphertext = encrypted data

Plaintext		1	0	0	1	0	1	1
Keystream	XOR	0	1	0	1	1	0	1
Ciphertext		1	1	0	0	1	1	0
Keystream	XOR	0	1	0	1	1	0	1
Plaintext		1	0	0	1	0	1	1

Two XOR’s with the same data always cancel out

Stream Ciphers (ctd)

Using the keystream and ciphertext, we can recover the plaintext

but

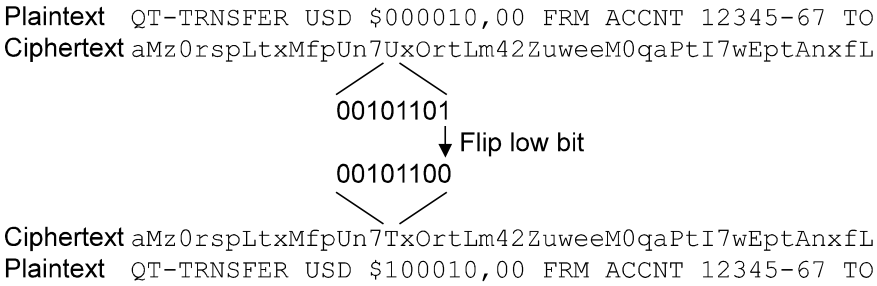
Using the plaintext and ciphertext, we can recover the keystream

Using two ciphertexts from the same keystream, we can recover the XOR of the plaintexts

- Any two components of an XOR-based encryption will recover the third
- Never reuse a key with a stream cipher
- Better still, never use a stream cipher

Stream Ciphers (ctd)

Vulnerable to bit-flipping attacks



```
Plaintext  QT-TRNSFER USD $000010,00 FRM ACCNT 12345-67 TO
Ciphertext aMz0rspLtxMfpUn7UxOrtLm42ZuweeM0qaPtI7wEptAnxfL
                     00101101
                             ↓ Flip low bit
                     00101100
Ciphertext aMz0rspLtxMfpUn7TxOrtLm42ZuweeM0qaPtI7wEptAnxfL
Plaintext  QT-TRNSFER USD $100010,00 FRM ACCNT 12345-67 TO
```

RC4

Stream cipher optimised for fast software implementation

2048-bit key, 8-bit output

Former trade secret of RSADSI, reverse-engineered and posted to the net in 1994

```
while( length-- )
  {
  x++; sx = state[ x ]; y += sx;
  sy = state[ y ]; state[ y ] = sx; state[ x ] = sy;
  *data++ ^= state[ ( sx+sy ) & 0xFF ];
  }
```

Takes about a minute to implement from memory

RC4 (ctd)

Extremely fast

Used in SSL (Netscape, MSIE), Lotus Notes, Windows password encryption, MS Access, Adobe Acrobat, MS PPTP, Oracle Secure SQL, ...

Usually used in a manner which allows the keystream to be recovered (Windows password encryption, early Netscape server key encryption, some MS server/browser key encryption, MS PPTP, Access, ...)

Illustrates the problem of treating a cipher as a magic black box

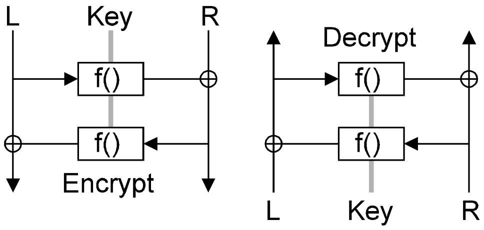
Recommendation: Avoid this, it’s too easy to get wrong

Block Ciphers

Originated with early 1970's IBM effort to develop banking security systems

First result was Lucifer, most common variant has 128-bit key and block size

- It wasn't secure in any of its variants



Called a Feistel or product cipher

Block Ciphers (ctd)

f()-function is a simple transformation, doesn't have to be reversible

Each step is called a round; the more rounds, the greater the security (to a point)

Most famous example of this design is DES:

- 16 rounds
- 56 bit key
- 64 bit block size (L,R = 32 bits)

Designed by IBM with, uh, advice from the NSA

Attacking Feistel Ciphers

Differential cryptanalysis

- Looks for correlations in f()-function input and output

Linear cryptanalysis

- Looks for correlations between key and cipher input and output

Related-key cryptanalysis

- Looks for correlations between key changes and cipher input/output

Differential cryptanalysis discovered in 1990; virtually all block ciphers from before that time are vulnerable...

...except DES. IBM (and the NSA) knew about it 15 years earlier

---

Strength of DES

Key size = 56 bits

Brute force = 2^{55} attempts

Differential cryptanalysis = 2^{47} attempts

Linear cryptanalysis = 2^{43} attempts

(but the last two are impractical)

> 56 bit keys don’t make it any stronger

> 16 rounds don’t make it any stronger

DES Key Problems

Key size = 56 bits

= 8 × 7-bit ASCII chars

Alphanumeric-only password converted to uppercase

= 8 × ~5-bit chars

= 40 bits

DES uses low bit in each byte for parity

= 32 bits

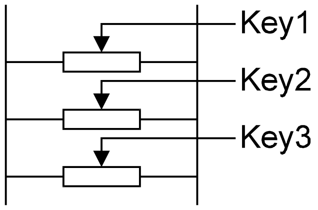
- Forgetting about the parity bits is so common that the NSA probably designs its keysearch machines to accomodate this

---

Breaking DES

DES was designed for efficiency in early-70’s hardware

Makes it easy to build pipelined brute-force breakers in late-90’s hardware



16 stages, tests 1 key per clock cycle

Breaking DES (ctd)

Can build a DES-breaker using

- Field-programmable gate array (FPGA), software-programmable hardware
- Application-specific IC (ASIC)

100 MHz ASIC = 100M keys per second per chip

Chips = $10 in 5K+ quantities

$50,000 = 500 billion keys/sec

= 20 hours/key (40-bit DES takes 1 second)

Breaking DES (ctd)

$1M = 1 hour per key ($^{1}/_{20}$ sec for 40 bits)

$10M = 6 minutes per key ($^{1}/_{200}$ sec for 40 bits)

(US black budget is ~$25-30 billion)

(distributed.net = ~70 billion keys/sec with 20,000 computers)

EFF (US non-profit organisation) broke DES in 2½ days

Amortised cost over 3 years = 8 cents per key

- If your secret is worth more than 8 cents, don’t encrypt it with DES

September 1998: German court rules DES “out of date and unsafe” for financial applications

Brute-force Encryption Breaking

Type of Attacker	Budget	Tool	Time and cost per key recovered		Keylen (bits for security	
			40 bits	56 bits	1995	2015
Pedestrian hacker	Tiny	PC	1 week	Infeasible	45	59
	$400	FPGA	5 hours $0.08	38 years $5,000	50	64
Small business	$10K	FPGA	12 mins $0.08	556 days $5,000	55	69
Corporate department	$300K	FPGA	24 secs $0.08	19 days $5,000	60	74
		ASIC	0.18 secs $0.001	3 hours $38		
Big company	$10M	FPGA	0.7 secs $0.08	13 hours $5,000	70	84
		ASIC	0.005 s $0.001	6 mins $38		
Intelligence agency	$300M	ASIC	0.0002 s $0.001	12 secs $38	75	89

Other Block Ciphers

Triple DES (3DES)

- Encrypt + decrypt + encrypt with 2 (112 bits) or 3 (168 bits) DES keys
- By late 1998, banking auditors were requiring the use of 3DES rather than DES

RC2

- Companion to RC4, 1024 bit key
- RSADSI trade secret, reverse-engineered and posted to the net in 1996
- RC2 and RC4 have special status for US exportability

Other Block Ciphers (ctd)

IDEA

- Developed as PES (proposed encryption standard), adapted to resist differential cryptanalysis as IPES, then IDEA
- Gained popularity via PGP, 128 bit key
- Patented

Blowfish

- Optimised for high-speed execution on 32-bit processors
- 448 bit key, relatively slow key setup

CAST-128

- Used in PGP 5.x, 128 bit key

Other Block Ciphers

Skipjack

- Classified algorithm originally designed for Clipper, declassified in 1998
- 32 rounds, breakable with 31 rounds
- 80 bit key, inadequate for long-term security

GOST

- GOST 28147, Russian answer to DES
- 32 rounds, 256 bit key
- Incompletely specified

Other Block Ciphers

AES

- Advanced Encryption Standard, replacment for DES
- 128 bit block size, 128/192/256 bit key
- Will take several years to be finalised

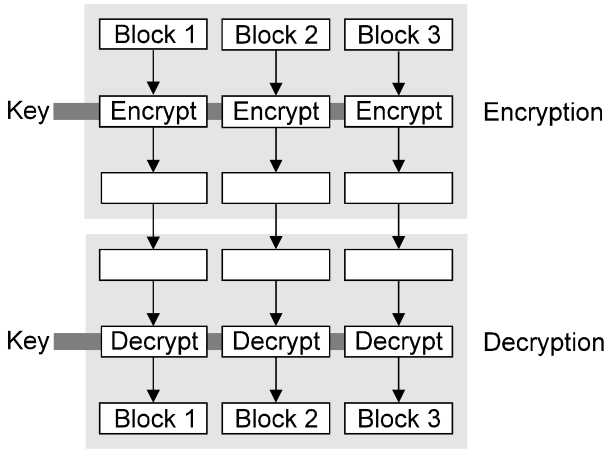
Many, many others

- No good reason not to use one of the above, proven algorithms

---

Using Block Ciphers

ECB, Electronic Codebook



Each block encrypted independently

Using Block Ciphers (ctd)

Original text

`Deposit  $10,000  in acct.  number  12-3456- 789012-3`

Intercepted encrypted form

`H2nx/GHE KgvldSbq GQHbrUt5 tYf6K7ug S4CrMTvH 7eMPZcE2`

Second intercepted message

`H2nx/GHE KgvldSbq GQHbrUt5 tYf6K7ug Pts21LGb a8oaNWpj`

Cut and paste blocks with account information

`H2nx/GHE KgvldSbq GQHbrUt5 tYf6K7ug S4CrMTvH a8oaNWpj`

Decrypted message will contain the attackers account — without them knowing the encryption key

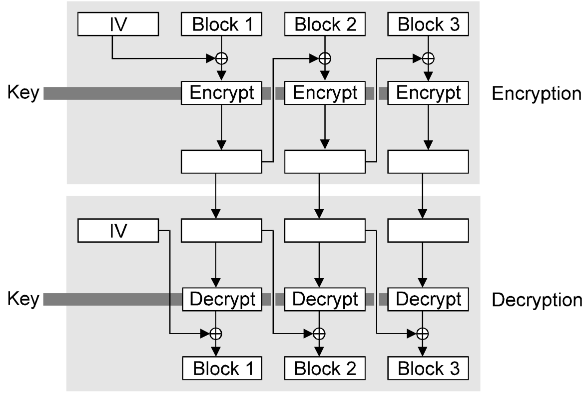
---

Using Block Ciphers (ctd)

Need to

- Chain one block to the next to avoid cut & paste attacks
- Randomise the initial block to disguise repeated messages

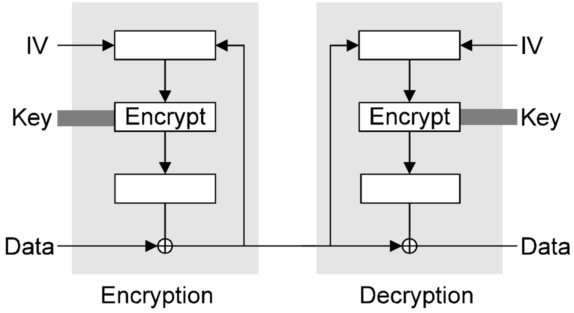
CBC (cipher block chaining) provides chaining, random initialisation vector (IV) provides randomisation



Using Block Ciphers (ctd)

Both ECB and CBC operate on entire blocks

CFB (ciphertext feedback) operates on bytes or bits



This converts a block cipher to a stream cipher (with the accompanying vulnerabilities)

Relative Performance

Fast

RC4
Blowfish, CAST-128, AES
Skipjack
DES, IDEA, RC2
3DES, GOST

Slow

Typical speeds

- RC4 = Tens of MB/second
- 3DES = MB/second

Recommendations

- For performance, use Blowfish
- For job security, use 3DES

Public Key Encryption

How can you use two different keys?

- One is the inverse of the other:
  key1 = 3, key2 = 1/3, message M = 4
  Encryption: Ciphertext $C = M \times key1$
  $= 4 \times 3$
  $= 12$
  Decryption: Plaintext $M = C \times key2$
  $= 12 \times 1/3$
  $= 4$

One key is published, one is kept private → public-key cryptography, PKC

Example: RSA

n, e = public key, n = product of two primes p and q

d = private key

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

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

p, q = 5, 7

$n = p \times q$
$= 35$

e = 3

$d = e^{-1} \bmod ((p-1)(q-1))$
$= 16$

Example: RSA (ctd)

Message M = 4

Encryption: $C = 4^3 \bmod 35$

$= 29$

Decryption: $M = 29^{16} \bmod 35$

$= \sim 2.5 \times 10^{23} \bmod 35$

$= 4$

(Use mathematical tricks otherwise the numbers get dangerous)

Public-key Algorithms

RSA (Rivest-Shamir-Adleman), 1977

- Digital signatures and encryption in one algorithm
- Private key = sign and decrypt
- Public key = signature check and encrypt
- Patented, expires September 2000

DH (Diffie-Hellman), 1976

- Key exchange algorithm

Elgamal

- DH variant, one algorithm for encryption, one for signatures
- Non-patented alternative to RSA

Public-key Algorithms (ctd)

DSA (Digital Signature Algorithm)

- Elgamal signature variant, designed by the NSA as the US government digital signature standard
- Intended for signatures only, but can be adapted for encryption

All have roughly the same strength:

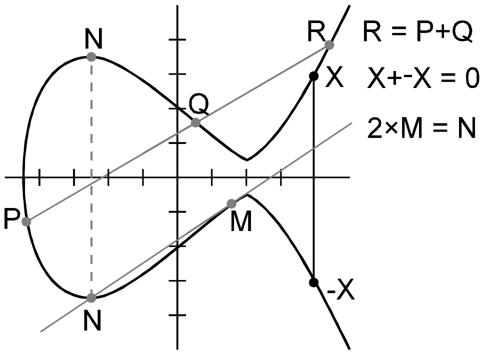
512 bit key is marginal
1024 bit key is recommended minimum size
2048 bit key is better for long-term security

Recommendation

- Anything suitable will do, RSA has wide acceptance but has patent problems in the US

Elliptic Curve Algorithms

Use mathematical trickery to speed up public-key operations



Elliptic Curve Algorithms (ctd)

Now we can add, subtract, etc. So what?

- Calling it “addition” is arbitrary, we can just as easily call it multiplication
- We can now move (some) conventional PKC’s over to EC PKC's (DSA → ECDSA)

Now we have a funny way to do PKC’s. So what?

- Breaking PKC’s over elliptic curve groups is much harder than beaking conventional PKC’s
- We can use much shorter keys
- Encryption/decryption is faster since keys are shorter
- Key sizes are much smaller

Advantages/Disadvantages of ECC’s

Advantages

- Useful for smart cards because of their low resource requirements
- Useful where high-speed operation is required

Disadvantages

- New, details are still being resolved
- Many techniques are still too new to trust
- ECC’s are a minefield of patents, pending patents, and submarine patents

Recommendation: Don’t use them unless you really need their special features

Key Sizes and Algorithms

Conventional vs public-key vs ECC key sizes

Conventional	Public-key	ECC
(40 bits)	—	—
56 bits	(400 bits)	—
64 bits	512 bits	—
80 bits	768 bits	—
90 bits	1024 bits	160 bits
112 bits	1792 bits	195 bits
120 bits	2048 bits	210 bits
128 bits	2304 bits	256 bits

(Your mileage may vary)

Key Sizes and Algorithms (ctd)

However

- Conventional key is used once per message
- Public key is used for hundreds or thousands of messages

A public key compromise is much more serious than a conventional key compromise

- Compromised logon password, attacker can
  - Delete your files
- Compromised private key, attacker can
  - Drain credit card
  - Clean out bank account
  - Sign contracts/documents
  - Identity theft

Key Sizes and Algorithms (ctd)

512 bit public key vs 40 bit conventional key is a good balance for weak security

Recommendations for public keys:

- Use 512-bit keys only for micropayments/smart cards
- Use 1K bit key for short-term use (1 year expiry)
- Use 1.5K bit key for longer-term use
- Use 2K bit key for certification authorities (keys become more valuable further up the hierarchy), long-term contract signing, long-term secrets

The same holds for equivalent-level conventional and ECC keys

Hash Algorithms

Reduce variable-length input to fixed-length (128 or 160 bit) output

Requirements

- Can’t deduce input from output
- Can’t generate a given output (CRC fails this requirement)
- Can’t find two inputs which produce the same output (CRC also fails this requirement)

Used to

- Produce fixed-length fingerprint of arbitrary-length data
- Produce data checksums to enable detection of modifications
- Distill passwords down to fixed-length encryption keys

Also called message digests or fingerprints

MAC Algorithms

Hash algorithm + key to make hash value dependant on the key

Most common form is HMAC (hash MAC)

hash(key, hash(key, data))

- Key affects both start and end of hashing process

Naming: hash + key = HMAC-hash

MD5 → HMAC-MD5

Algorithms

MD2: 128-bit output, deprecated

MD4: 128-bit output, broken

MD5: 128-bit output, weaknesses

SHA-1: 160-bit output, NSA-designed US government secure hash algorithm, companion to DSA

RIPEMD-160: 160-bit output

HMAC-MD5: MD5 turned into a MAC

HMAC-SHA: SHA-1 turned into a MAC

Recommendation: Use SHA-1, HMAC-SHA