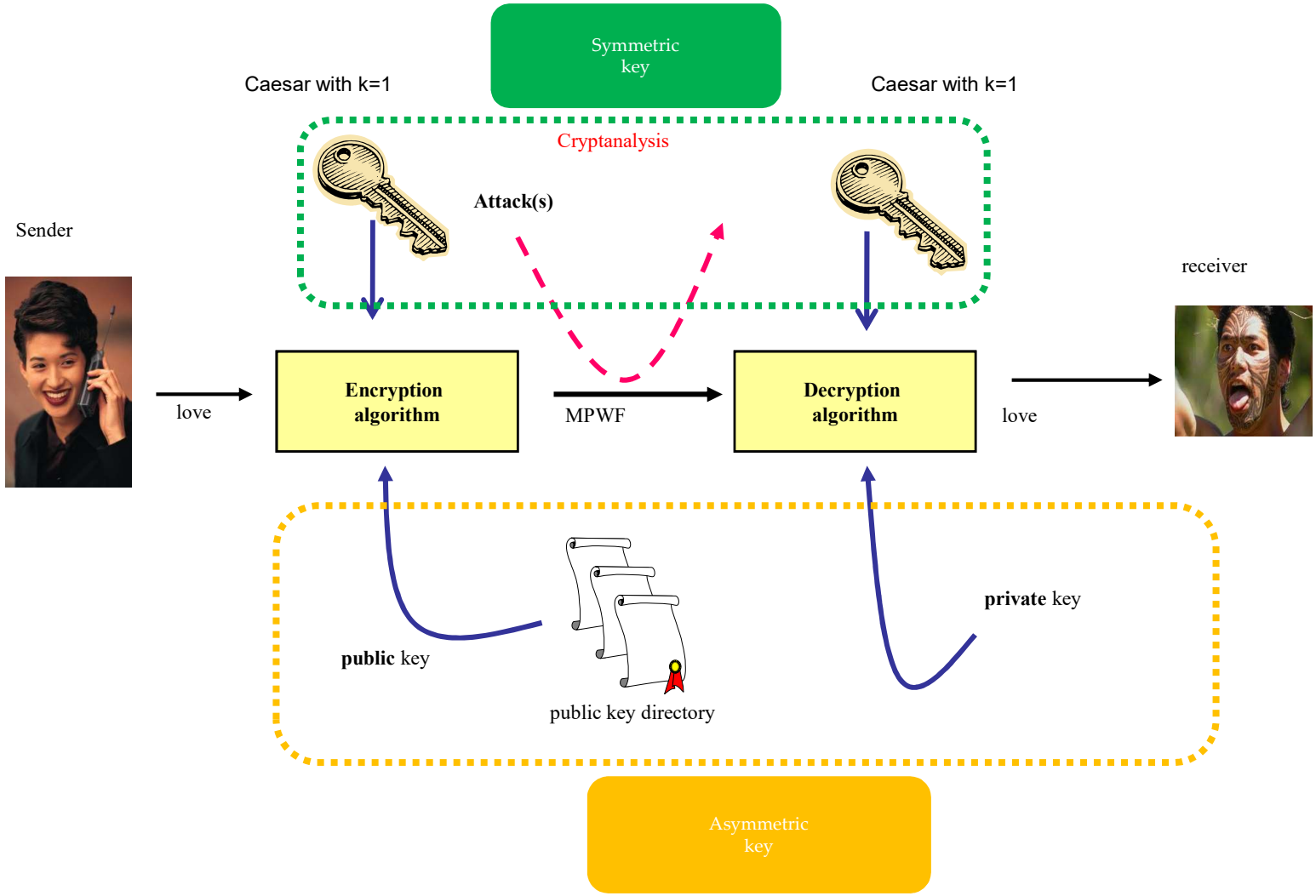
COSC 362

Chapter 2. Cryptographic Tools ("Lightweight coverage")

Overview


Symmetric key
Caesar with k=1
Caesar with k=1
Cryptanalysis
Attack(s)
Sender
receiver
love
Encryption algorithm
MPWF
Decryption algorithm
love
public key
public key directory
private key
Asymmetric key

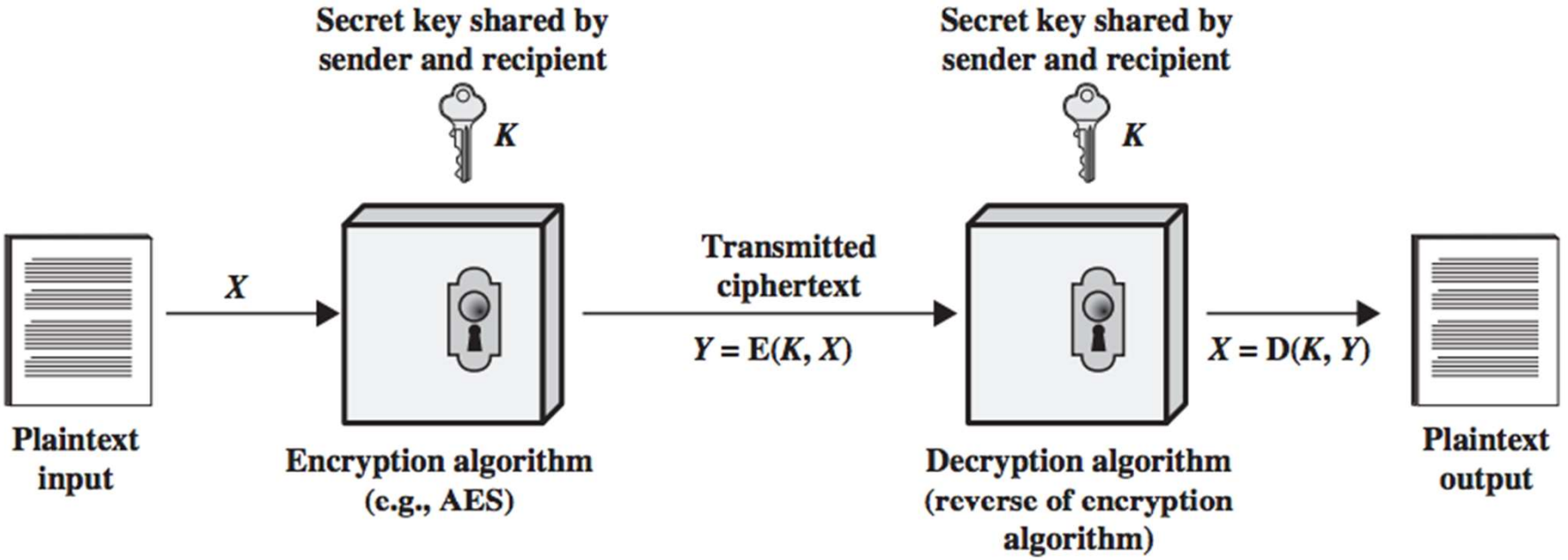
Basic Terminology

- Plaintext (**P**):
  - the original message, e.g., love
- Ciphertext (**C**):
  - the coded (or encrypted) message, e.g., MPWF
- Cipher (Encryption/Decryption algorithm):
  - algorithm for transforming from plaintext (ciphertext) to ciphertext (plaintext), e.g., Caesar, DES, AES.
- Key(s):
  - info. used in cipher known only to sender/receiver in symmetric cipher, e.g., k=2, (or 64bits, 128bits long size)
- Encrypt (encipher): **E(P)**
  - converting plaintext to ciphertext; e.g., love -> MPWF
- Decrypt (decipher): **D(C)**
  - recovering ciphertext from plaintext; e.g., MPWF->love

Symmetric vs. Asymmetric

	Symmetric	Asymmetric
Key relation	Enc. Key = Dec. key	Enc. Key ≠ Dec. key
Encryption Key	**Secret**	**Public, {Private}**
Decryption Key	**Secret**	**Private, {Public}**
Algorithm	**Classified/Open**	**Open**
Example	**DES (56 bits), AES**	**RSA (1024 bits)**
Key Distribution	**Required**	**Not required**
Number of key	**Many (Mbits/second)**	**Small (eg., kbits/second)**
Performance	**Fast**	**slow**

Simplified Model of Symmetric Encryption

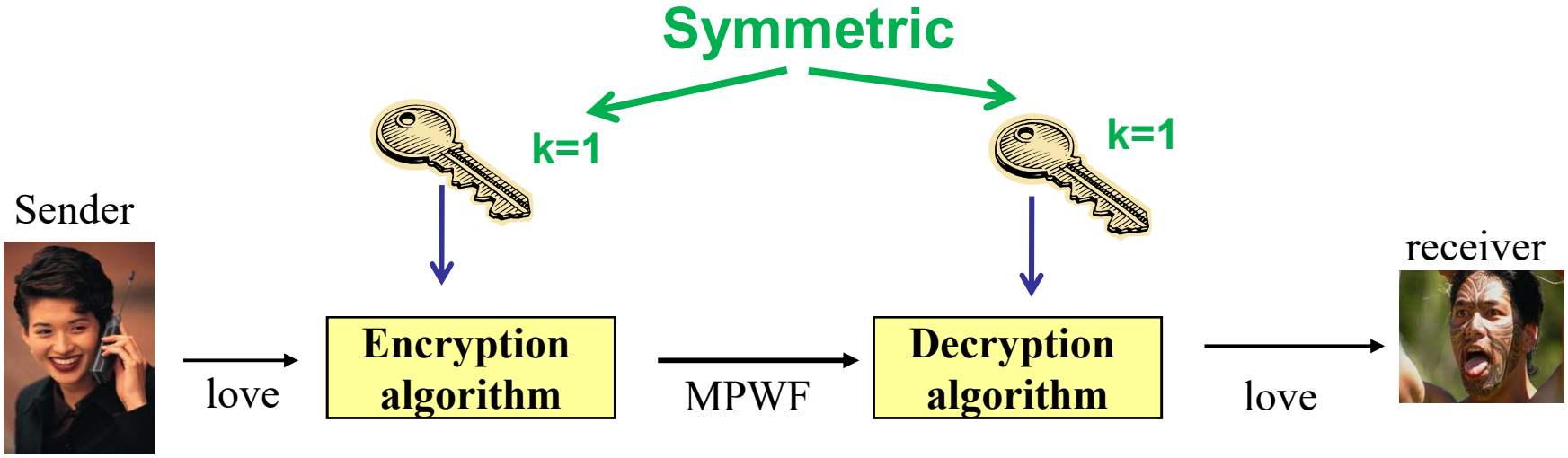


*Stallings and Brown, Ch 2, Fig 2.1

Classical (Historical) ciphers

- Substitution ciphers
  - Monoalphabetic
  - Polyalphabetic
- Transposition ciphers

Symmetric key encryption
using Caesar Cipher



Q: If key (k) is 5? What will be the ciphertext of the plantext “love”?

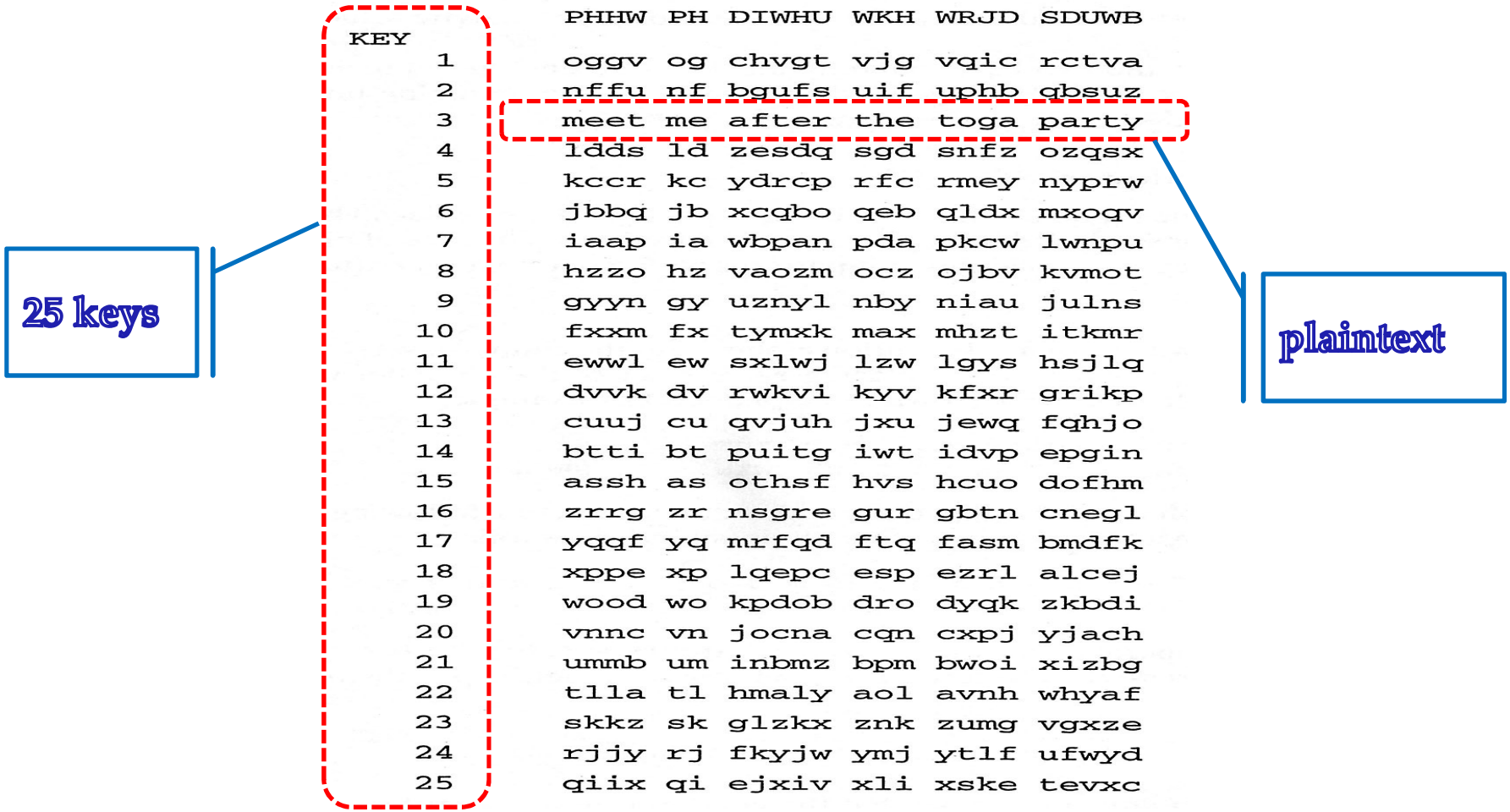
A: QTAJ

$$C \equiv E(P) \equiv P+k \bmod 26$$

a **modulo** *n* (abbreviated as *a* mod *n*)

Caesar cipher (cont.)

- Why Caesar cipher is too week?
  - Possible key size is? 25



KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

Classical (Historical) ciphers

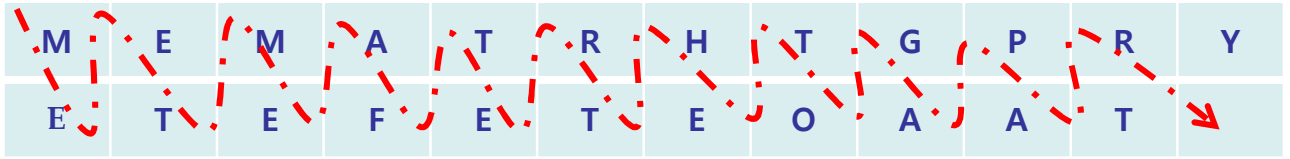
- Substitution ciphers
  - Monoalphabetic
  - Polyalphabetic
- Transposition ciphers

Transposition (**permutation**) Cipher

- these hide the message by rearranging the letter order
- without altering the actual letters used
- can recognize these since have the same frequency distribution as the original text

Transposition Cipher

- Rail-fence cipher (a.k.a., a zigzag cipher)
  - write message letters out diagonally over a number of rows (with the depth 2 (with a key of 2))
  - then read off cipher row by row
  - Plaintext: meetmeafterthetogaparty



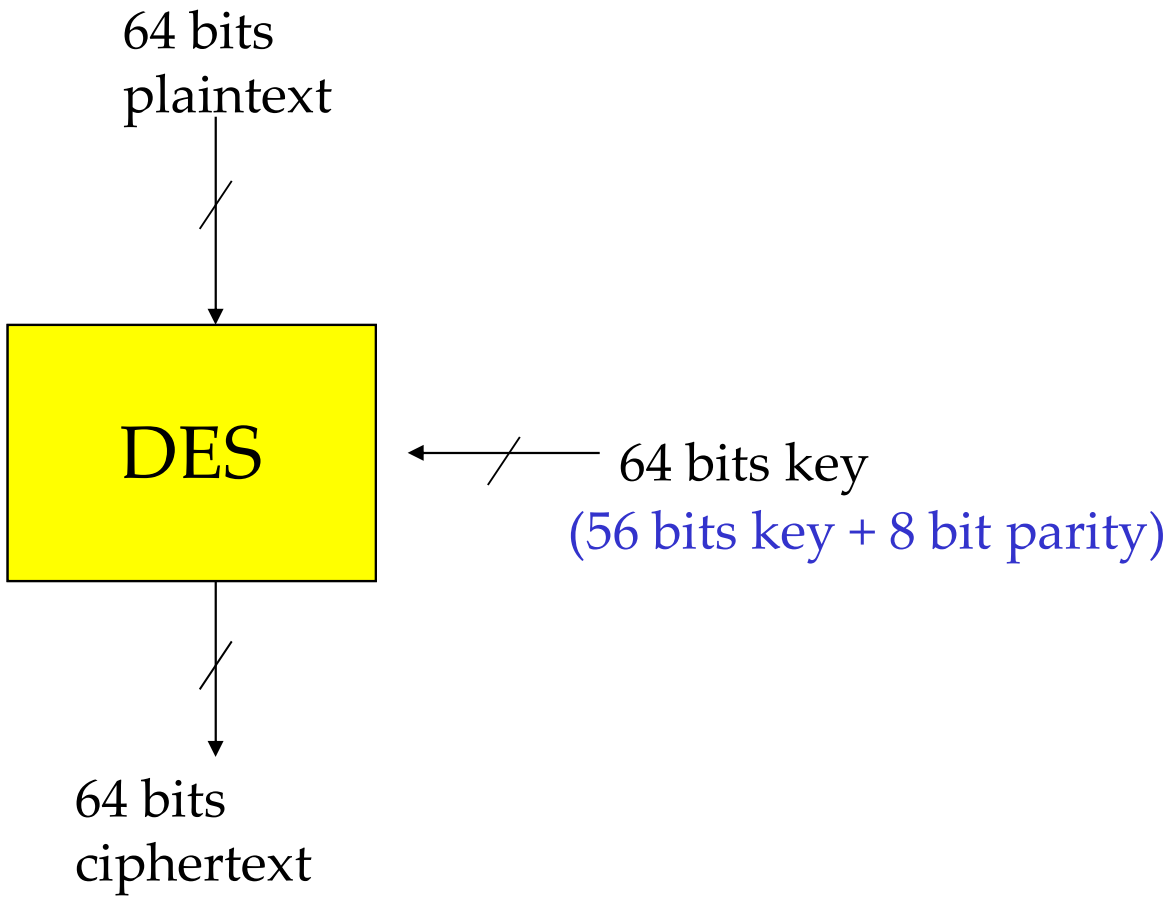
  - ciphertext: MEMATRHTGPRYETEFETEOAAT

http://rumkin.com/tools/cipher/railfence.php

Now think about …

- Q: Why does the ciphers introduced so far not secure?
- A: because of language characteristics / weak key size. Brute-force attacks are so easy and effective
- Q: Any ideas to improve them (you already know the answer)?
- A: Use both substitution and transposition *together*. Brute-force attacks are still so easy and effective if encryption strategies can be guessed

Data Encryption Standard (DES): Structure



- Standard: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
- Simplified DES theory: http://www.cs.uri.edu/cryptography/dessimplified.htm
- example: http://homepage.smc.edu/morgan_david/vpn/assignments/assgt-sdes-encrypt-sample.htm

Data Encryption Standard

From Wikipedia, the free encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export

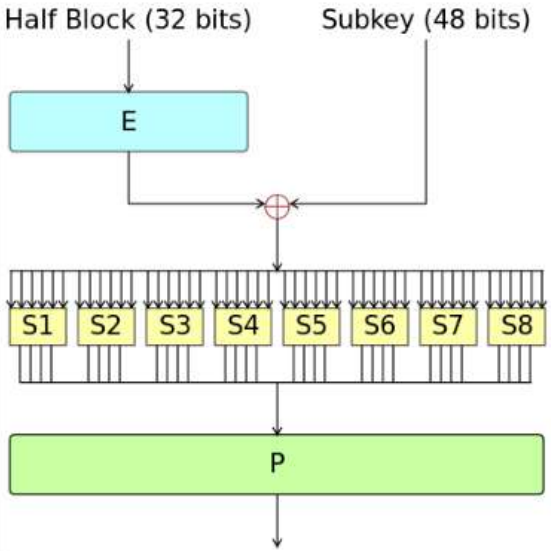
Create a book
Download as PDF
Printable version

The **Data Encryption Standard** (**DES**, /ˌdiːˌiːˈɛs, dɛz/) is a symmetric-key algorithm for the encryption of electronic data. Although now considered insecure, it was highly influential in the advancement of modern cryptography.

Developed in the early 1970s at IBM and based on an earlier design by Horst Feistel, the algorithm was submitted to the National Bureau of Standards (NBS) following the agency's invitation to propose a candidate for the protection of sensitive, unclassified electronic government data. In 1976, after consultation with the National Security Agency (NSA), the NBS eventually selected a slightly modified version (strengthened against differential cryptanalysis, but weakened against brute-force attacks), which was published as an official Federal Information Processing Standard (FIPS) for the United States in 1977.

The publication of an NSA-approved

Data Encryption Standard



The Feistel function (F function) of DES

General

Designers	IBM
First published	1975 (Federal Register) (standardized in January 1977)
Derived from	Lucifer
Successors	Triple DES, G-DES, DES-X, LOKI89, ICE

Cipher detail

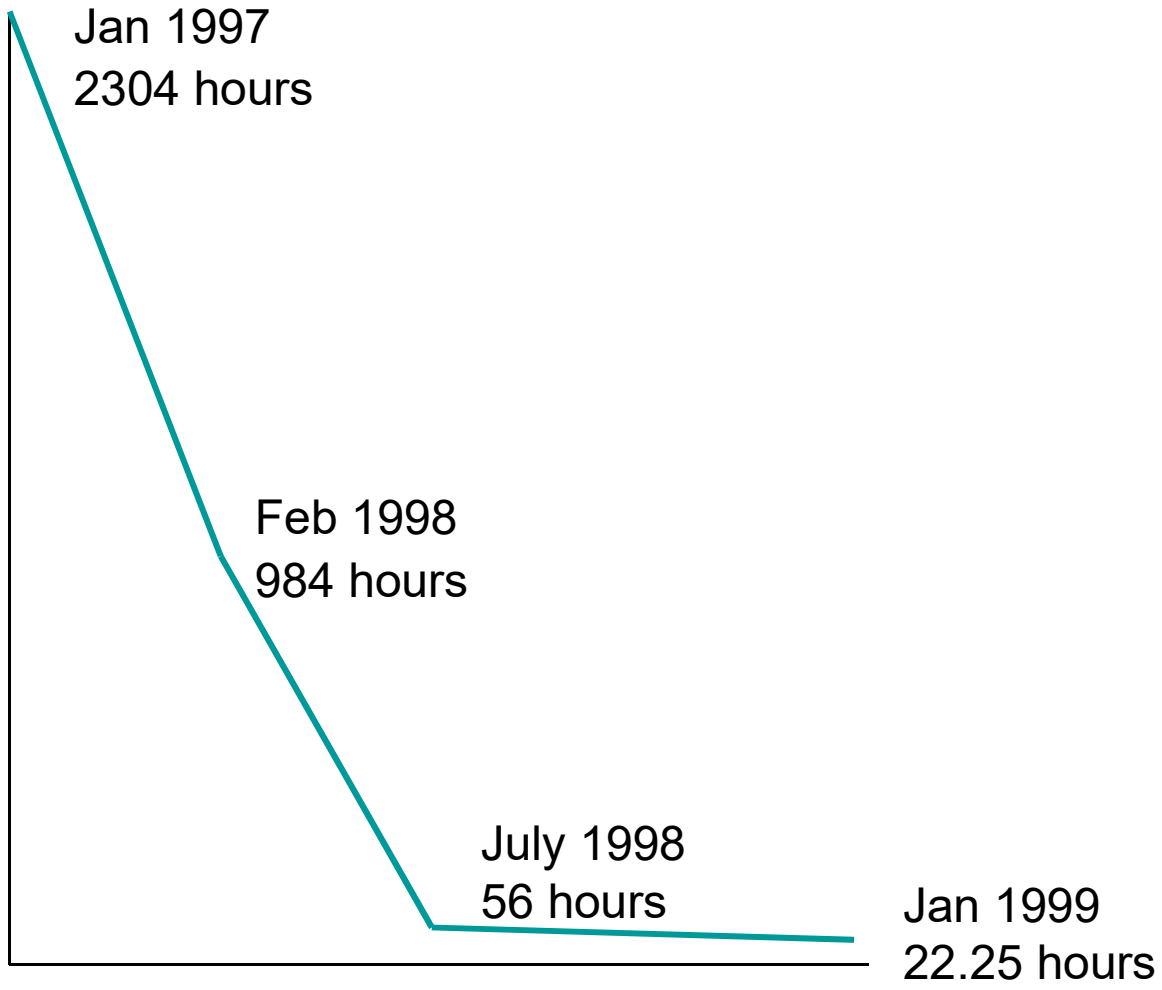
Key sizes	56 bits (+8 parity bits)
Block sizes	64 bits

DES: security concern

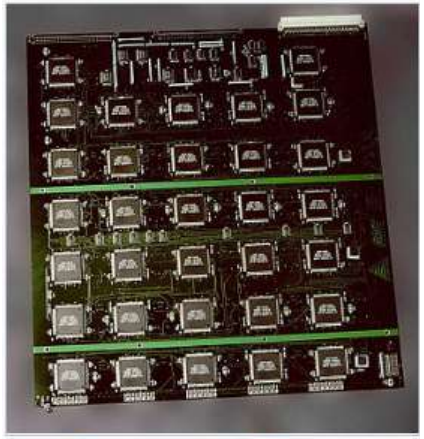
- 56 bit key is too short
  - There are 2^56 (≈7.21*10^16) possible keys
  - Moore's law: speed of processor doubles per 1.5 year
  - 1997: 3500 machines broke DES in about 4 months
  - 1998: 1M dollar machine broke DES in about 4 days
  - …
- What if cloud computing, massively parallel computing technologies are used?

DES: security concern (cont.)

- Cracking the 56-bit DES Encryption Algorithm



Brute-force Attacks on DES



The EFF's US$250,000 DES cracking machine contained 1,856 custom chips and could brute-force a DES key in a matter of days—the photo shows a DES Cracker circuit board fitted with several Deep Crack chips.

The next confirmed DES cracker was the COPACOBANA machine built in 2006 by teams of the Universities of Bochum and Kiel, both in Germany. Unlike the EFF machine, COPACOBANA consists of commercially available, reconfigurable integrated circuits. 120 of these field-programmable gate arrays (FPGAs) of type XILINX Spartan-3 1000 run in parallel. They are grouped in 20 DIMM modules, each containing 6 FPGAs. The use of reconfigurable hardware makes the machine applicable to other code breaking tasks as well.[26] One of the more interesting aspects of COPACOBANA is its cost factor. One machine can be built for approximately $10,000.[27] The cost decrease by roughly a factor of 25 over the EFF machine is an example of the continuous improvement of digital hardware—see Moore's law. Adjusting for inflation over 8 years yields an even higher improvement of about 30x. Since 2007, SciEngines GmbH, a spin-off company of the two project partners of COPACOBANA has enhanced and developed successors of COPACOBANA. In 2008 their COPACOBANA RIVYERA reduced the time to break DES to less than one day, using 128 Spartan-3 5000's. SciEngines RIVYERA held the record in brute-force breaking DES, having utilized 128 Spartan-3 5000 FPGAs.[28] Their 256 Spartan-6 LX150 model has further lowered this time.

In 2012, David Hutton and Moxie Marlinspike announced a system with 48 Xilinx Virtex-6 LX240T FPGAs, each FPGA containing 40 fully pipelined DES cores running at 400MHz, for a total capacity of 768 gigakeys/sec. The system can exhaustively search the entire 56-bit DES key space in about 26 hours and this service is offered for a fee online.[29][30]

- "Faster attacks" (e.g., differential or linear cryptanalysis) have been proposed

Brute Force Search

- always possible to simply try every key
  - e.g., PIN number (0000)
- most basic attack, proportional to key size
- assume either know / recognise plaintext

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs	Time required at 10^6 decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31} μs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} μs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} μs= 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} μs= 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} μs= 6.4×10^{12} years	6.4×10^6 years

Multiple Encryption & DES

- DES is not secure enough.
- The once large key space, 2^{56}, is now too small.
- But users in commerce and finance are not ready to give up on DES.
- Solution: to use multiple DES with multiple keys

Q: how many times can we use?

A: 2, 3, …

DES, Triple DES

- DES is now considered to be insecure for many applications.
  - DES has been withdrawn as a standard by the NIST
- The algorithm is believed to be practically secure in the form of triple DES
  - Attacking cost is lot more than triple with triple DES

Triple DES (3DES or TDES)

The standards define three keying options:

- Keying option 1:
  - All three keys are independent.
  - is the strongest, with 3 × 56 = 168 bits
  - e.g., PGP (Pretty Good Privacy) and S/MIME (Secure/Multipurpose Internet Mail Extensions).
- Keying option 2:
  - K_1 and K_2 are independent, and $K_3 = K_1$.
  - provides less security, with 2 × 56 = 112 key bits.
- Keying option 3:
  - All three keys are identical, i.e. $K_1 = K_2 = K_3$.
  - is equivalent to DES, with only 56 key bits.
  - provides **backward compatibility** with DES

Advanced Encryption Standard (AES)

- DES cracked, Triple-DES slow: what next?
- 1997 NIST called for algorithms
- Final five
  - Rijndael (Two Belgians: Joan Daemen, Vincent Rijmen)
  - Serpent (Ross Anderson)
  - Twofish (Bruce Schneier)
  - RC6 (Don Rivest, Lisa Yin)
  - MARS (Don Coppersmith, IBM)
- 2000 Rijndael won; 2002 Rijndael became AES
- An iterative rather than Feistel cipher
  - operates on entire data block in every round
- Byte operations: easy to implement in software

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

Overview of AES

- Rijndael allows many block sizes and key sizes
  - AES restricts block length to 128 bit
  - The key size can be independently specified to 128, 192 or 256 bits (AES-128, AES-192, AES-256)

Key size (words/bytes/bits)	4/16/**128**	6/24/**192**	8/32/**256**
Number of rounds	**10**	**12**	**14**
Expanded key size (words/byte)	44/176	52/208	60/240

More on AES

AES has been adopted by the U.S. government and is now used worldwide. It supersedes the Data Encryption Standard (DES),[10] which was published in 1977. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

In the United States, AES was announced by the NIST as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001.[7] This announcement followed a five-year standardization process in which fifteen competing designs were presented and evaluated, before the Rijndael cipher was selected as the most suitable (see Advanced Encryption Standard process for more details).

AES became effective as a federal government standard on May 26, 2002, after approval by the Secretary of Commerce. AES is included in the ISO/IEC 18033-3 standard. AES is available in many different encryption packages, and is the first (and only) publicly accessible cipher approved by the National Security Agency (NSA) for top secret information when used in an NSA approved cryptographic module (see Security of AES, below).

https://thebestvpn.com/advanced-encryption-standard-aes/

It should be noted that AES is free for any public, private, commercial, or non-commercial use. (Although you should proceed with caution when implementing AES in software since the algorithm was designed on a big-endian system and the majority of personal computers run on little-endian systems.)

1. **Archive and Compression Tools**

If any of you have ever downloaded a file off the internet and then gone to open that file only to notice that the file was compressed, (meaning that the original file size was reduced to minimize its affect on your hard drive) then you have likely installed software that relies on an AES encryption.

Common compression tools like WinZip, 7 Zip, and RAR allow you to compress and then decompress files in order to optimize storage space, and nearly all of them use AES to ensure file security.

2. **Disk/Partition Encryption**

If you're already familiar with the concept of cryptography and have taken extra measures to ensure the security of your personal data, the disk/partition encryption software that you use likely uses an AES algorithm.

BitLocker, FileVault, and CipherShed are all encryption software that run on AES to keep your information private.

3. **VPNs**

The AES algorithm is also commonly applied to VPNs, or Virtual Private Networks.

For those of you who are unfamiliar with the term, a VPN is a tool that allows you to use a public internet connection in order to connect to a more secure network.

Public-Key (Asymmetric) Encryption

PKC At A Glance

Article Talk Read Edit View

Public-key cryptography

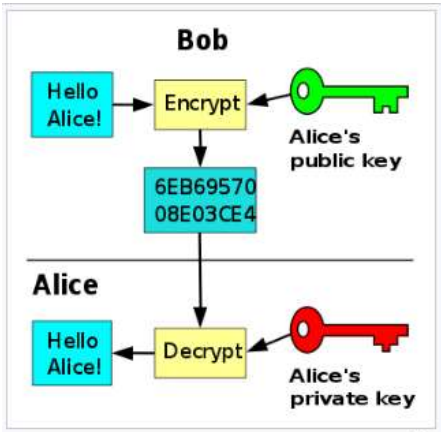
From Wikipedia, the free encyclopedia

This article's **lead section may not adequately s** comply with Wikipedia's lead section guidelines, the lead to provide an accessible overview of the way that it can stand on its own as a concise ver *(December 2015)*

Public key cryptography, or **asymmetrical cryptography**, is any cryptographic system that uses pairs of keys: *public keys* which may be disseminated widely, and *private keys* which are known only to the owner. This accomplishes two functions: authentication, where the public key verifies a holder of the paired private key sent the message, and encryption, where only the paired private key holder can decrypt the message encrypted with the public key.

In a public key encryption system, any person can encrypt a message using the receiver's public key. That encrypted message can only be decrypted with the receiver's private key. To be practical, the generation of a public and private key -pair must be computationally economical. The strength of a public key cryptography system relies on the computational effort (*work factor* in cryptography) required to find the private key from its paired public key. If so, effective security only requires keeping the private key private; the public key can be enly distributed without compromising security.[1]

An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.



In an asymmetric key encryption scheme, anyone can encrypt messages using the public key, but only the holder of the paired private key can decrypt. Security depends on the secrecy of the private key.

CRYPTOGRAPHY PIONEERS RECEIVE ACM A.M. TURING AWARD

Diffie and Hellman's Invention of Public-Key Cryptography and Digital Signatures Revolutionized Computer Security

ACM, the Association for Computing Machinery, today named Whitfield Diffie, former Chief Security Officer of Sun Microsystems and Martin E. Hellman, Professor Emeritus of Electrical Engineering at Stanford University, recipients of the 2015 ACM A.M. Turing Award for critical contributions to modern cryptography. The ability for two parties to use encryption to communicate privately over an otherwise insecure channel is fundamental for billions of people around the world. On a daily basis, individuals establish secure online connections with banks, e-commerce sites, email servers and the cloud. Diffie and Hellman's groundbreaking 1976 paper, "New Directions in Cryptography," introduced the ideas of public-key cryptography and digital signatures, which are the foundation for most regularly-used security protocols on the Internet today. The Diffie-Hellman Protocol protects daily Internet communications and trillions of dollars in financial transactions.

The ACM Turing Award, often referred to as the "Nobel Prize of Computing," carries a $1 million prize with financial support provided by Google, Inc. It is named for Alan M. Turing, the British mathematician who articulated the mathematical foundation and limits of computing and who was a key contributor to the Allied cryptoanalysis of the German Enigma cipher during World War II.

"Today, the subject of encryption dominates the media, is viewed as a matter of national security, impacts government-private sector relations, and attracts billions of dollars in research and development," said ACM President Alexander L. Wolf. "In 1976, Diffie and Hellman imagined a future where people would regularly communicate through electronic networks and be vulnerable to having their communications stolen or altered. Now, after nearly 40 years, we see that their forecasts were remarkably prescient."

"Public-key cryptography is fundamental for our industry," said Andrei Broder, Google Distinguished Scientist. "The ability to protect private data rests on protocols for confirming an owner's identity and for ensuring the integrity and confidentiality of communications. These widely used protocols were made possible through the ideas and methods pioneered by Diffie and Hellman."

Whitfield Diffie is a former Vice President and Chief Security Officer of Sun Microsystems, where he became a Sun Fellow. As Chief Security Officer,

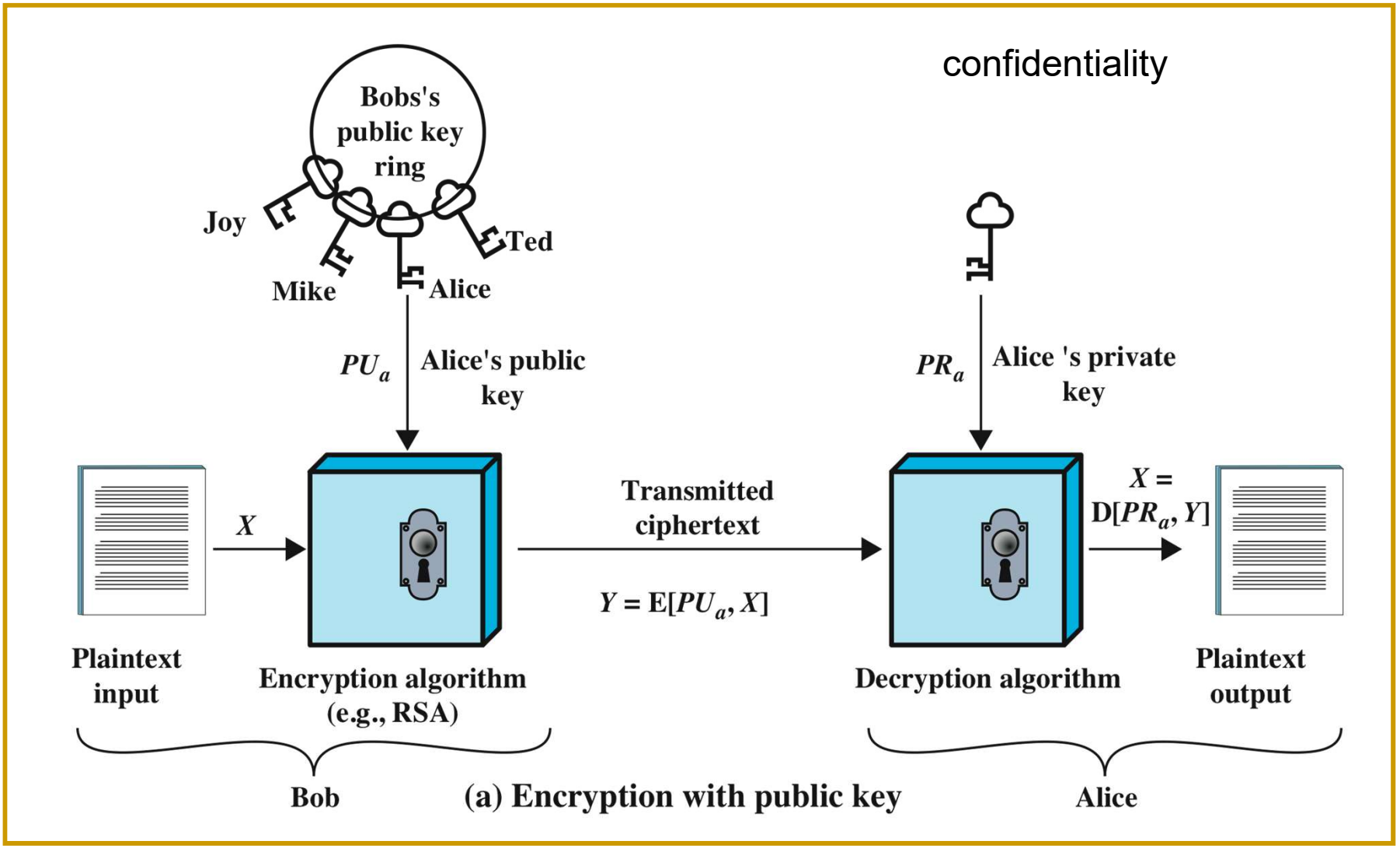
Martin E. Hellman is Professor Emeritus of Electrical Engineering at Stanford University, where he was

Requirements for Public-Key Cryptography

- It is computationally easy for a party to generate a pair of public and private key
- It is computationally easy for a sender A, knowing the public key and the plaintext message, to generate the corresponding ciphertext
- It is computationally easy for the receiver to decrypt the resulting ciphertext using the private key to recover the plaintext message
- It is computationally infeasible for an opponent, knowing the public key, to determine the private key
- It is computationally infeasible for an opponent, knowing the public key and a ciphertext, to recover the plaintext message
- Either of the two related keys can be used for encryption, with the other used for decryption
  - useful, but not necessary for all public-key applications

Symmetric vs. Asymmetric

	Symmetric	Asymmetric
Key relation	Enc. Key = Dec. key	Enc. Key ≠ Dec. key
Encryption Key	**Secret**	**Public, {Private}**
Decryption Key	**Secret**	**Private, {Public}**
Algorithm	**Classified/Open**	**Open**
Example	**DES (56 bits), AES**	**RSA (1024 bits)**
Key Distribution	**Required**	**Not required**
Number of key	**Many** (Mbits/second)	**Small** (eg., kbits/second)
Performance	**Fast**	**slow**



* Stallings and Brown, Ch 2, Fig 2.7

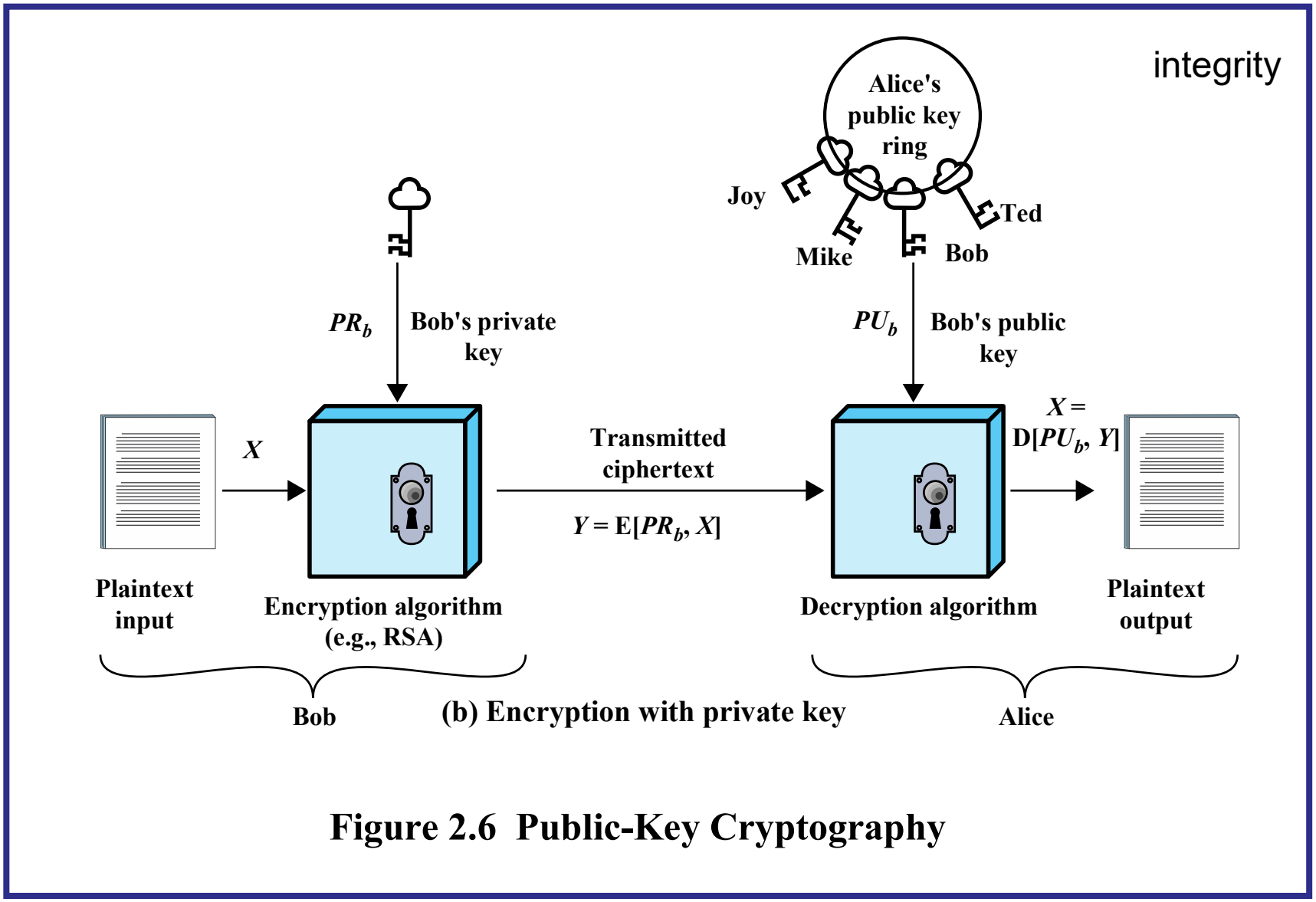
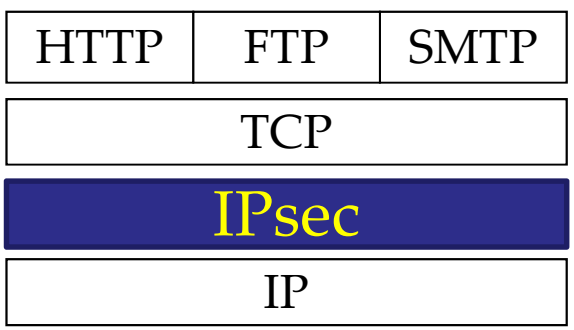


Figure 2.6 Public-Key Cryptography

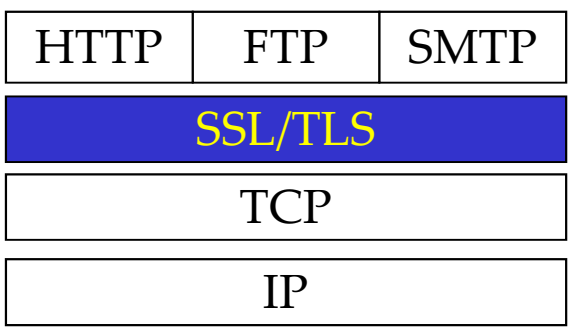
* Stallings and Brown, Ch 2, Fig 2.7

DH Applications

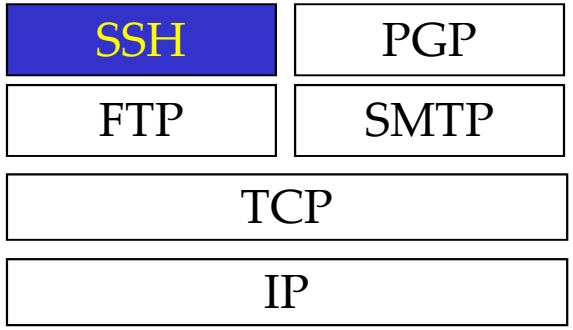
- DH is currently used in many protocols, namely:
  - Internet Protocol Security (IPSec)
    - Internet Key Exchange (IKE)
  - **Secure Sockets Layer (SSL)**/Transport Layer Security (TLS)
    - Key agreement; in conjunction with DES (40-bit key) or 3-DES (128-bit key)
  - Secure Shell (SSH)
  - Public Key Infrastructure (PKI)



At the network approach



At the transport layer



At the application layer

Source: Riley Lochridge, The Diffie-Hellman Algorithm, 2003

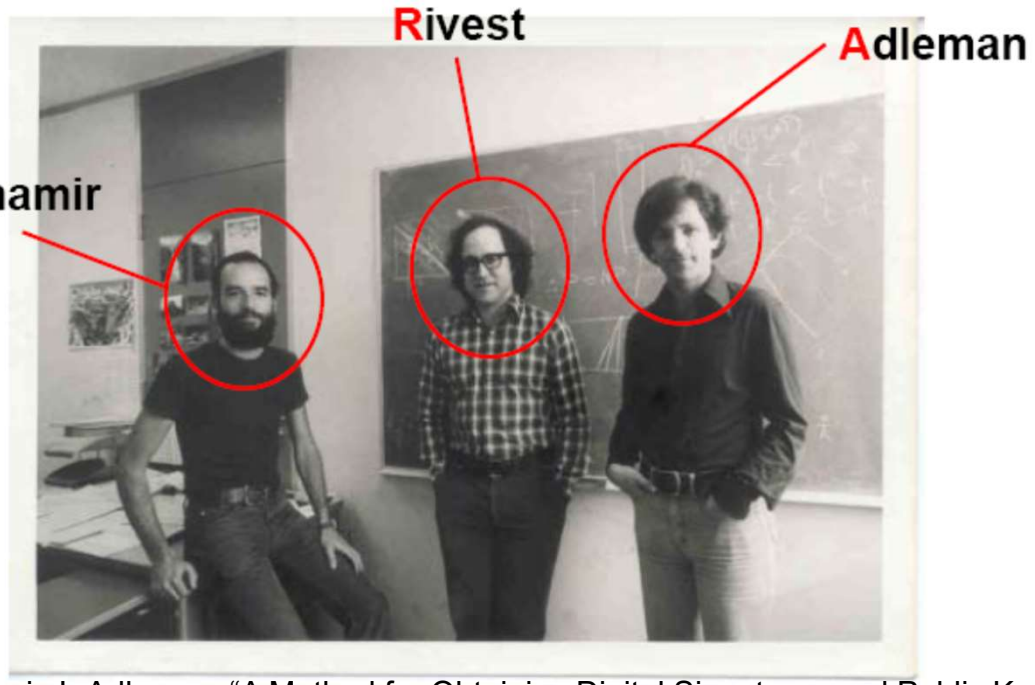
RSA

- 1st public key cryptosystem
- Believed to be secure if IFP (Integer Factorization Problem) is hard and worldwide standard for last 30 years

Cf. DH – key exchange

RSA (Sam Rodgers, Josh Shaw and Alexandre Alves)

RSA (Ron Rivest, Adi Shamir and Leonard Adleman)



R.L.Rivest, A.Shamir, L.Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", CACM,(Communications of the Association for Computing Machinery) Vol.21, No.2, pp.120-126,Feb,1978

RSA

(Ron Rivest, Adi Shamir, Len Adelman)

- RSA is the most widely used asymmetric encryption algorithms (Turing Award 2002)

BIRTH:

1947, Schenectady, New York, USA

EDUCATION:

Niskayuna High School, Niskayuna, New York, USA (1965); BA (Mathematics, Yale University, 1969); Ph.D. (Computer Science, Stanford University, 1973)

EXPERIENCE:

INRIA, Rocquencourt, France (post-doctorate position, 1973-1974); MIT (professor of Electrical Engineering and Computer Department, member of MIT's Computer Science and Artificial Intelligence Laboratory, CSAIL, a member of their Theory of Computation Group and a leader of its Cryptography and Information Security Group, from 1974)

HONORS AND AWARDS:

Member, National Academy of Engineering (1990); Fellow of the ACM (1993); Member, American Academy of Arts and Sciences (1993); National Computer Systems Security Award (1996); ACM Paris Kanellakis Theory and Practice Award (1997); IEEE Koji Kobayashi Computers and

RONALD (RON) LINN RIVEST

United States – 2002

CITATION

Together with Leonard M. Adleman and Adi Shamir, for their ingenious contribution to making public-key cryptography useful in practice.

SHORT ANNOTATED BIBLIOGRAPHY | ACM TURING AWARD LECTURE VIDEO | RESEARCH SUBJECTS | ADDITIONAL MATERIALS

Ron Rivest grew up in Niskayuna, New York, a suburb of Schenectady. He attended public schools and graduated from the Niskayuna High School in 1965. He graduated from Yale University in 1969 with a B.A. in mathematics, and from Stanford University in 1973 with a PhD in Computer Science. He learned from the best: his PhD supervisor was Turing Award recipient Robert Floyd, and he worked closely with Turing Award laureate Don Knuth.

Following graduate study he accepted a post-doctoral position at INRIA in Rocquencourt, France before taking a job at MIT, where he has been ever since. He currently holds the Andrew and Erna Viterbi professorship in the Department of Electrical Engineering and Computer Science.

At MIT he met Leonard M. Adleman and Adi Shamir, who collaborated with Ron on their fundamental advance in cryptography. They were inspired by a 1976 paper [4] by cryptographers Whitfield Diffie and Martin Hellman discussing several new developments in cryptography. It described ways for the sender and receiver of private messages to avoid needing a shared secret key, but it did not provide any realistic way to implement these

KOREA UNIVERSITY 1905

RSA (cryptosystem)

From Wikipedia, the free encyclopedia

For other uses, see RSA (disambiguation).

RSA (**Rivest–Shamir–Adleman**) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem". The acronym RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1978. Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), had developed an equivalent system in 1973, but this was not declassified until 1997.[1]

RSA

General	
Designers	Ron Rivest, Adi Shamir, and Leonard Adleman
First published	1977
Certification	PKCS#1, ANSI X9.31, IEEE 1363
Cipher detail	
Key sizes	1,024 to 4,096 bit typical
Rounds	1
Best public cryptanalysis	
General number field sieve for classical computers Shor's algorithm for quantum computers A 768-bit key has been broken	

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, and if the public key is large enough, only someone with knowledge of the prime numbers can decode the message feasibly.[2] Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem remains an open question.

RSA is a relatively slow algorithm, and because of this, it is less commonly used to directly encrypt user data. More often, RSA passes encrypted shared keys for symmetric key cryptography which in turn can perform bulk encryption-decryption operations at much higher speed.

https://www.quora.com/Where-is-RSA-encryption-used

Quora Home 1 Answer Notifications Search Quora

HTTPS is used with countless websites where the exchange of personal or private information takes place (passwords, eCommerce, etc.)

Computer applications are often digitally signed so that the consumer can verify that the software has not been modified since the publisher released it. If the software is modified after signing (even by one bit) the digital signature will fail. This provides a way to prevent non-trusted code from executing.

Then there are other technologies like commercial satellite radio, satellite TV, etc. If the content publisher simply broadcast the media in the clear, they would have a difficult time convincing people to pay their monthly bill. Instead they need a means to distribute the decryption keys to their paid subscribers, while keeping the general public out -- RSA is a candidate for that too.

RSA is used for many other Digital Rights Management (DRM) applications.

These are just a couple examples.

The security of the RSA Algorithm is based on the belief that factoring large integers is and will continue to be computationally expensive. RSA has been in use for more than 30 years and to this day is still considered secure, provided keys of sufficient length are used. Keys created today are typically at least 2048 bits.

Why is RSA so popular?

- It's not governed by any active patents. Anyone can use it, royalty free in any private or commercial product.
- RSA can do encryption, decryption, signature and signature verification -- all with the same two functions.
- It has been around for more than 30 years and has not been compromised.

How secure is an *n*-bit RSA key?

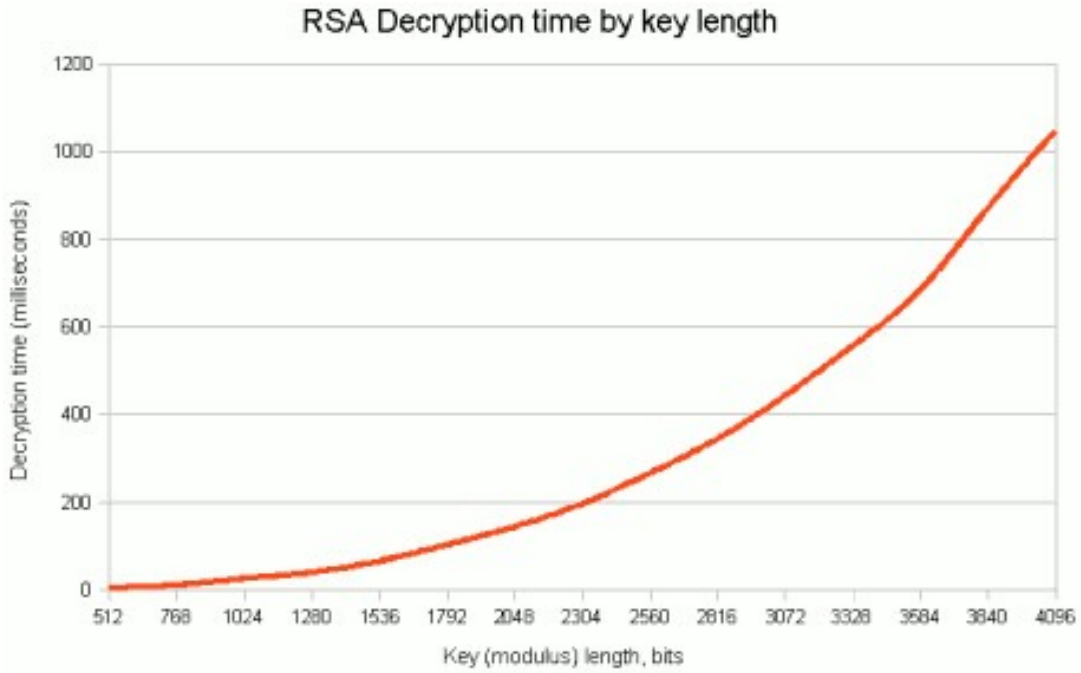
- Shamir & Tromer (2003)
  - suggested that for "a few dozen million US dollars", a hardware device could be built to break a 1024-bit RSA key within around a year.
- Franke et al. (2005)
  - estimated a cost of 200 million dollars for a machine to factorise a 1024-bit number in one year.
- Kaliski (2003)
  - recommended RSA key sizes depending on lifetime of confidential data.

Lifetime of data (from 2003)	RSA key size
Up to 2010	1024 bits
Up to 2030	2048 bits
Up to 2031 onwards	3072 bits

- Recommendations
  - for high-security applications or for data that needs to remain confidential for more than a few years, you should use at least a 2048-bit key
  - to keep data **confidential for more than the next two decades**, RSA recommends a **key size larger than 2048 bits**

RSA key lengths vs performance

- With every doubling of the RSA key length, **decryption is 6-7 times slower**.



The timings were made on a 2GHz Pentium.

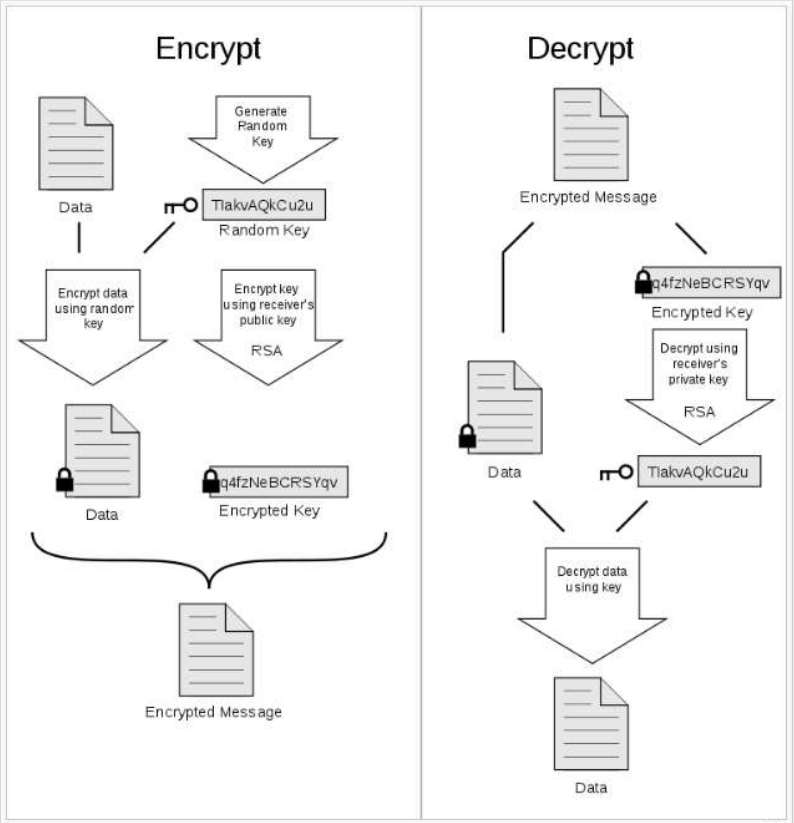
If we use a 4096-bit modulus, it takes around a second of CPU time to decrypt a block of data

Just One More: PGP

(Pretty Good Privacy)

Design [edit]

PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and finally public-key cryptography; each step uses one of several supported algorithms. Each public key is bound to a user name or an e-mail address. The first version of this system was generally known as a web of trust to contrast with the X.509 system, which uses a hierarchical approach based on certificate authority and which was added to PGP implementations later. Current versions of PGP encryption include both options through an automated key management server.



How PGP encryption works

Compatibility [edit]

As PGP evolves, versions that support newer features and algorithms are able to create encrypted messages that older PGP systems cannot decrypt, even with a valid private key. Therefore, it is essential that partners in PGP communication understand each other's capabilities or at least agree on PGP settings.

Confidentiality [edit]

PGP can be used to send messages confidentially. For this, PGP combines symmetric-key encryption and public-key encryption. The message is encrypted using a symmetric encryption algorithm, which requires a symmetric key. Each symmetric key is used only once and is also called a session key. The message and its session key are sent to the receiver. The session key must be sent to the receiver so they know how to decrypt the message, but to protect it during transmission it is encrypted with the receiver's public key. Only the private key belonging to the receiver can decrypt the session key.

Digital signatures [edit]

PGP supports message authentication and integrity checking. The latter is used to detect whether a message has been altered since it was completed (the *message integrity* property) and the former to determine whether it was actually sent by the person or entity claimed to be the sender (a *digital signature*). Because the content is encrypted, any changes in the message will result in failure of the decryption with the appropriate key. The sender

PGP Example

URGENT: Permission to Publish Auxiliary Material -- Your Camera-Ready Paper for OOPSLA 2017 (oopsla17main-oopsla242-p) 받은편지함 x

Conference Publishing - Info <info@conference-publishing.com> 17. 9. 28.
gifaranga, minseok_jeon, scha, hakjoo_oh에게

영어 › 한국어 메일 번역 영어 번역 안함 ×

[Data-Driven Context-Sensitivity for Points-to Analysis]
[Sehun Jeong, Minseok Jeon, Sungdeok Cha, and Hakjoo Oh]
[Korea University, South Korea]

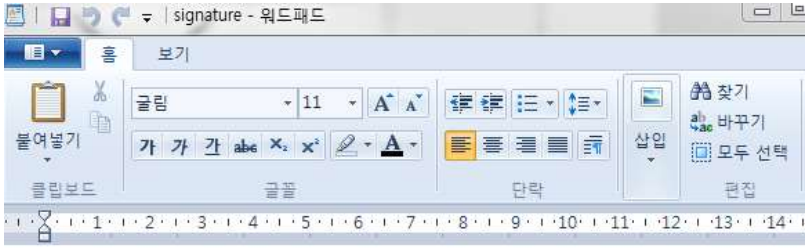
Dear Sehun Jeong, Minseok Jeon, Sungdeok Cha, and Hakjoo Oh,

You submitted Auxiliary Material (artifact) for inclusion in the ACM Digital Library,
but in your publishing-rights form, you did not assign the necessary permission to ACM.
This looks like a mistake made when completing the form.

Therefore, we removed your form and ask you to complete a new form,
in order to make your material available.
Please complete the new form as soon as possible:
http://campus2.acm.org/public/publications/oa.cfm?paperID=010501&email=575C575143505F5251755A5F4354501B51561F5B43&publicationID=020D04

Best regards,
Simone

signature.asc




```
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2

iQEcBAEBCAAGBQJZzJbEAAoJEHROW9KxiKI3czEIAKnGEcc37m2+tCgbwIJKNwRI
mY21NTW/cIdg2ukBrojnNazUIWd1wcjupb0PmI0lcz6FDAIJBMAyYZFDCX79EEgR
fSWFZrWDXJ1HQciiVr4OWe6lsfrE0uhBt/D4UqWxc9dorNIiIYln/3KVyESPrZUV
XNtj/TEhIhcIN07N+xXbiT7m5e/f3tmY82wu1uf/ZuVd0zKI63vNJ8M29n5487x1
LNAfRXWthSJzHKeWs/jdPAwsOQqCFIOotaZip/SolbFjMBAUNJ5SW9EaJTCNxJd3
6yc4L5BRzvQTkULtiD44lhoDa261n7WApzjAcv62nWSFByeb9zsadHrxYwETs4A=
=/IWO
-----END PGP SIGNATURE-----
```

Summary

- Three distinct aspects to the use of public-key encryption
  - The secure distribution of public keys
  - The use of public-key encryption to distribute secret keys
  - The use of public-key encryption to create temporary keys for message encryption