Computer security Lecture 8

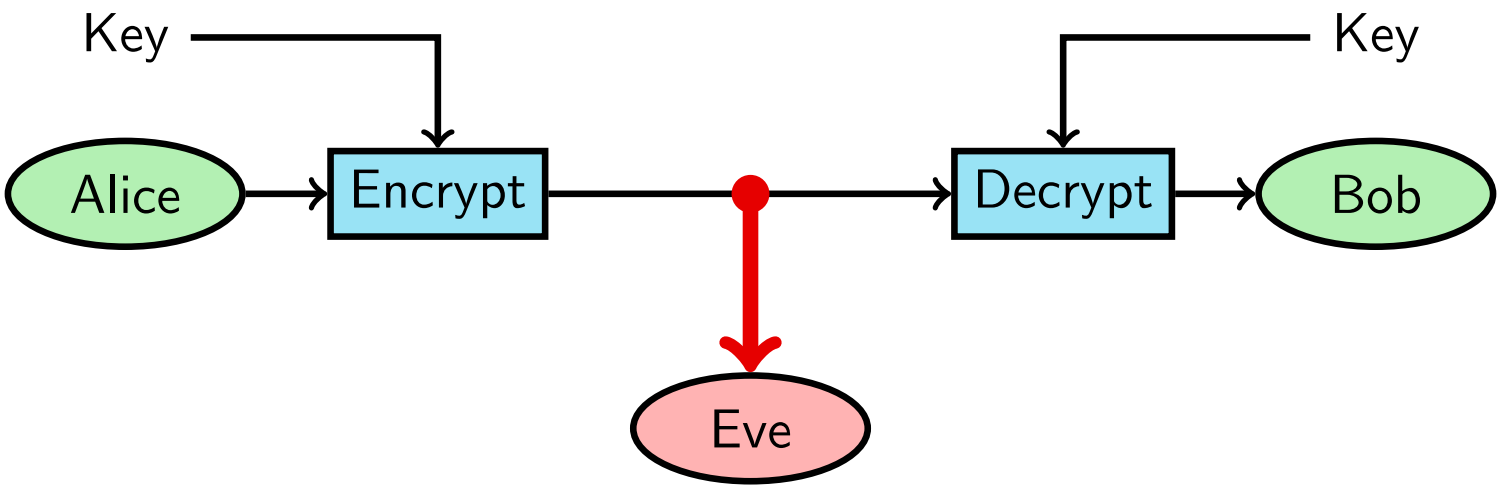
Cryptography as security tool

Jan-Åke Larsson

LINKÖPING UNIVERSITY

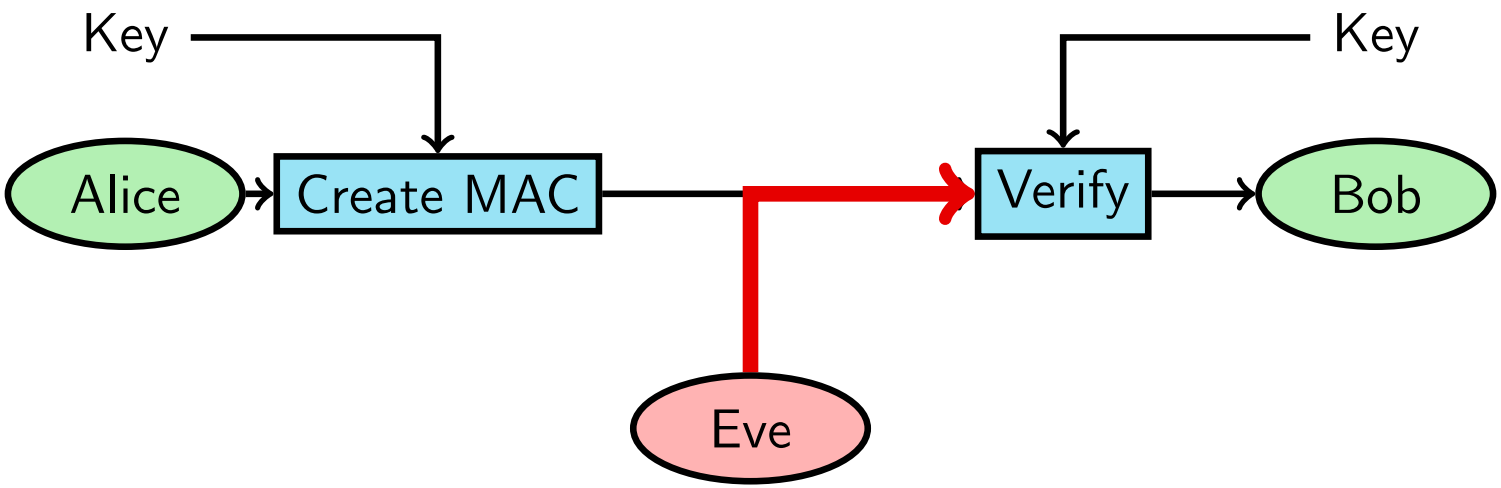
“Cryptography” is a Greek word that means “hidden writing”

Used to hide message from someone, and sometimes prevent them from creating a new message



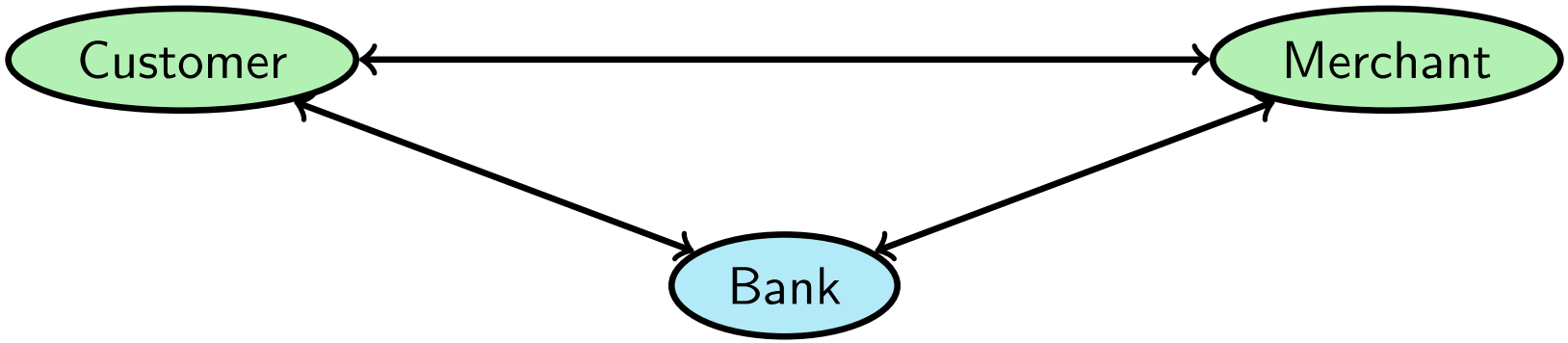
“Cryptography” is a Greek word that means “hidden writing”

Used to hide message from someone, and sometimes prevent them from creating a new message



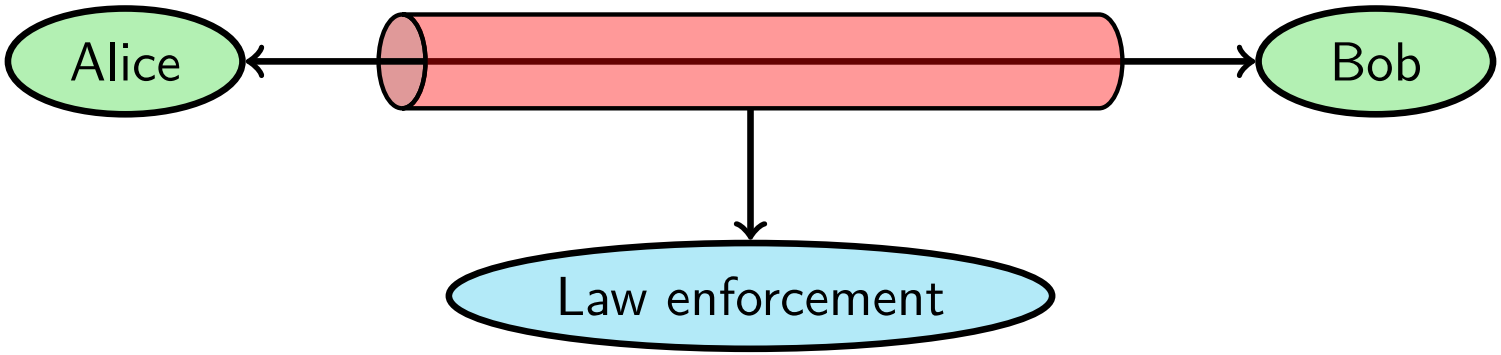
Cryptography is today used in other settings too

There are situations where participants don't really trust each other, but a third party



Cryptography is today used in other settings too

There are situations where the authorities want access to the communication channel



Cryptography

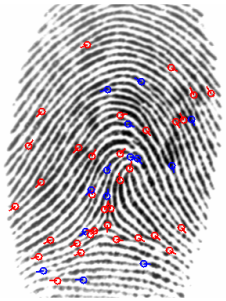
- A security *tool*, not a general solution
- Cryptography usually converts a communication security problem into a key management problem
- So now you must take care of the key security problem, which becomes a problem of computer security

Images of cryptographic tasks

Hide information securely,
opens with identical keys
(symmetric key cryptography)

Make something visible,
but hinder changes
(signatures)

Everyone can hide information, opens
with private key
(asymmetric key cryptography)



Enable simple checking of data (hash
functions)

Terminology

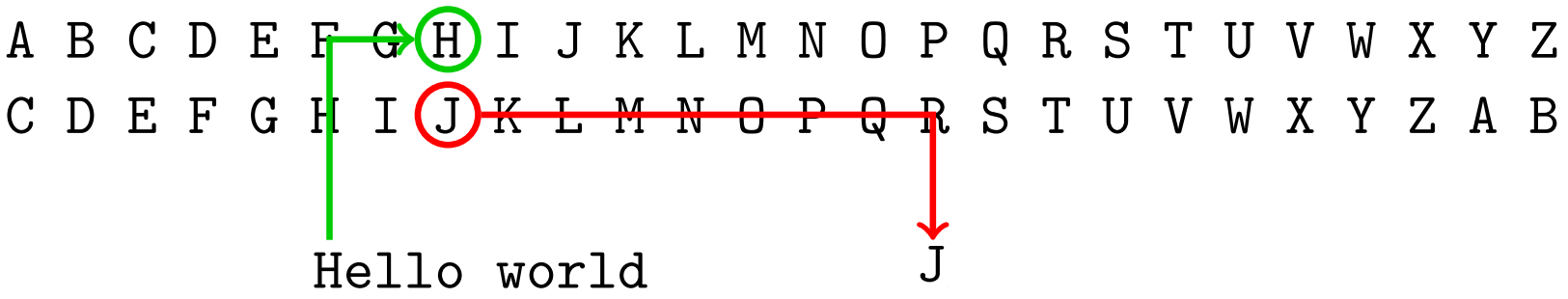
- The *plaintext* is the information in its normal form
- The *ciphertext* or *cryptogram* is the transformed plaintext
- The secret parameter for the encryption (known only to the sender and intended recipients) is called the *key*
- The key decides how the transformation is done

Kerckhoff's principle

- A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.

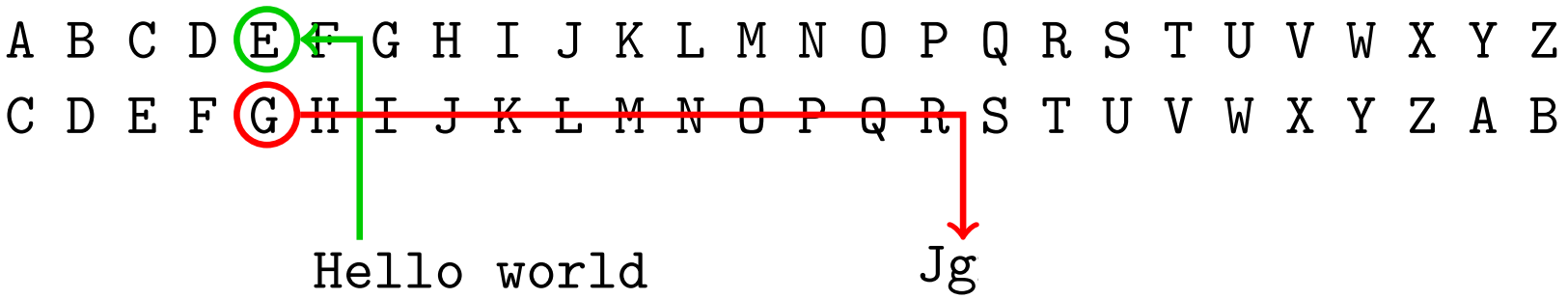
Classical crypto example: Caesar cipher

- Exchange every plaintext letter into the letter x positions further on in the alphabet
- The key is the letter that A is transformed into



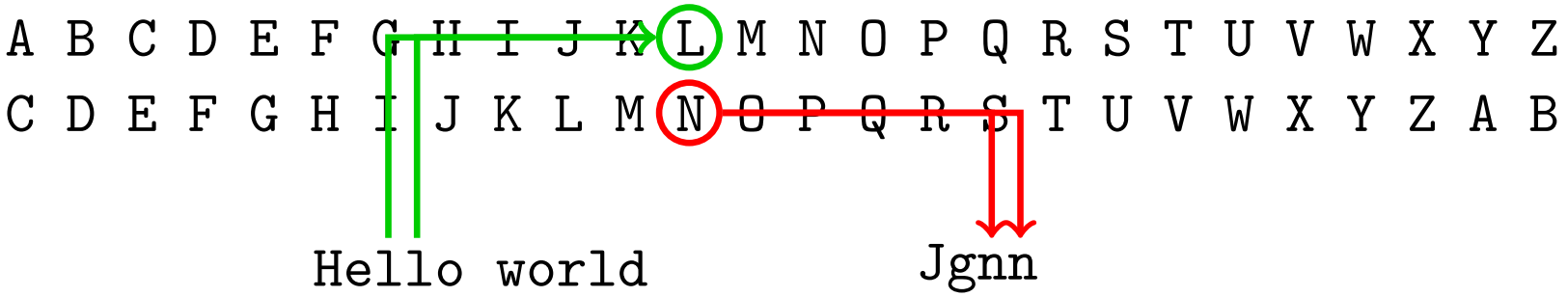
Classical crypto example: Caesar cipher

- Exchange every plaintext letter into the letter x positions further on in the alphabet
- The key is the letter that A is transformed into



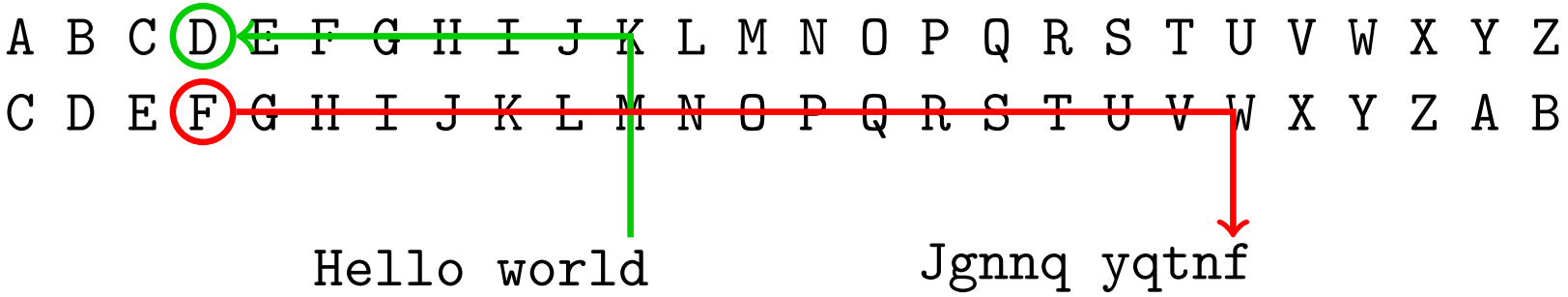
Classical crypto example: Caesar cipher

- Exchange every plaintext letter into the letter x positions further on in the alphabet
- The key is the letter that A is transformed into



Classical crypto example: Caesar cipher

- Exchange every plaintext letter into the letter x positions further on in the alphabet
- The key is the letter that A is transformed into



Alternative description of Caesar cipher

- Replace every plaintext letter with its (zero-offset) position in the alphabet ("A"=0, "B"=1, etc., up to the number of letters n)
- Express the key as an integer k using the same system
- If the plaintext as an integer is m, the cryptogram as an integer $c = m + k$ modulo n
- The cryptogram letter is then the letter corresponding to the number c

- The plaintext "H" gives $m = 7$, and $k = 2$ results in $c = 7 + 2 \pmod{26}$, so cryptogram is "J"

Breaking Caesar, example

- Cryptogram: Icnnkc qopkc fkxkuc guv kp rctvgu swcgtwo wpco kpeqnwpv Dgnikcg...
- Try each key, stop trying for each key when the plaintext becomes impossible

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
c	b	a	z	y	x	w	v	u	t	s	r	q	p	o	n	m	l	k	j	i	h	g	f	e	d
n		l	k	j		h	g	f	e	d	c	b		z	y	x			u	t	s	r		p	o
n		l		j				f	e	d				z	y				u	t		r		p	o
k		i		g				c	b					w	v				r	q		o		m	l
		a						u	t					o	n				j	i		g		e	d
		o						i	h					c					x	w		u		s	r
		m						g	f					a					v	u		s		q	p
		n						h						b						v		t		r	
		i						c						w						q		o		m	
		a												o											
		d												r											

Breaking Caesar, example

- Cryptogram: Icnnkc qopkc fkxkuc guv kp rctvgu swcgtwo wpco kpeqnwpv Dgnikcg...
- Try each key, stop trying for each key when the plaintext becomes impossible

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
c	b	a	z	y	x	w	v	u	t	s	r	q	p	o	n	m	l	k	j	i	h	g	f	e	d
n		l	k	j		h	g	f	e	d	c	b		z	y	x			u	t	s	r		p	o
n		l		j				f	e	d				z	y				u	t		r		p	o
k		i		g				c	b					w	v				r	q		o		m	l
		a												o	n							g		e	d
		o												z	y							u		s	r
		m												w	v							s		q	p
		n												u	t							t		r	
		i												r	q							o		m	
		a												o											
		d												r											

Only remaining possible key: C

Plaintext: Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, ...

Simple substitution

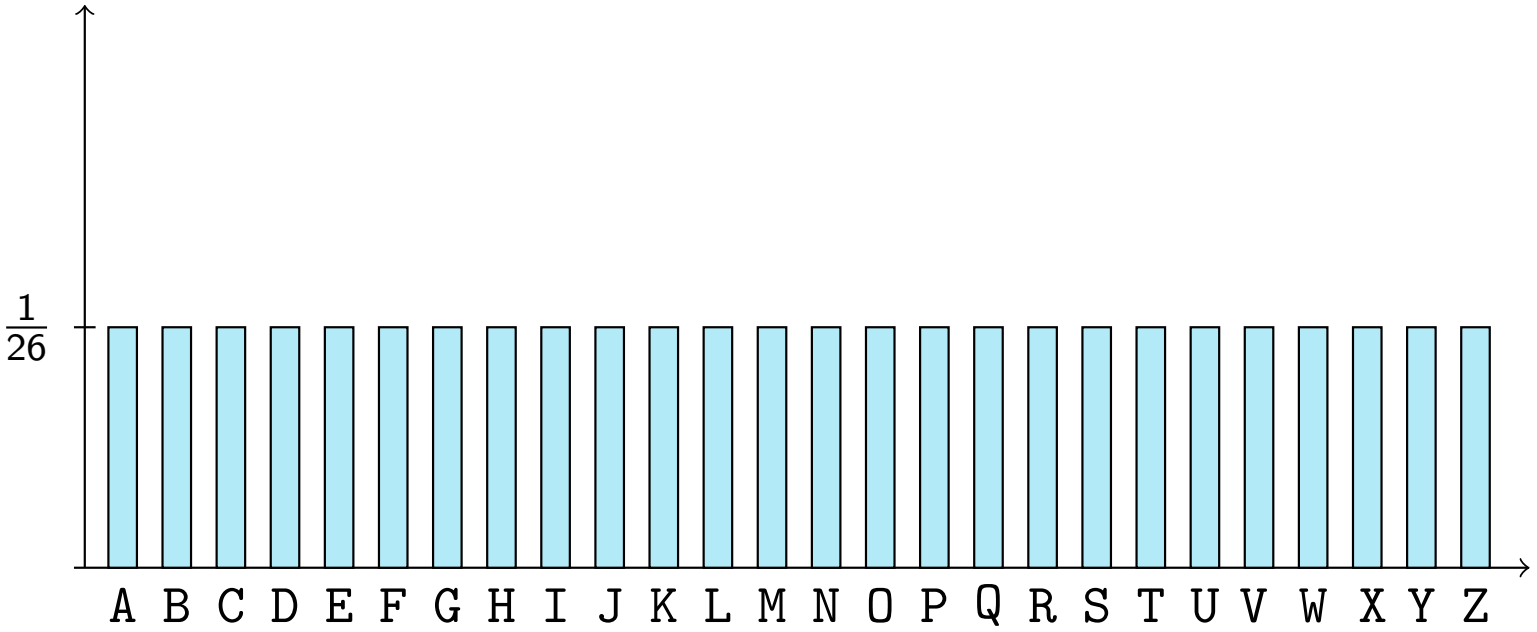
- Create a table of plaintext characters and their corresponding crypto characters
- Crypto characters can be just ordinary letters, but also anything else
- Each crypto character must occur only once in the table to enable unique decryption

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
G	Z	E	J	D	Y	I	T	Q	A	U	M	B	W	R	F	C	X	H	N	S	L	O	K	P	V

Breaking simple substitution

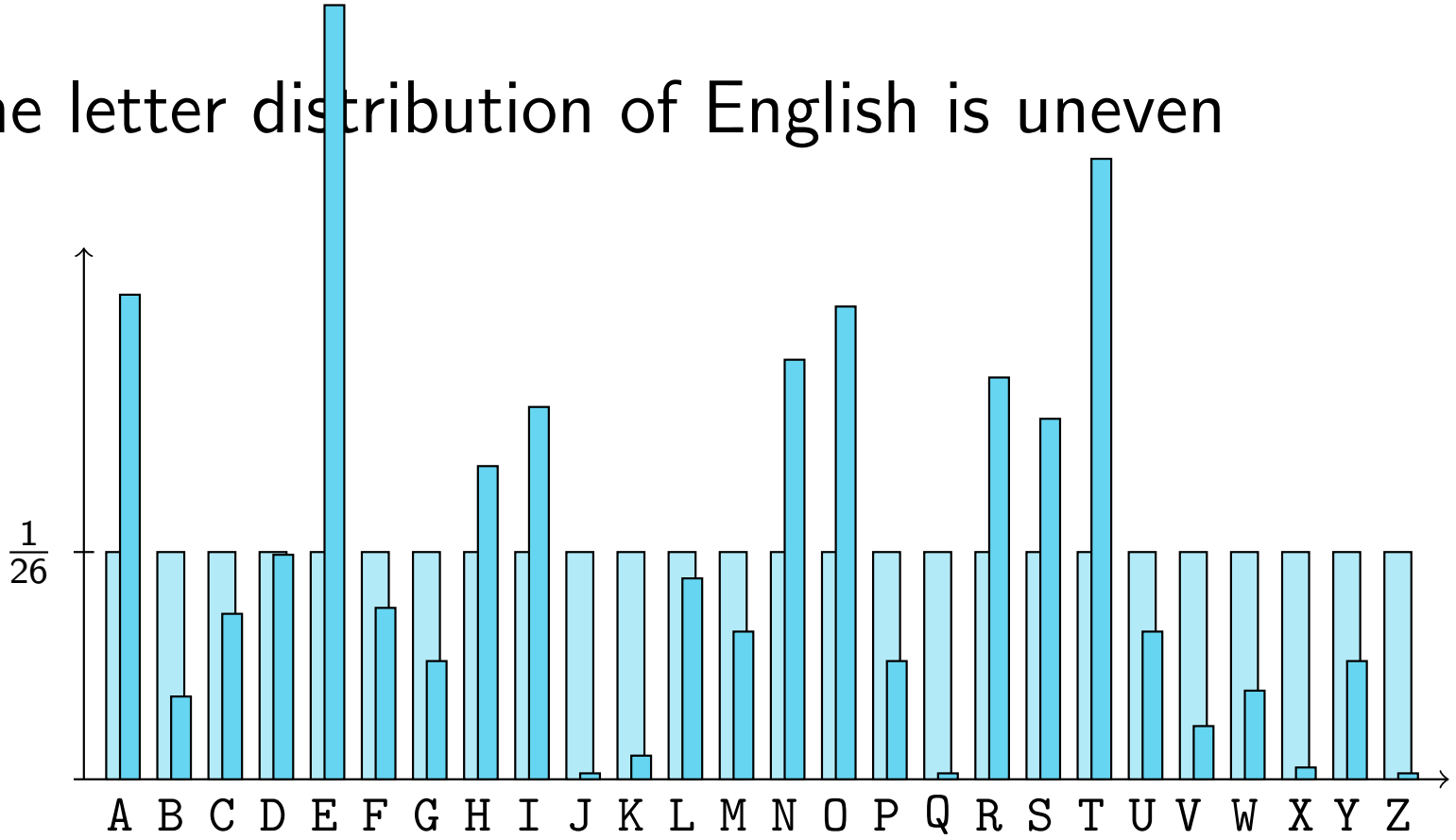
- Every crypto letter will occur exactly as often as its plaintext counterpart occurs in plaintext.
- Every combination of crypto letters (digrams, trigrams etc.) will occur as often as the corresponding plaintext combinations.
- Count how often letters, bigrams and trigrams occur in the cryptogram, and try to identify the ones corresponding to common plaintext letters and common letter combinations.
- Fill in so that remaining gaps form words.

The letter distribution of English is uneven



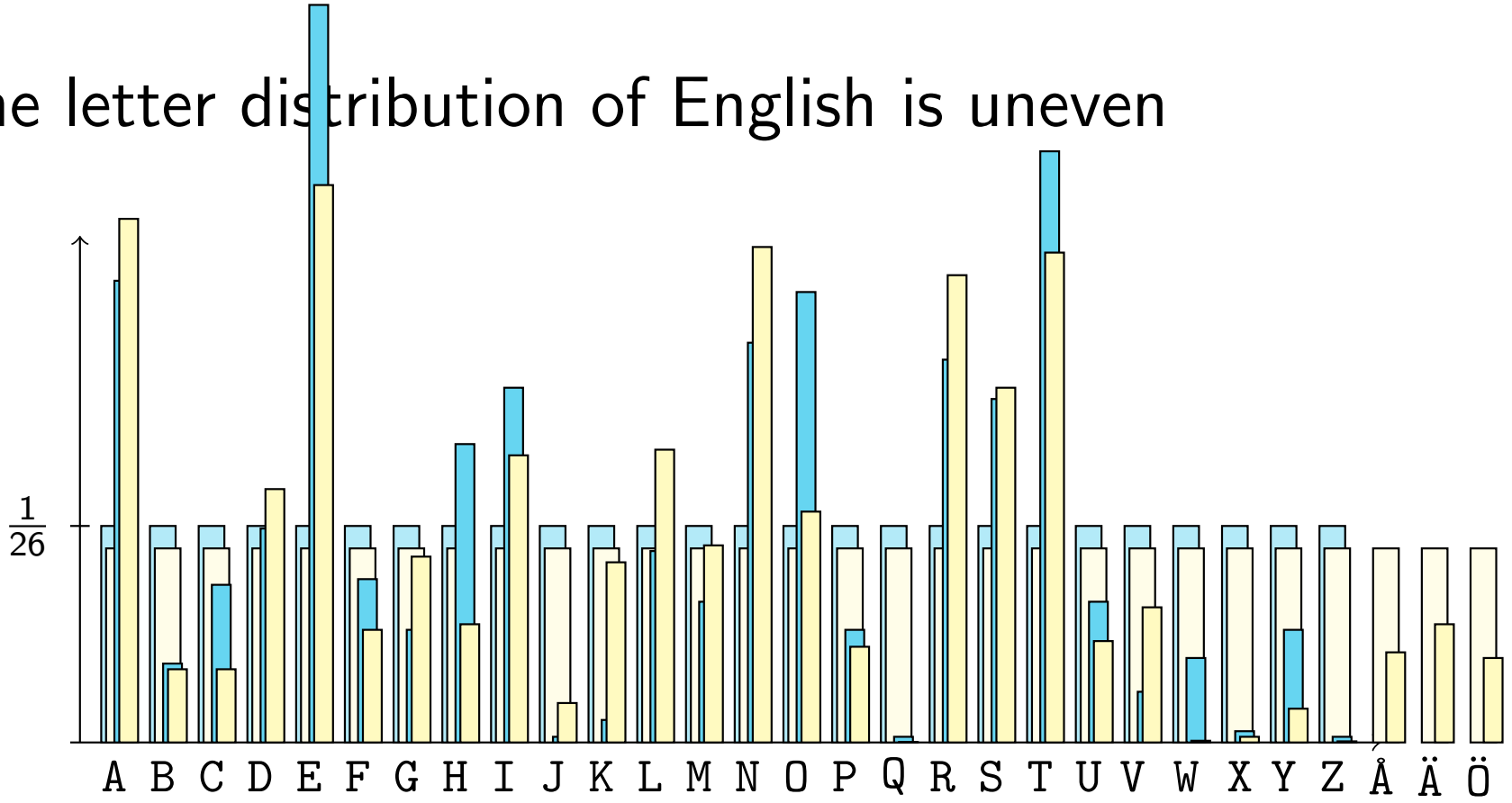
- An even distribution would look like the above

The letter distribution of English is uneven



- An even distribution would look like the above
- But the single letter distribution of English is uneven

The letter distribution of English is uneven



- An even distribution would look like the above
- But the single letter distribution of Swedish is uneven

Cryptanalysis

The goal of cryptanalysis can be

- reveal the key
- decrypt (part of) cryptograms without the key
- encrypt (part of) plaintexts without the key

In order to break a cipher you could

- Try all possible keys (brute force, exhaustive search)
- Use plaintext alphabet statistics
- Use both single letter statistics, and digram, trigram, and word statistics
- Do calculations adjusted to the algorithm

Algorithm strength

- This is measured in the amount of work needed to break the cipher
- The comparison is with brute force
- For Caesar crypto, you need to check 26 keys, or just under 2^5 values
- For simple substitution you need to check 26! keys ($\approx 4 \times 10^{26}$), or just over 2^{88} values
- But with letter statistics this drops quickly

Key size

- There are 31536000 seconds in a year $\approx 2^{25}$
- If you can try one key every microsecond, you can find a 45 bit key in around one year ($10^6 \approx 2^{20}$)
- If you have 1000 such processors, you can find a 55 bit key in around one year
- With 1000 processors that try one key every nanosecond, the expected time to find a 128 bit key is more than 10^{18} years

Key length

Table 7.4: Security levels (symmetric equivalent)

Security (bits)	Protection	Comment
32	Real-time, individuals	Only auth. tag size
64	Very short-term, small org	Not for confidentiality in new systems
72	Short-term, medium org Medium-term, small org	
80	Very short-term, agencies Long-term, small org	Smallest general-purpose < 4 years protection (E.g., use of 2-key 3DES, $< 2^{40}$ plaintext/ciphertexts)
96	Legacy standard level	2-key 3DES restricted to 10^6 plaintext/ciphertexts, ≈ 10 years protection
112	Medium-term protection	≈ 20 years protection (E.g., 3-key 3DES)
128	Long-term protection	Good, generic application-indep. Recommendation, ≈ 30 years
256	"Foreseeable future"	Good protection against quantum computers unless Shor's algorithm applies.

Strength of algorithms

- There is one perfectly secure algorithm: the One Time Pad, which needs a (random secret shared) key as long as the message
- Modern ciphers are immune to simple attacks
- Algorithm weakness is almost never a problem with approved algorithms today
- Badly chosen keys, bad key management, and bad implementations are the current problems

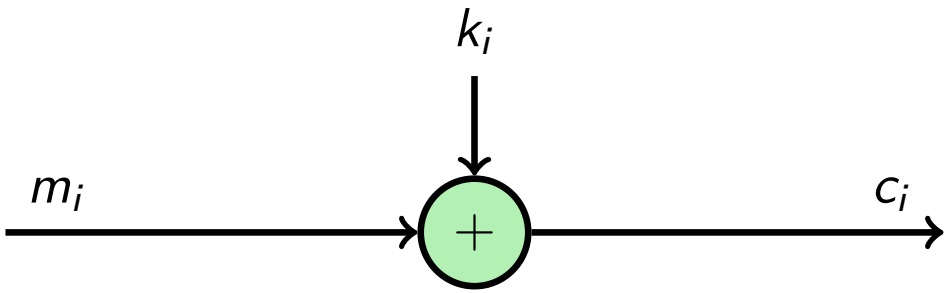
Algorithm types

There are three basic algorithm types

- Stream ciphers, very fast and suited for confidentiality, not data integrity
- Block ciphers, can be symmetric key which are fast, and asymmetric that are slower, both suited for confidentiality and integrity (with some precautions)
- One way functions, fast, only for data integrity

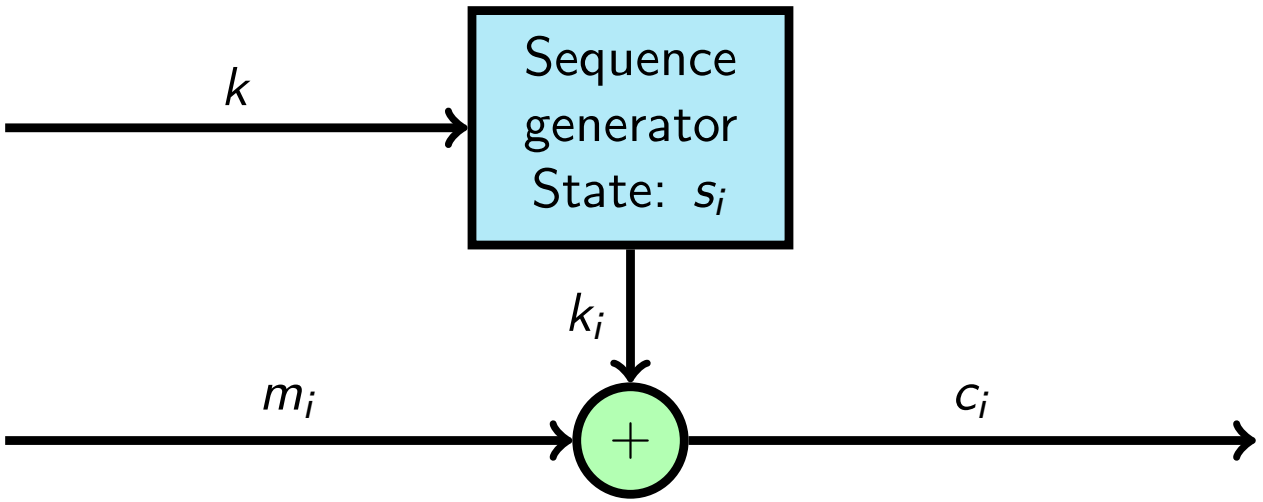
The perfect stream cipher: the One-time-pad

- Useful for sending secret data without preserved data integrity
- Use a long random bitsequence as key
- Add this to the plaintext to form the cryptogram



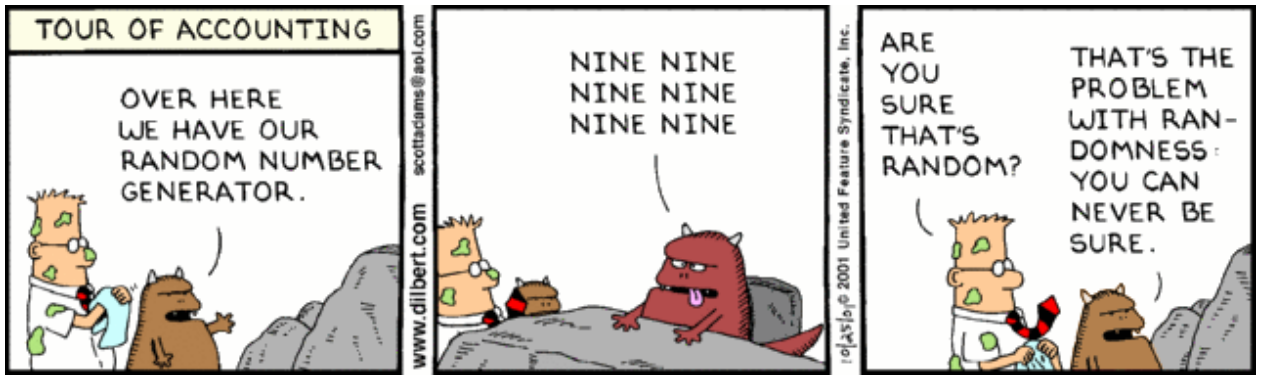
A practical stream cipher

- Useful for sending secret data without preserved data integrity
- Use a long *pseudo*-random bitsequence as key, generated from a short seed (key) k and initial state s_0
- Add this to the plaintext to form the cryptogram



Proper randomness, and sharing it

- Generate a pseudorandom "key stream" from a seed, a "real key" much shorter than the full "key stream" added to the message
- Make the set of possible seeds, the real keys, so large that exhaustive search is impossible
- Eliminate shortcuts to finding this key from the "key stream"



Stream ciphers and message integrity

- Imagine you are able to intercept the bank transfer that puts your salary into your bank account, and that it is encrypted with a stream cipher, and not authenticated
- You know what your salary is, and in what position it is on the bank transfer:

```
... CR LF SPC SPC SPC SPC SPC SPC  3  0  0  0  0 CR LF ...
... 13 10  32  32  32  32  32  32 51 48 48 48 48 13 10 ...
```

Stream ciphers and message integrity

- Imagine you are able to intercept the bank transfer that puts your salary into your bank account, and that it is encrypted with a stream cipher, and not authenticated
- You know what your salary is, and in what position it is on the bank transfer:

```
... CR LF SPC SPC SPC SPC SPC SPC  3   0   0   0   0  CR LF ...
... 13 10  32  32  32  32  32  32 51  48  48  48  48  13 10 ...
```

- You intercept the ciphertext

```
... 74 112 4 19 235 156 99 113 125 19 1 192 191 25 212 ...
```

Stream ciphers and message integrity

- Imagine you are able to intercept the bank transfer that puts your salary into your bank account, and that it is encrypted with a stream cipher, and not authenticated
- You know what your salary is, and in what position it is on the bank transfer:

```
... CR LF SPC SPC SPC SPC SPC SPC  3  0  0  0  0 CR LF ...
... 13 10  32  32  32  32  32  32 51 48 48 48 48 13 10 ...
```

- You intercept the ciphertext

```
... 74 112 4 19 235 156 99 113 125 19 1 192 191 25 212 ...
```

- The 19 in the ciphertext is at the same position as the first 48 in the cleartext

Stream ciphers and message integrity

- Imagine you are able to intercept the bank transfer that puts your salary into your bank account, and that it is encrypted with a stream cipher, and not authenticated
- You know what your salary is, and in what position it is on the bank transfer:

```
... CR LF SPC SPC SPC SPC SPC SPC 3  3  0  0  0  CR LF ...
... 13 10  32  32  32  32  32  32 51 51 48 48 48 13 10 ...
```

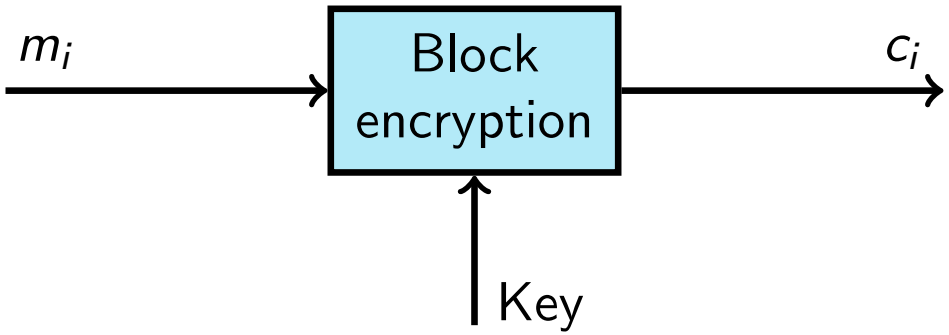
- You intercept the ciphertext

```
... 74 112 4 19 235 156 99 113 125 16 1 192 191 25 212 ...
```

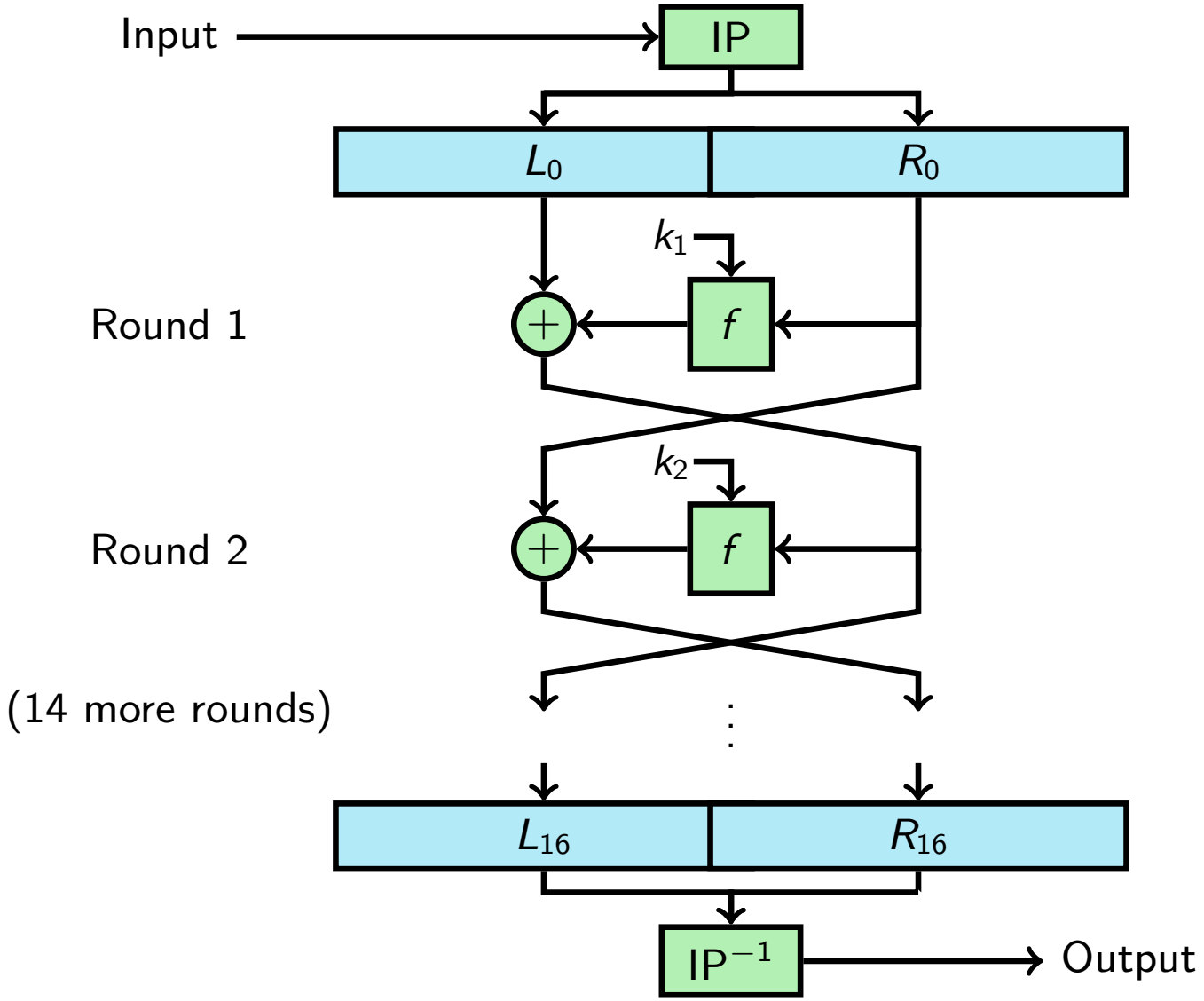
- Changing 19 to 16 in the ciphertext changes the first 48 to 51 in the received cleartext. Noone will notice :)

Block ciphers

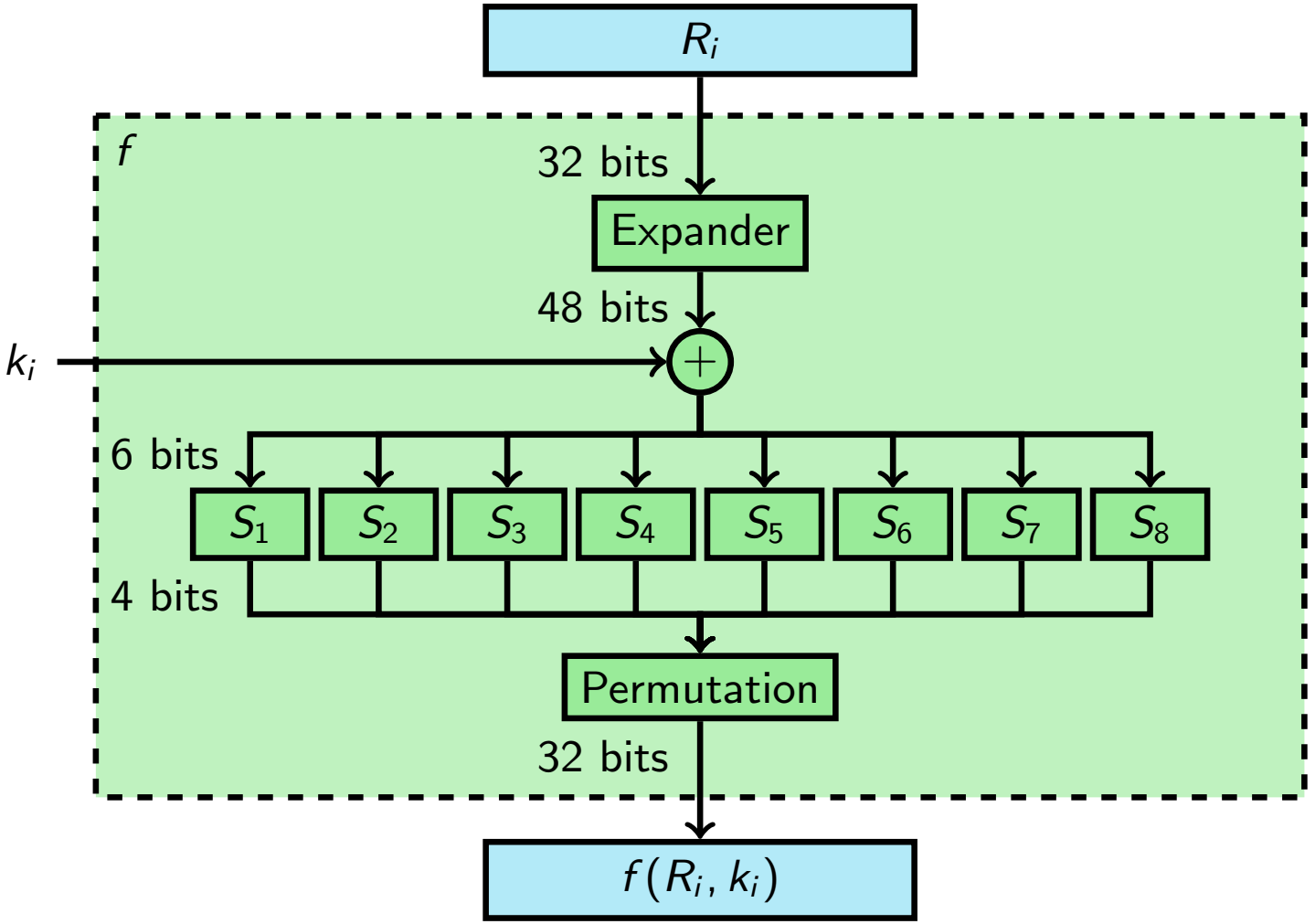
- Messages are treated in blocks of characters with fixed block size
- The key remains fixed for at least one session



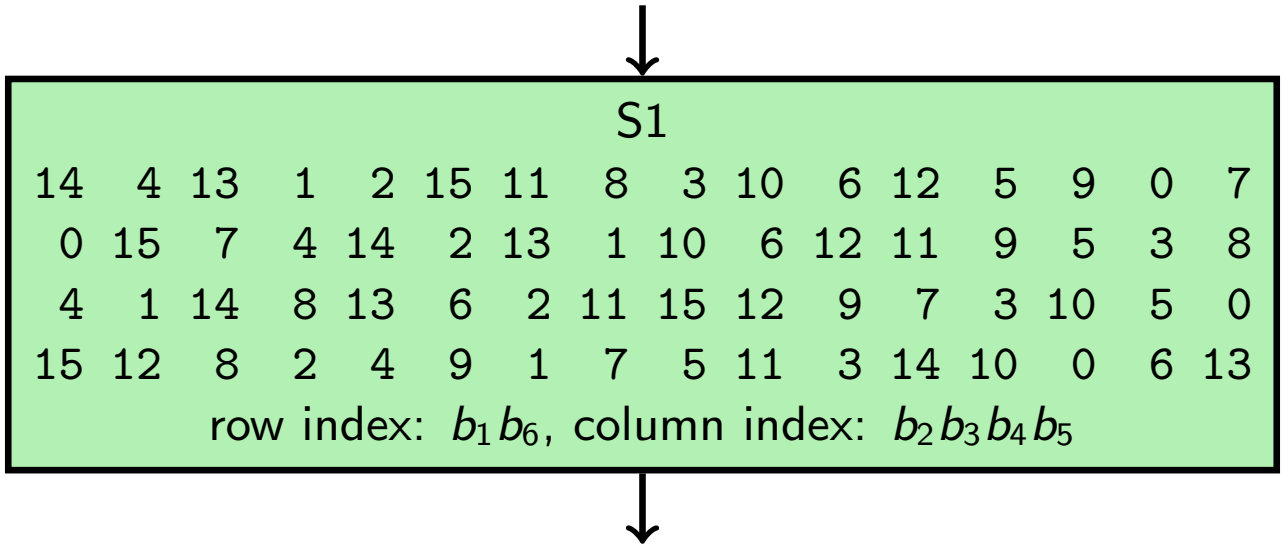
Data Encryption Standard



DES Substitution boxes



DES



- There was a lot of controversy surrounding the S-box construction
- People were worried there were backdoors in the system
- But in the late eighties it was found that even small changes in the S-boxes gave a weaker system
- After the (re-)discovery of *differential cryptanalysis*, in 1994 IBM published the construction criteria

Key length

Table 7.4: Security levels (symmetric equivalent)

Security (bits)	Protection	Comment
32	Real-time, individuals	Only auth. tag size
64	Very short-term, small org	Not for confidentiality in new systems
72	Short-term, medium org Medium-term, small org	
80	Very short-term, agencies Long-term, small org	Smallest general-purpose < 4 years protection (E.g., use of 2-key 3DES, $< 2^{40}$ plaintext/ciphertexts)
96	Legacy standard level	2-key 3DES restricted to 10^6 plaintext/ciphertexts, ≈ 10 years protection
112	Medium-term protection	≈ 20 years protection (E.g., 3-key 3DES)
128	Long-term protection	Good, generic application-indep. Recommendation, ≈ 30 years
256	"Foreseeable future"	Good protection against quantum computers unless Shor's algorithm applies.

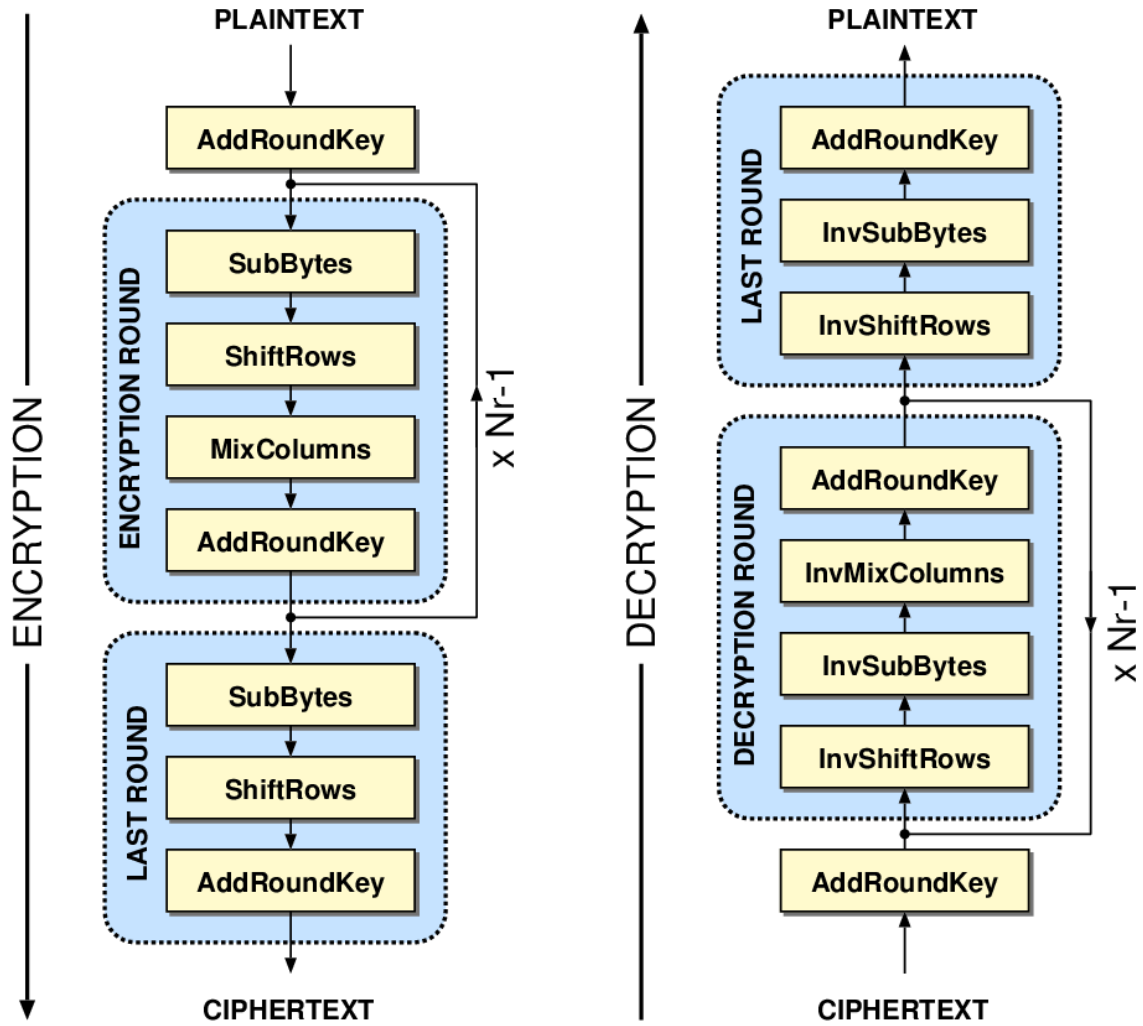
AES competition

- NIST put out a call for new algorithms in 1997, this was the start of the AES competition
- Requirements were: 128 bit blocks, 128, 192 and 256 bit keys, and that it should work well on several kinds of hardware and in software
- Five finalists
  - Rijndael (J. Daemen and V. Rijmen)
  - Serpent (R. Anderson, E. Biham, and L. Knudsen)
  - Twofish (B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson)
  - RC6 (RSA Labs)
  - MARS (IBM)

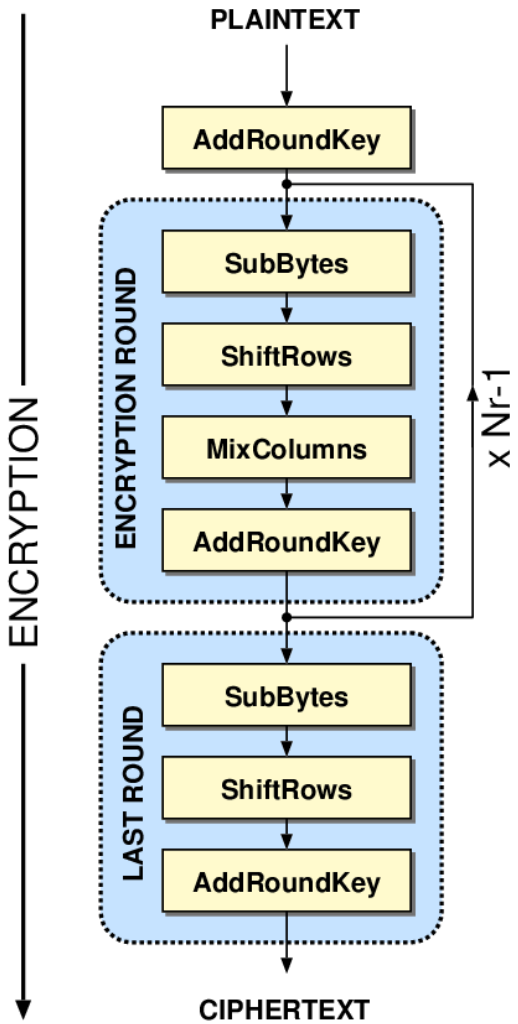
AES competition

- NIST put out a call for new algorithms in 1997, this was the start of the AES competition
- Requirements were: 128 bit blocks, 128, 192 and 256 bit keys, and that it should work well on several kinds of hardware and in software
- Five finalists
  - Rijndael (J. Daemen and V. Rijmen) $\leftarrow$ AES (2000)
  - Serpent (R. Anderson, E. Biham, and L. Knudsen)
  - Twofish (B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson)
  - RC6 (RSA Labs)
  - MARS (IBM)

AES structure: 10, 12, or 14 rounds



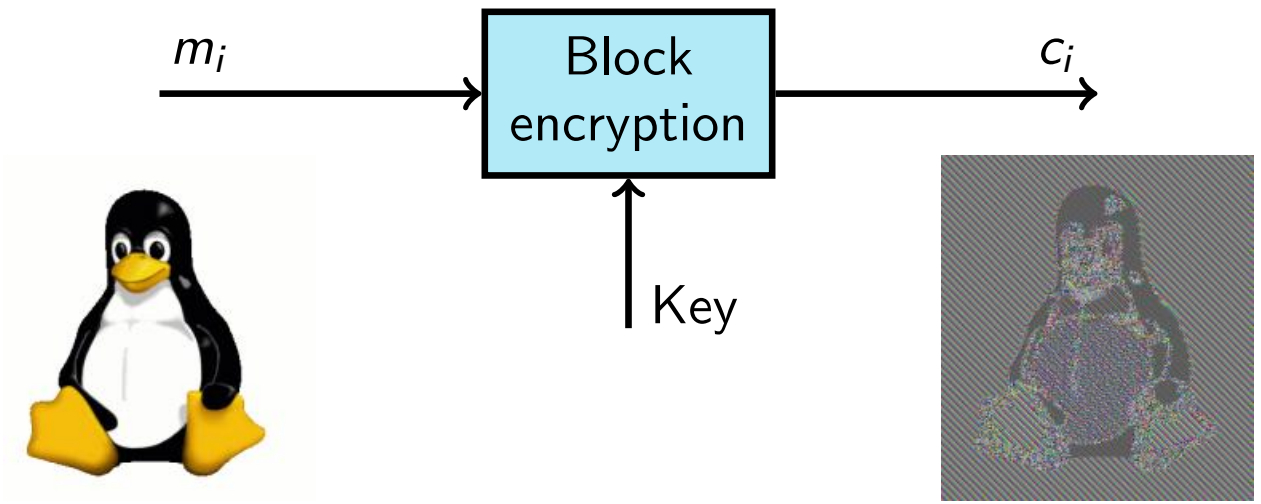
AES structure: 10, 12, or 14 rounds



- Not a Feistel system
- Addition modulo 2 of the round key
- 8-bit "S-boxes", this time chosen algebraically as x^{-1} in GF(2^8)
- Shift 8-bit parts between 32-bit sub-blocks, then mix bits in each 32-bit block
- Uses "whitening"
- The key schedule uses the S-box

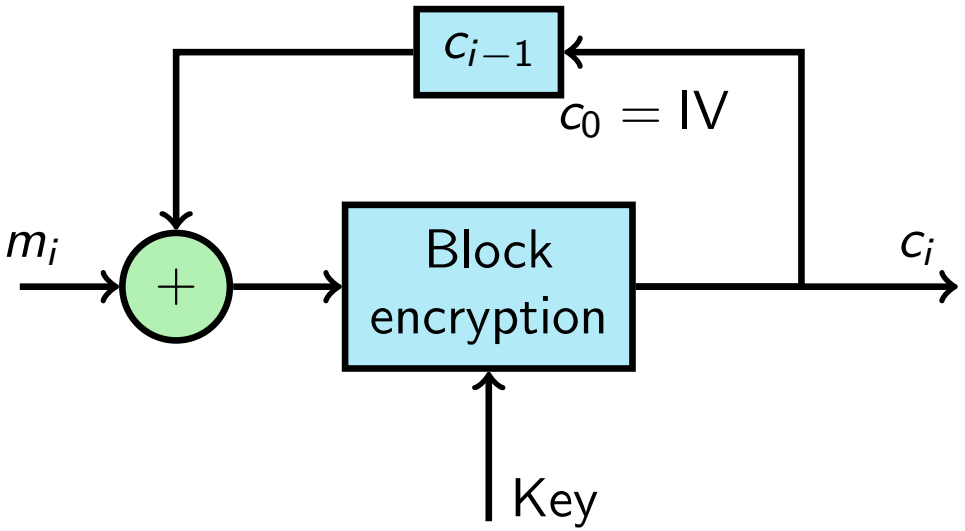
Block ciphers

- Messages are treated in blocks of characters with fixed block size
- The key remains fixed for at least one session
- Direct use ("Electronic Code Book") makes repeated plaintext blocks produce repeated cipher blocks
- Don't use ECB



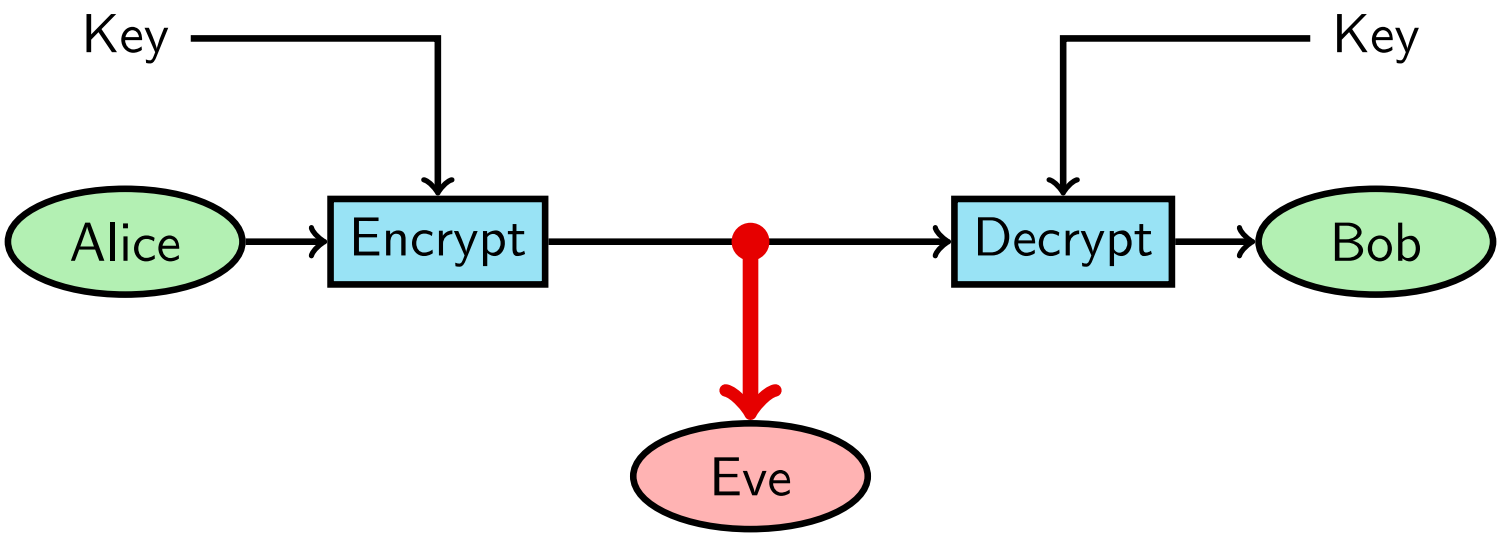
Block cipher in Cipher Block Chaining mode

- Most common mode to preserve data integrity
- One whole block at a time
- Feedback of previous cipher block
- Propagates transmission errors (one bit transmission error destroys one whole block plus one more bit)



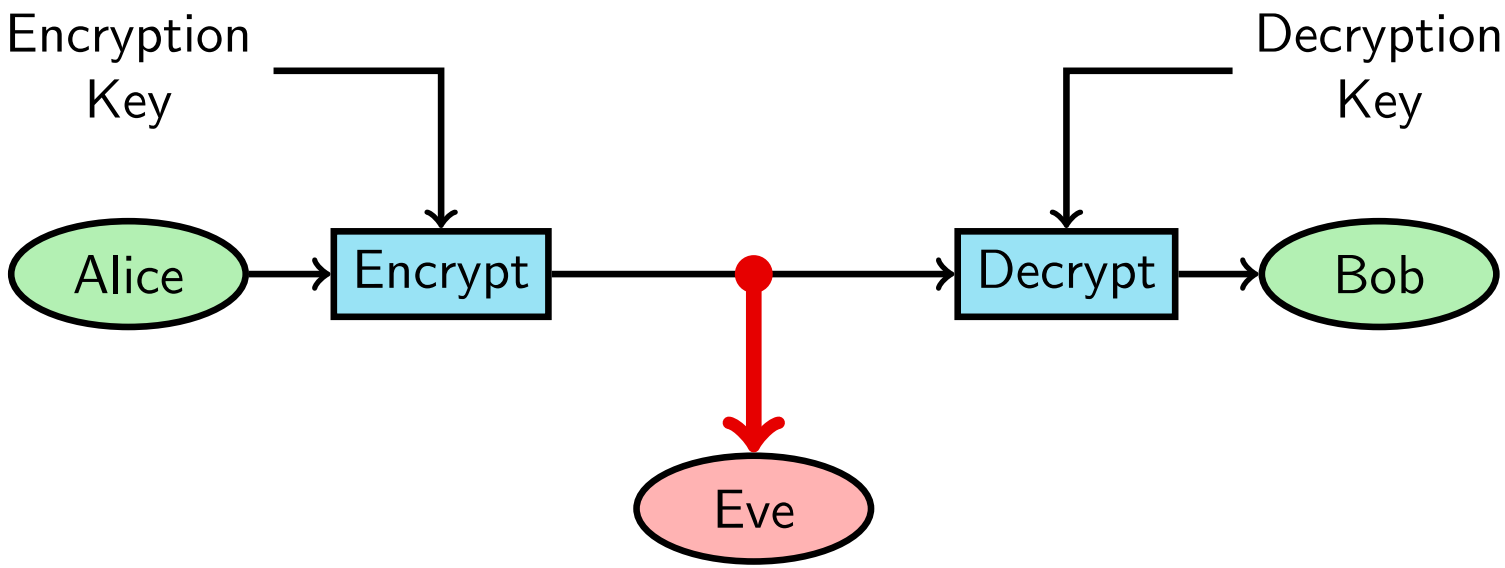
Symmetric key cryptography

In symmetric key systems, the encryption key is the same as the decryption key



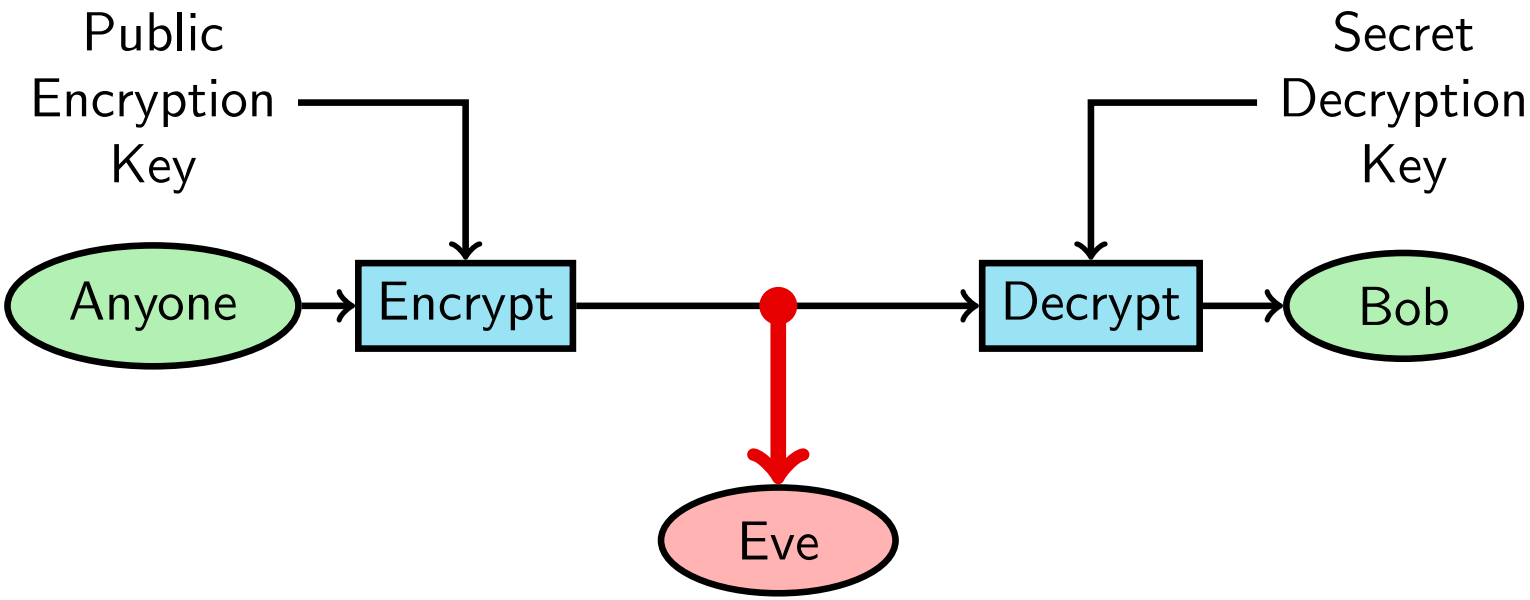
Asymmetric key cryptography

In asymmetric key systems, the encryption key and the decryption key are different

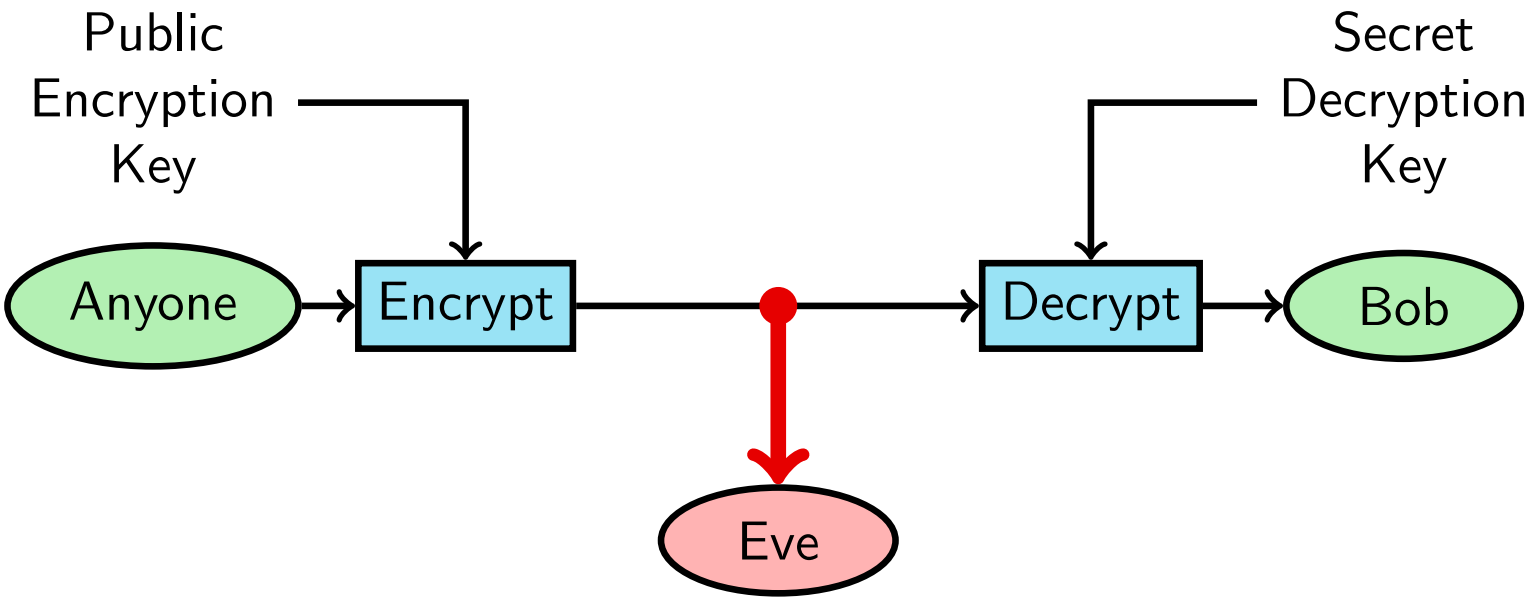


Public key cryptography

The encryption key can be public, so that anyone can send secret messages to Bob



Public key cryptography

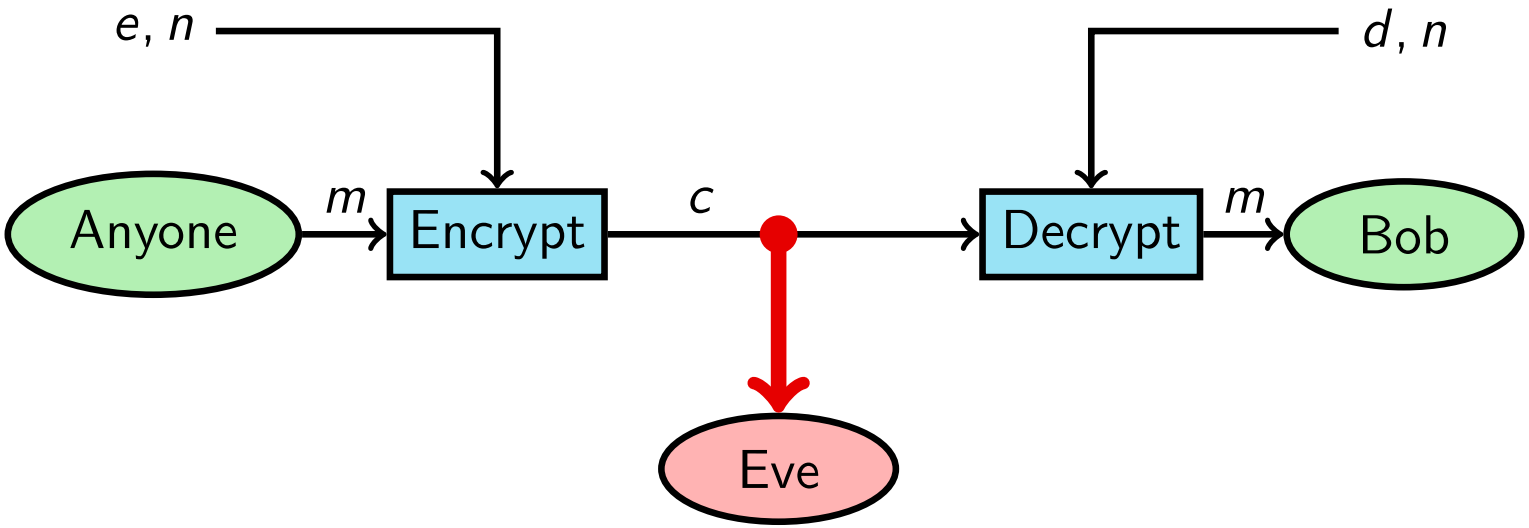


- Each user has a pair of keys, one public encryption key e, one secret decryption key d
- It is computationally hard to calculate d from e
- This is achieved by basing the algorithm on a difficult problem from number theory

Public key example: RSA

- Choose two very large primes p and q, and publish $n = pq$
- Choose a public encryption key e $(\gcd(e, (p-1)(q-1)) = 1)$
- Calculate the secret decryption key $d = e^{-1} \bmod (p-1)(q-1)$

- Encrypt with $c = m^e \bmod n$
- Decrypt with $c^d = m^{ed} = m^1 = m \bmod n$



Key length

Table 7.2: Key-size Equivalence.

Security (bits)	RSA	DLOG		EC
		field size	subfield	
48	480	480	96	96
56	640	640	112	112
64	816	816	128	128
80	1248	1248	160	160
112	2432	2432	224	224
128	3248	3248	256	256
160	5312	5312	320	320
192	7936	7936	384	384
256	15424	15424	512	512

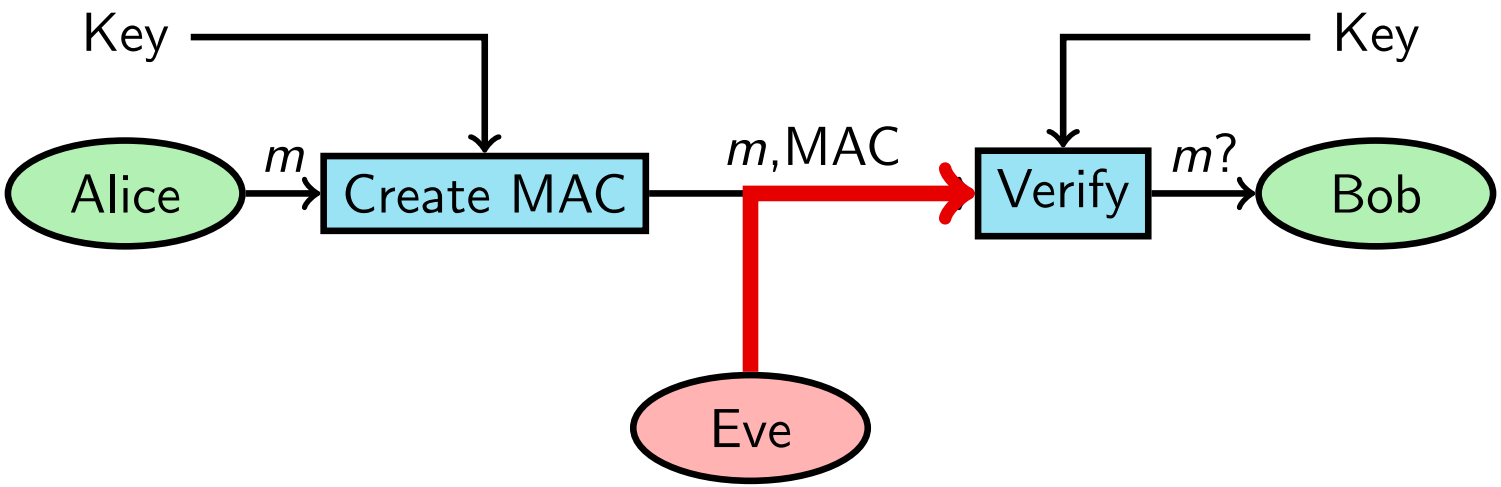
Table 7.3: Effective Key-size of Commonly used RSA/DLOG Keys.

RSA/DLOG Key	Security (bits)
512	50
768	62
1024	73
1536	89
2048	103

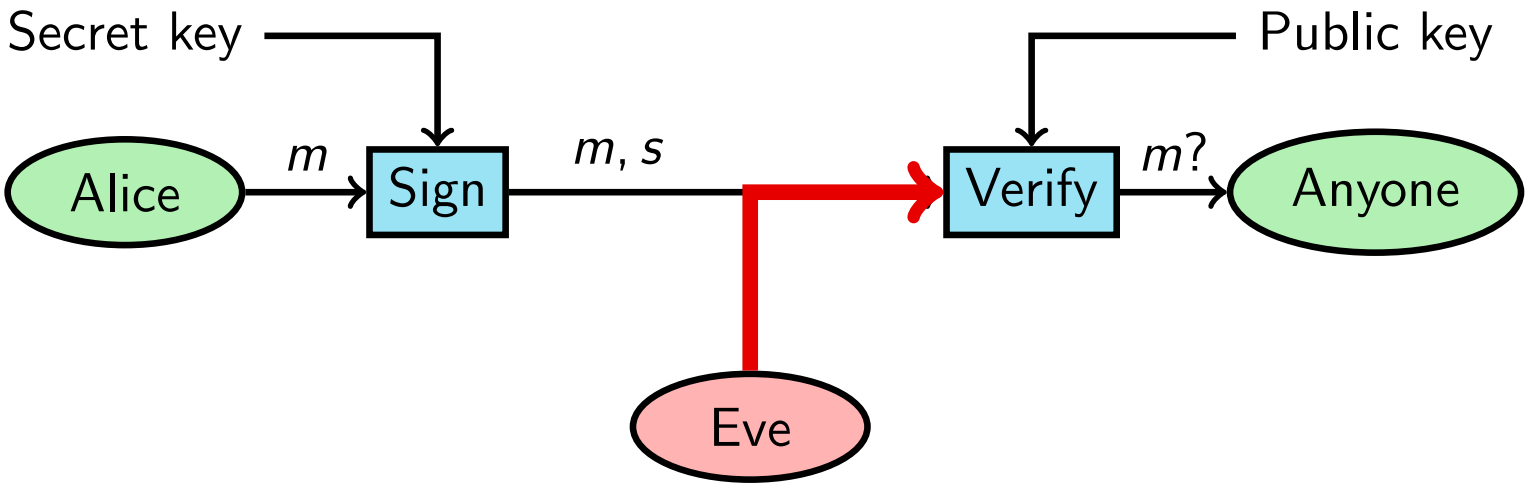
From "ECRYPT II Yearly Report on Algorithms and Keysizes (2009-2010)"

MACs and digital signatures

- A Message Authentication Code is a "seal of autenticity" created in a symmetric key system
- If the same MAC can be created at the verifier, then the message hasn't changed in transit
- Everyone that can check the MAC can create the MAC

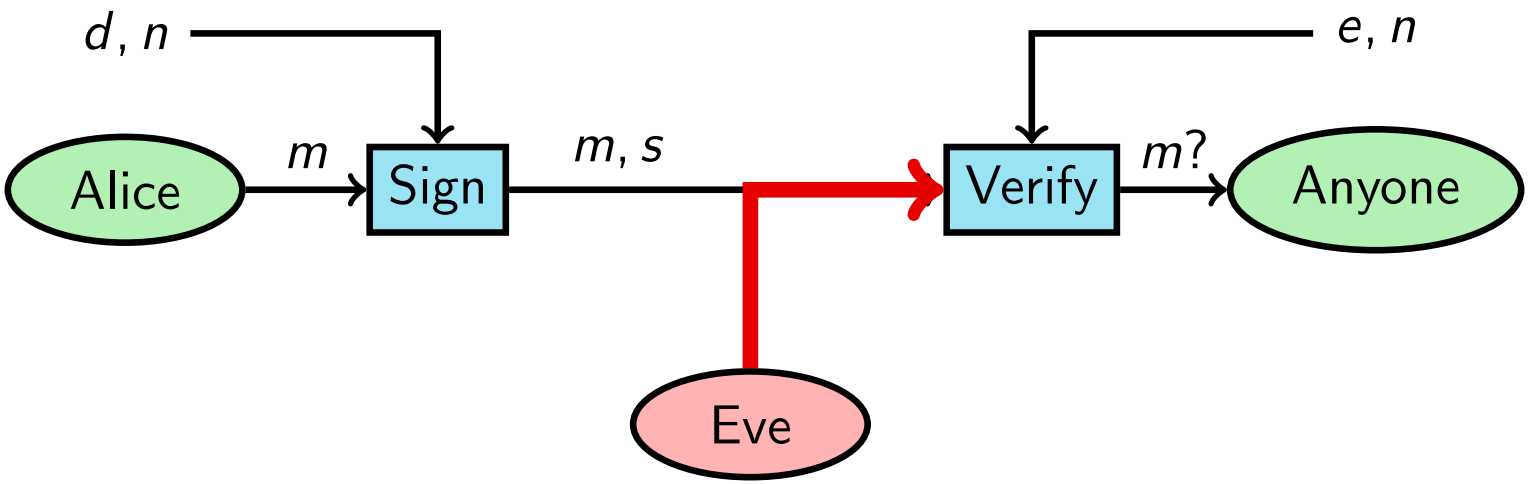


MACs and digital signatures



- A Digital Signature is a "seal of autenticity" created in an asymmetric key system
- If the Digital Signature can be verified at the verifier, then the message hasn't changed in transit
- Here, only Alice can sign

MACs and digital signatures



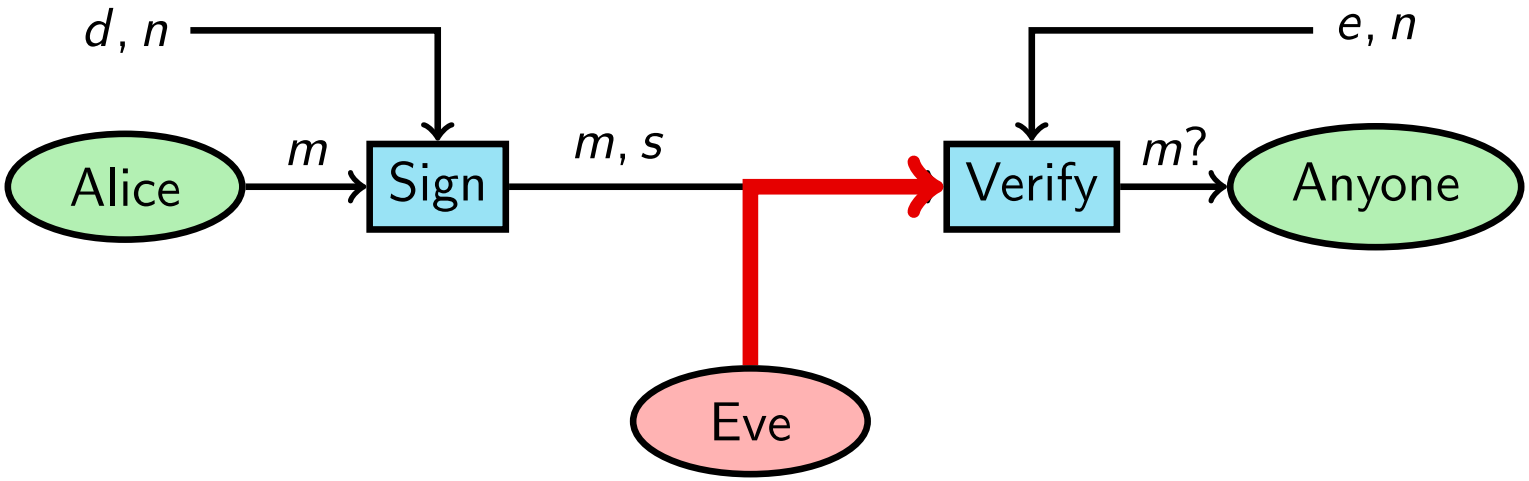
- In RSA, the secret "decryption" key now is the secret signing key, while the public "encryption" key now is the public verification key
- The same pair e, d should not be used both for confidentiality and integrity

One-way hash functions

- Large messages take time to MAC or sign
- A hash function creates an output that is much smaller than the message (or file)
- For any hash function, it should be easy to calculate $h(x)$ from x
- Then, it is easy to create MACs or digital signatures for $h(x)$

- A one-way function is one where it is easy to calculate $y = f(x)$, but computationally hard to calculate $x = f^{-1}(y)$

Example: RSA digital signatures



- Set up for RSA and choose a collision-resistant hash function
- Sign using $s = (h(m))^d \bmod n$
- Verify by checking that $h(m)$ equals $s^e \bmod n$

One-way hash functions

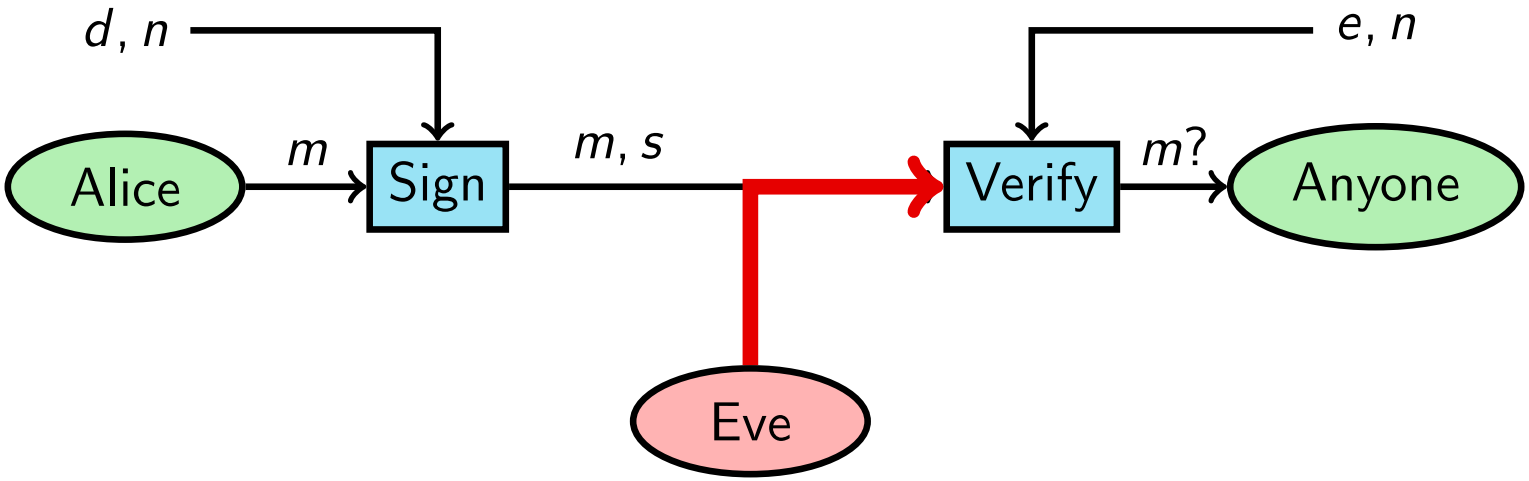
- A one-way function is one where it is easy to calculate $y = f(x)$, but computationally hard to calculate $x = f^{-1}(y)$

- But a hash function has output much shorter than the input
- There must be messages for which the output collides (so f^{-1} doesn't exist)
- "Collision resistant" does not mean that no two messages get the same hash value, it means it is hard to find two such messages

Three kinds of hash functions

- For a one-way hash function, it is hard to find a preimage: to calculate x' from $h(x)$ so that $h(x') = h(x)$
- For a weakly collision-resistant hash function, it is hard to find a second preimage: to calculate x' from x so that $h(x') = h(x)$
- For a strongly collision-resistant hash function, it is hard to find a collision: to find x and x' $(x' \neq x)$ such that $h(x') = h(x)$

Example: RSA digital signatures



- Set up for RSA and choose a collision-resistant hash function
- Sign using $s = (h(m))^d \bmod n$
- Verify by checking that $h(m)$ equals $s^e \bmod n$

Standard algorithms?

- ISO does not standardize algorithms
- NIST (National Institute for Standards and Technology) does register standards for federal use in the US
- NIST standards and other well tested algorithms are used as some kinds of de facto standards
- Examples are DES, AES and the SHA family

De facto standard algorithms

- Stream ciphers: The A family in GSM and E family in Bluetooth
- Symmetric block ciphers: DES (56 bits), AES (128-256 bits), Blowfish, ...
- Asymmetric ciphers: RSA (any key size) and Elliptic Curve Cryptography (ECC, any key size), ...
- Cryptographic hashes: MD5 (128 bit hashes, weak), SHA-1 (160 bit hashes, deprecated), SHA-2 (224-512 bit hashes), SHA-3 (224-512 bit hashes, ?), ...

Key length

Table 7.4: Security levels (symmetric equivalent)

Security (bits)	Protection	Comment
32	Real-time, individuals	Only auth. tag size
64	Very short-term, small org	Not for confidentiality in new systems
72	Short-term, medium org Medium-term, small org	
80	Very short-term, agencies Long-term, small org	Smallest general-purpose < 4 years protection (E.g., use of 2-key 3DES, $< 2^{40}$ plaintext/ciphertexts)
96	Legacy standard level	2-key 3DES restricted to 10^6 plaintext/ciphertexts, ≈ 10 years protection
112	Medium-term protection	≈ 20 years protection (E.g., 3-key 3DES)
128	Long-term protection	Good, generic application-indep. Recommendation, ≈ 30 years
256	"Foreseeable future"	Good protection against quantum computers unless Shor's algorithm applies.

Key length

Table 7.2: Key-size Equivalence.

Security (bits)	RSA	DLOG		EC
		field size	subfield	
48	480	480	96	96
56	640	640	112	112
64	816	816	128	128
80	1248	1248	160	160
112	2432	2432	224	224
128	3248	3248	256	256
160	5312	5312	320	320
192	7936	7936	384	384
256	15424	15424	512	512

Table 7.3: Effective Key-size of Commonly used RSA/DLOG Keys.

RSA/DLOG Key	Security (bits)
512	50
768	62
1024	73
1536	89
2048	103

From "ECRYPT II Yearly Report on Algorithms and Keysizes (2009-2010)"

Basic questions for crypto users

- Is the algorithm strong enough?
- Does the algorithm really work against my perceived threat?
- What about key management?
- What standard algorithm should I use?
- NEVER design a cryptosystem on your own
- Watch out for snake oil