Cryptography

Cryptography: terminology (1/2)

Cryptography

- Art or science of hidden writing (confidential writing)
  - from Gr. kryptós, hidden + graph, r. de graphein, to write
- Initially used to maintain confidentiality of information
- Steganography: art of concealing data
  - from Gr. steganós, hidden + graph, r. de graphein, to write

Cryptanalysis

- Art or science of breaking Cryptographic systems or encrypted information

Cryptology

- Cryptography + cryptanalysis

Cryptography: terminology (2/2)

Cipher

- Specific cryptographic technique

Cipher operation

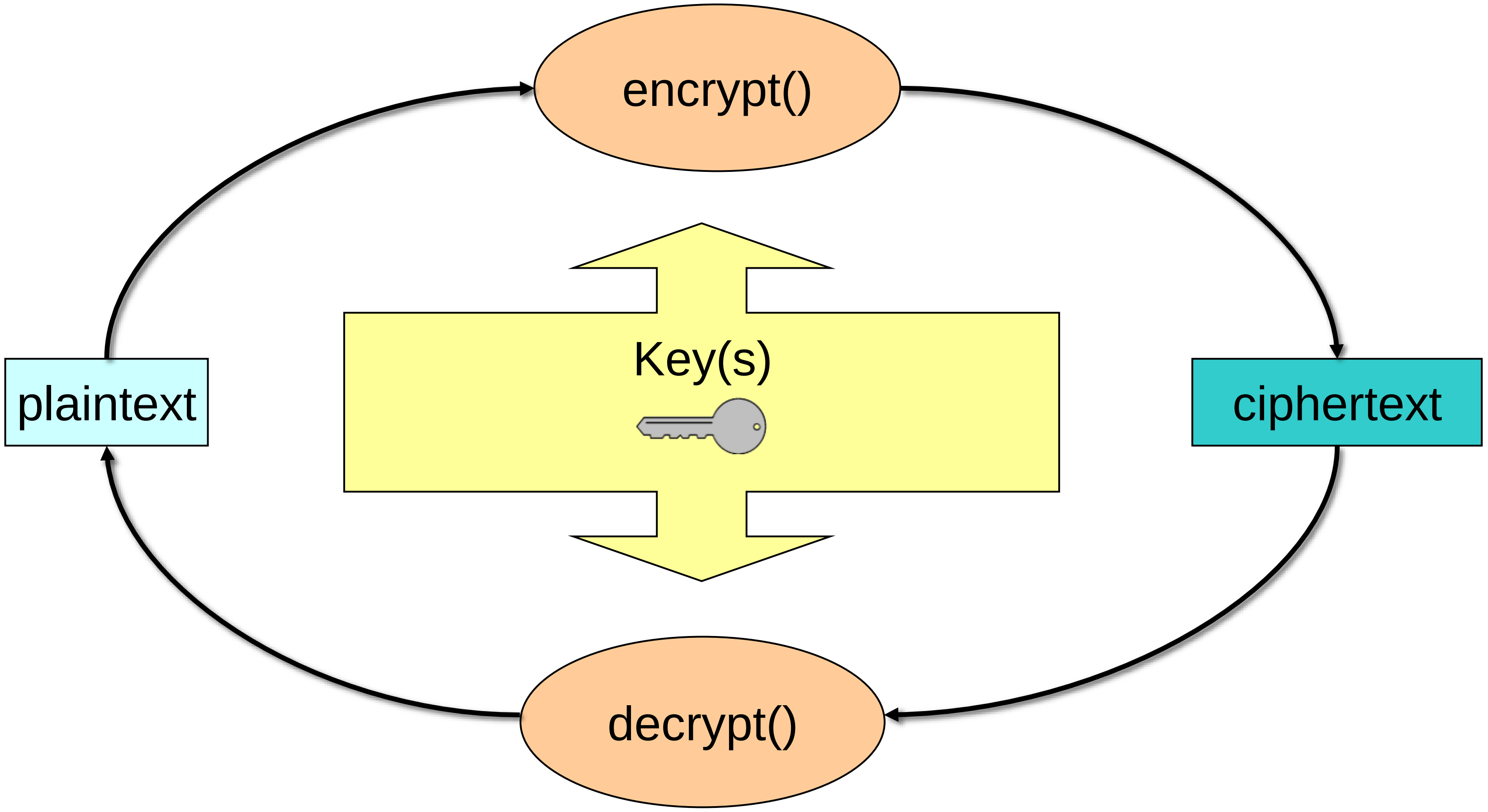
- **Encryption**: plaintext (or cleartext) → ciphertext (or cryptogram)
- **Decryption**: ciphertext → plaintext

- **Algorithm**: way of transforming data

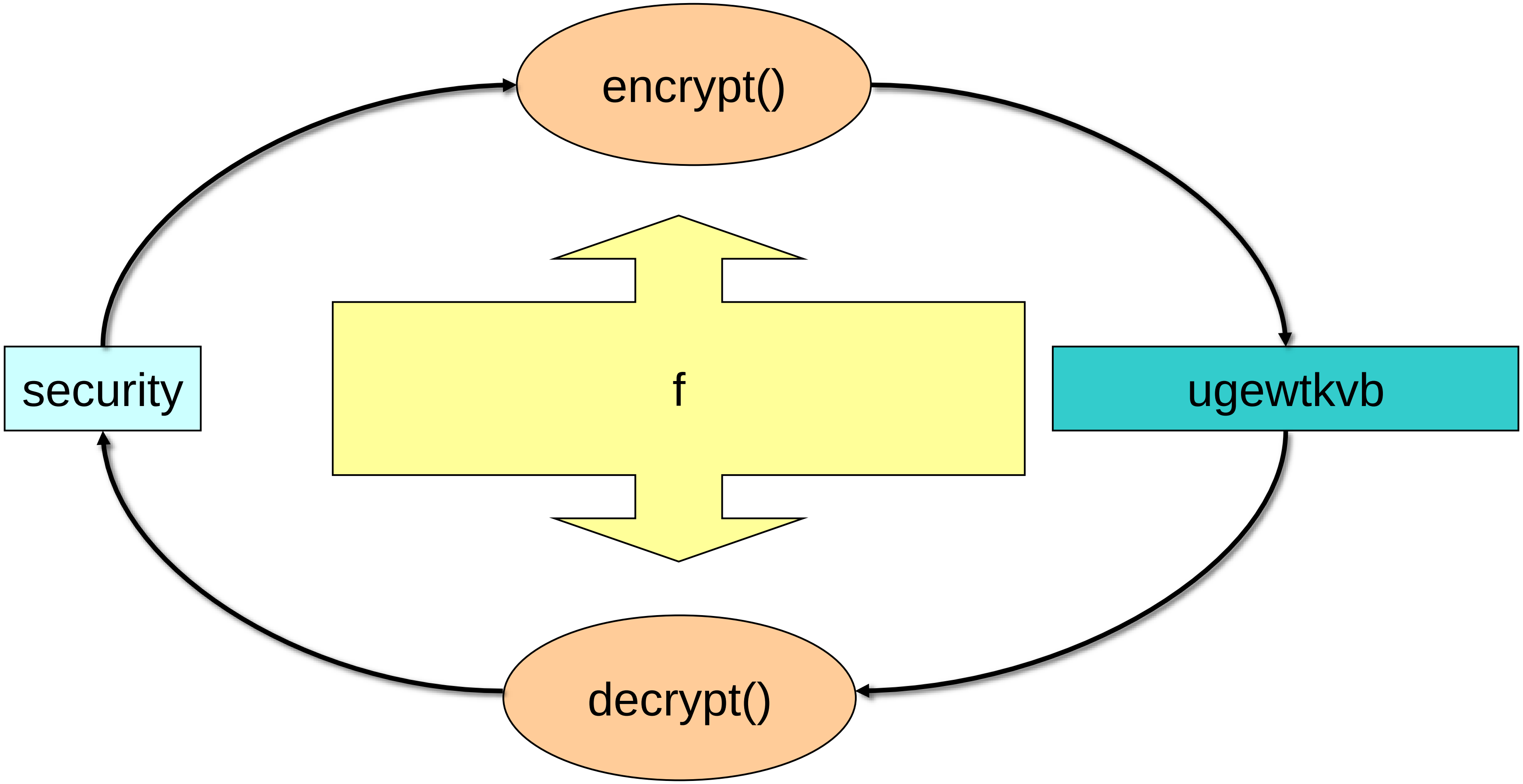
Key: algorithm parameter

- influences algorithm execution

Operations of a Cipher



Operations of a Cipher



Use cases (symmetric ciphers)

Self protection with key K

- Alice encrypts plaintext **P** with key **K** -> `Alice: C = {P}`$_k$
- Alice decrypts cryptogram **C** with key **K** -> `Alice: P'= {C}`$_k$
- **P'** should be equal to **P** (requires checking)

Secure communication with key K

- Alice encrypts plaintext **P** with key **K** -> `Alice: C = {P}`$_k$
- Bob decrypts **C** with key **K** -> `Bob: P' = {C}`$_k$
- **P'** should be equal to **P** (requires checking)

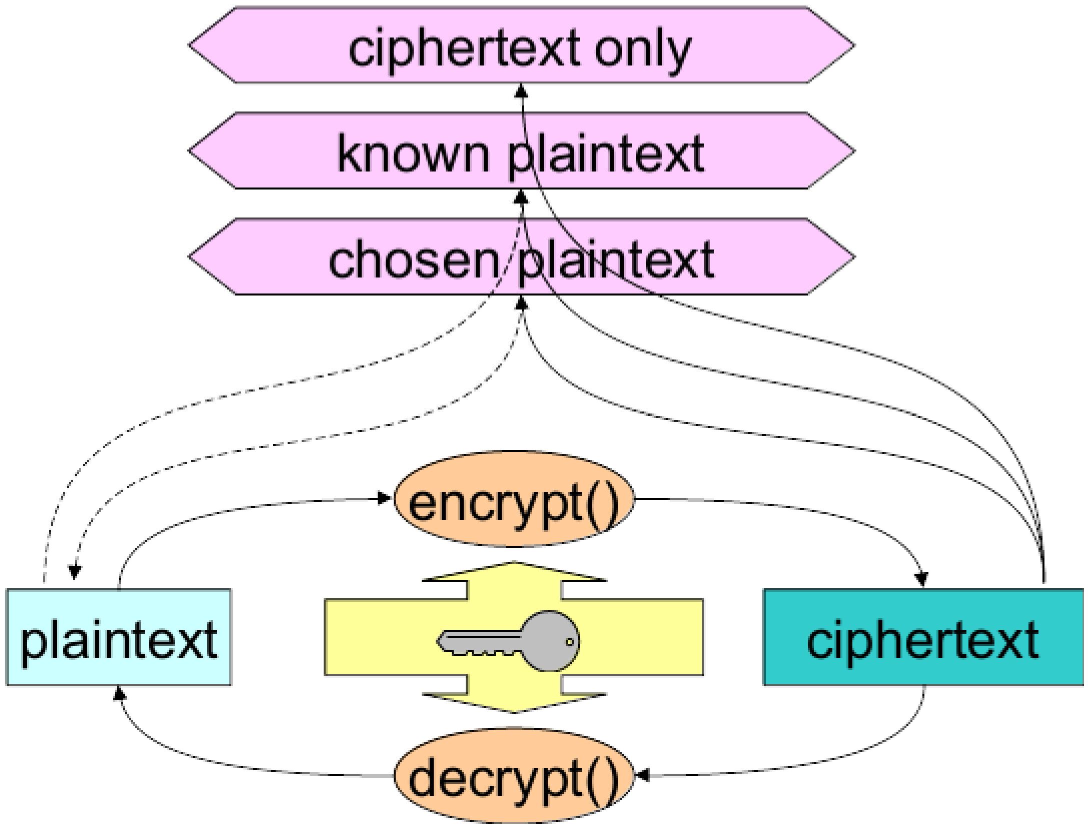
Cryptanalysis: goals

Discover original plaintext
- Which originated a given ciphertext

Discover a cipher key
- Allows the decryption of ciphertexts created with the same key

Discover the cipher algorithm
- Or an equivalent algorithm
- Usually algorithms are not secret, but there are exceptions
  - Lorenz, A5 (GSM), RC4, Crypto-1 (Mifare)
  - Algorithms for DRM (Digital Rights Management)
- Using reverse engineering

Cryptanalysis attacks
Some approaches



Cryptanalysis attacks: Approaches

Brute force

- Exhaustive search of the key space until finding a suitable key
- Usually unfeasible for a large key space
  - e.g. 128 bits keys have a search space of 2^{128} values.
- Randomness is fundamental!

Clever attacks

- Reduce the search space to a smaller set of potential candidates: words, numbers, restricted size or alphabet
- Identify patterns in different operations, etc..

Ciphers: evolution of technology

Manual ciphers

- Substitution or transposition algorithms

Source: Wikimedia Commons e CryptoMuseum

Ciphers: evolution of technology

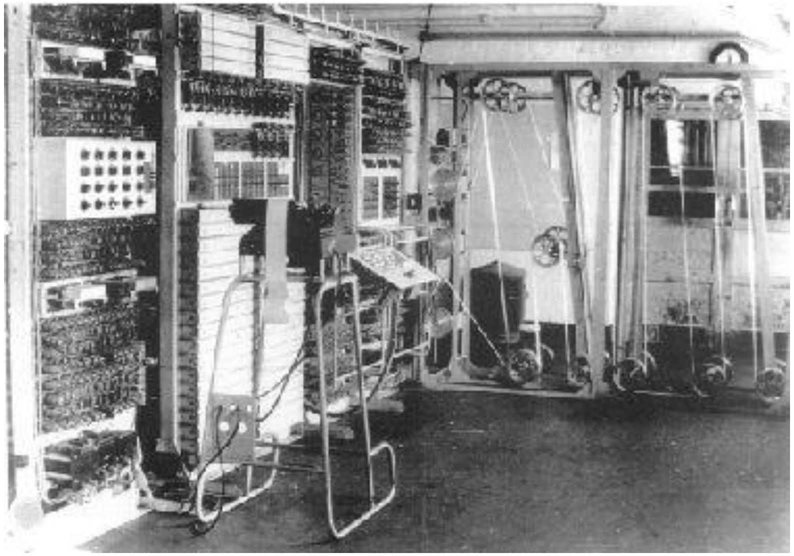
Mechanical ciphers

- Starting from XIX century
  - Enigma Machine
  - M-209 Converter
- More complex substitution algorithms
  - Key devices for the 2nd World War

Ciphers: evolution of technology

Informatic Ciphers

- Appear with the computers
- Using more complex substitution algorithms
- High reliance on mathematically hard problems and large numbers
- Widespread use by most population



Ciphers: basic types (1/4)

Transposition: the plaintext is scrambled:

taxcl hitre eniad ptsm lesb

T	H	E	P	L
A	I	N	T	E
X	T	I	S	S
C	R	A	M	B
L	E	D		

with block permutations (31524):

eniad taxcl lesbh itrep tsm

Ciphers: basic types (2/4)

Substitution

- Each original symbol is replaced by another
- Original symbols were letters, digits and punctuation
- Actually using blocks of bits

Substitution strategies

- Mono-alphabetic (one to one)
- Polyalphabetic (many one to one)
- Homophonic (one to any)

Ciphers: basic types (3/4) monoalphabetic

Use a single substitution alphabet (with #α elements)

Examples

- Additive (translation)
  - crypto-symbol = (symbol + key) mod # α
  - symbol = (crypto-symbol – key) mod # α
  - Possible keys = # α
  - Caesar Cipher (ROT-x)
- With sentence key
  - ABCDEFGHIJKLMNOPQRSTUVWXYZ
    QRUVWXZSENTCKYABDFGHIJLMOP
    - Possible keys = # α! -> 26! $\approx 2^{88}$

Problems

- Reproduce plaintext pattern
  - Individual characters, digrams, trigrams, etc.
- Statistical analysis facilitates cryptanaly
  - "The Gold Bug", Edgar Alan Poe

Ciphers: basic types (3/4) monoalphabetic

Problems

- Reproduce plaintext pattern
  - Individual characters, digrams, trigrams, etc.
- Statistical analysis facilitates cryptanalysis
  - “The Gold Bug”, Edgar Alan Poe

```
a good glass in the
bishop's hostel in the
devil's seat fifty-one
degrees and thirteen
minutes northeast and
by north main branch
seventh limb east side
shoot from the left eye
of the death's-head a
bee line from the tree
through the shot forty
feet out
```

```
53‡‡†305))6*;4826)4‡.)
4‡);806*;48†860))85;1‡
(;:‡*8†83(88)5*†;46(;8
8*96*?;8)*‡(;485);5*†2
:*‡(;4956*2(5*—4)88*;4
069285);)6†8)4‡‡;1(‡9;
48081;8:8‡1;48†85;4)48
5†528806*81(‡9;48;(88;
4(‡?34;48)4‡;161;:188;
‡?;
```

Ciphers: basic types (3/4) monoalphabetic

53‡‡†305))6*;4826)4‡.)4‡);80
agoodglassinthebishopshostel

6*;48†8¶60))85;1‡(;:‡*8†83(88)
inthedevilsseatfortyonedegrees

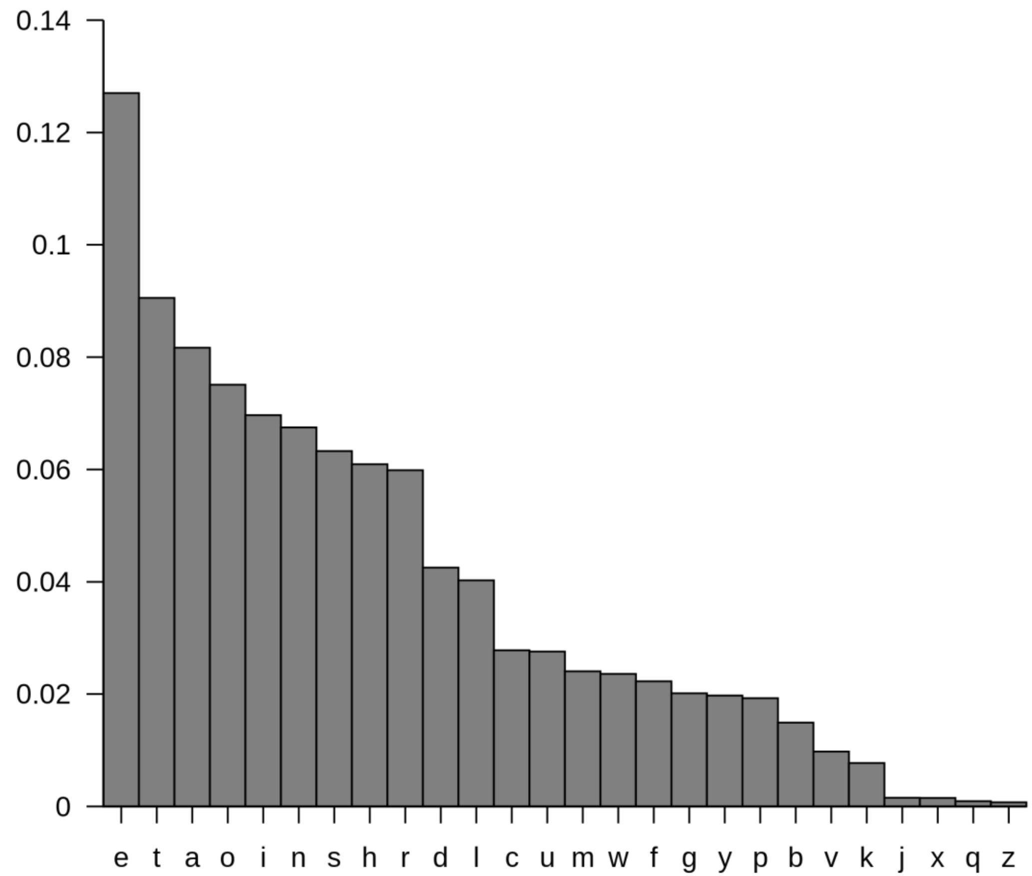
5*†;46(;88*96*?;8)*‡(;485);5*†
andthirteenminutesnortheastand

2:*‡(;4956*2(5*-4)8¶8*;40692
bynorthmainbranchseventhlimb

85);)6†8)4‡‡;1(‡9;48081;8:8‡1
eastsideshootfromthelefteyeof

;48†85;4)485†528806*81(‡9;48
thedeathsheadabeelinefromthe

;(88;4(‡?34;48)4‡;161;:188;‡?;
treethroughtheshotfiftyfeetout



a 5 (12)
b 2 (5)
c - (1)
d † (8)
e 8 (33)
f 1 (8)
g 3 (4)
h 4 (19)
i 6 (11)
j
k
l 0 (6)
m 9 (5)
n * (13)
o ‡ (16)
p . (1)
q
r ((10)
s) (16)
t ; (26)
u ? (3)
v ¶ (2)
w
x
y : (4)
z

Ciphers: basic types (3/4) monoalphabetic

Frequency of Tuples

- NO, TH, TA, OS, AS

Frequency of Triplets

- THE, TOO, THA, YES…

Conditional Probabilities

- P(A | B) will differ from P(Z | B)

Ciphers: basic types (4/4) polyalphabetic

Use N substitution alphabets

- Periodical ciphers, with period N

Example

- Vigenère cipher

Problems

- Once known the period, are as easy to cryptanalyze as **N** mono-alphabetic ones
  - The period can be discovered using statistics
- Kasiski method
  - Factoring of distances between equal ciphertext blocks
- Coincidence index
  - Factoring of self-correlation offsets that yield higher coincidences

Vigenère cipher
(or the Vigenère square)

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

Example of ciphering the letter M with the key S, originating the cryptogram E

Cryptanalysis of a Vigenère cryptogram: Example (1/2)

Plaintext:

Eles não sabem que o sonho é uma constante da vida
tão concreta e definida como outra coisa qualquer,
como esta pedra cinzenta em que me sento e descanso,
como este ribeiro manso, em serenos sobressaltos
como estes pinheiros altos

Cipher with the Vigenère square and key “poema”

- plaintext elesnaosabemqueosonhoeumaconstantedavidataoconcretaedefinida
- key poemapoemapoemapoemapoemapoemapoemapoemapoemapoemapoemapoema
- cryptogram tzienpcwmbtaugedgszhdsyyarcre**tp**bxqdpj**mpa**iosoocqvq**tp**shqfxb**mpa**

Kasiski test

- With text above:
- With the complete poem:

mpa	$20 = 2 \times 2 \times 5$
tp	$20 = 2 \times 2 \times 5$

$175 = 5 \times 5 \times 7$	1
$105 = 3 \times 5 \times 7$	3
$35 = 5 \times 7$	1
$20 = 2 \times 2 \times 5$	4

Cryptanalysis of a Vigenère cryptogram: Example (2/2)

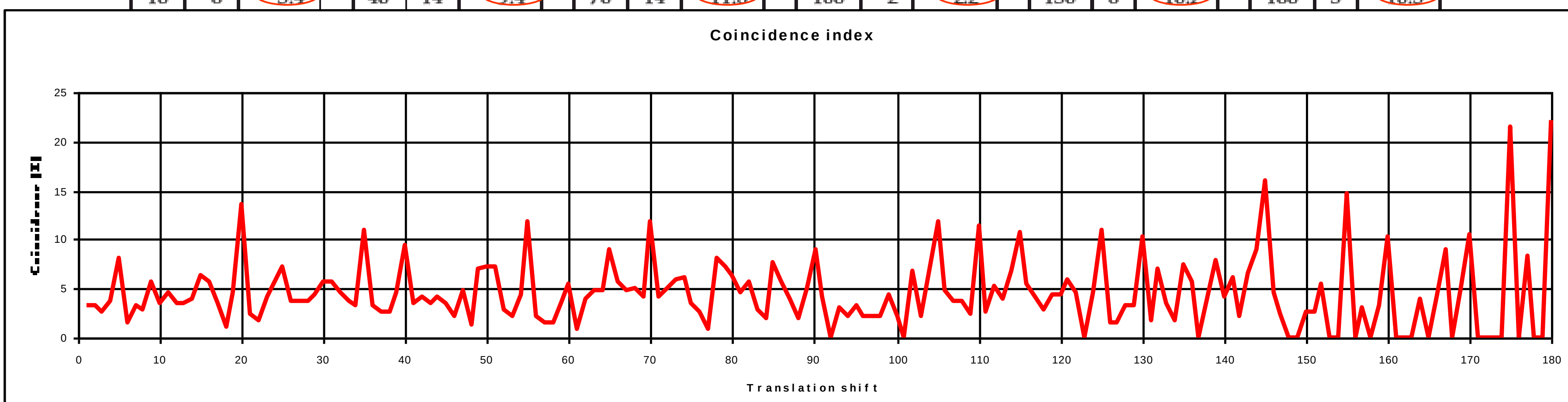
Coincidence index (with full poem)

D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)
1	6	3.2	31	9	5.7	61	1	0.8	91	4	4.1	121	4	5.9	151	1	2.6
2	6	3.2	32	7	4.5	62	5	3.9	92	0	0.0	122	3	4.5	152	2	5.4
3	5	2.7	33	6	3.8	63	6	4.8	93	3	3.1	123	0	0.0	153	0	0.0
4	7	3.8	34	5	3.2	64	6	4.8	94	2	2.1	124	3	4.6	154	0	0.0
5	15	8.2	35	17	11.0	65	11	8.9	95	3	3.2	125	7	10.9	155	5	14.7
6	3	1.6	36	5	3.3	66	7	5.7	96	2	2.2	126	1	1.6	156	0	0.0
7	6	3.3	37	4	2.6	67	6	4.9	97	2	2.2	127	1	1.6	157	1	3.1
8	5	2.8	38	4	2.6	68	6	5.0	98	2	2.2	128	2	3.3	158	0	0.0
9	10	5.6	39	7	4.7	69	5	4.2	99	4	4.4	129	2	3.3	159	1	3.3
10	6	3.4	40	14	9.4	70	14	11.8	100	2	2.2	130	6	10.2	160	3	10.3
11	8	4.5	41	5	3.4	71	5	4.2	101	0	0.0	131	1	1.7	161	0	0.0
12	6	3.4	42	6	4.1	72	6	5.1	102	6	6.9	132	4	7.0	162	0	0.0
13	6	3.4	43	5	3.4	73	7	6.0	103	2	2.3	133	2	3.6	163	0	0.0
14	7	4.0	44	6	4.1	74	7	6.1	104	6	7.1	134	1	1.8	164	1	4.0
15	11	6.3	45	5	3.5	75	4	3.5	105	10	11.9	135	4	7.4	165	0	0.0
16	10	5.8	46	3	2.1	76	3	2.7	106	4	4.8	136	3	5.7	166	1	4.3
17	6	3.5	47	7	4.9	77	1	0.9	107	3	3.7	137	0	0.0	167	2	9.1
18	2	1.2	48	2	1.4	78	9	8.1	108	3	3.7	138	2	3.9	168	0	0.0
19	8	4.7	49	10	7.1	79	8	7.3	109	2	2.5	139	4	8.0	169	1	5.0
20	23	13.6	50	10	7.2	80	7	6.4	110	9	11.4	140	2	4.1	170	2	10.5
21	4	2.4	51	10	7.2	81	5	4.6	111	2	2.6	141	3	6.2	171	0	0.0
22	3	1.8	52	4	2.9	82	6	5.6	112	4	5.2	142	1	2.1	172	0	0.0
23	7	4.2	53	3	2.2	83	3	2.8	113	3	3.9	143	3	6.5	173	0	0.0
24	9	5.5	54	6	4.4	84	2	1.9	114	5	6.7	144	4	8.9	174	0	0.0
25	12	7.3	55	16	11.9	85	8	7.7	115	8	10.8	145	7	15.9	175	3	21.4
26	6	3.7	56	3	2.3	86	6	5.8	116	4	5.5	146	2	4.7	176	0	0.0
27	6	3.7	57	2	1.5	87	4	3.9	117	3	4.2	147	1	2.4	177	1	8.3
28	6	3.7	58	2	1.5	88	2	2.0	118	2	2.8	148	0	0.0	178	0	0.0
29	7	4.4	59	5	3.8	89	5	5.0	119	3	4.3	149	0	0.0	179	0	0.0
30	9	5.7	60	7	5.4	90	9	9.1	120	3	4.3	150	1	2.6	180	2	22.2

Cryptanalysis of a Vigenère cryptogram: Example (2/2)

Coincidence index (with full poem)

D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)	D	I	P (%)
1	6	3.2	31	9	5.7	61	1	0.8	91	4	4.1	121	4	5.9	151	1	2.6
2	6	3.2	32	7	4.5	62	5	3.9	92	0	0.0	122	3	4.5	152	2	5.4
3	5	2.7	33	6	3.8	63	6	4.8	93	3	3.1	123	0	0.0	153	0	0.0
4	7	3.8	34	5	3.2	64	6	4.8	94	2	2.1	124	3	4.6	154	0	0.0
5	15	8.2	35	17	11.0	65	11	8.9	95	3	3.2	125	7	10.9	155	5	14.7
6	3	1.6	36	5	3.3	66	7	5.7	96	2	2.2	126	1	1.6	156	0	0.0
7	6	3.3	37	4	2.6	67	6	4.9	97	2	2.2	127	1	1.6	157	1	3.1
8	5	2.8	38	4	2.6	68	6	5.0	98	2	2.2	128	2	3.3	158	0	0.0
9	10	5.6	39	7	4.7	69	5	4.2	99	4	4.4	129	2	3.3	159	1	3.3
10	6	3.4	40	14	9.4	70	14	11.8	100	2	2.2	130	6	10.2	160	3	10.3
29	7	4.4	59	5	3.8	89	5	5.0	119	3	4.3	149	0	0.0	179	0	0.0
30	9	5.7	60	7	5.4	90	9	9.1	120	3	4.3	150	1	2.6	180	2	22.2



Rotor Machines (1/3)

David J Morgan, www.flickr.com

Rotor Machines (2/3)

Rotor machines implement complex polyalphabetic ciphers

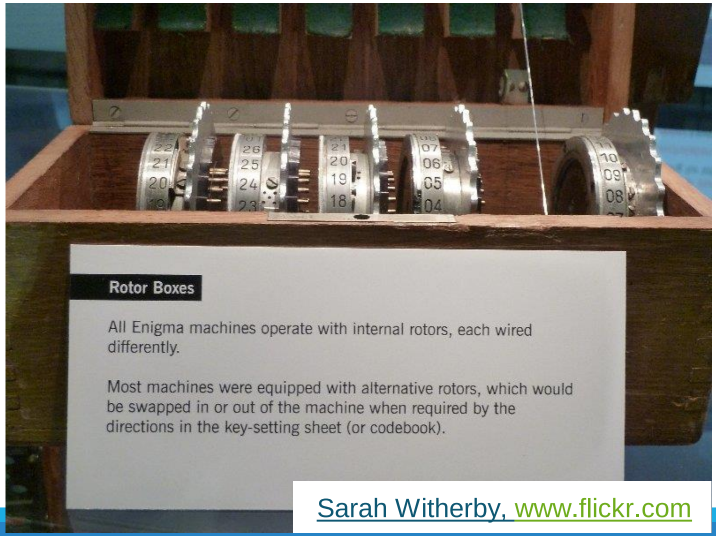
- Each rotor contains a permutation
  - Same as a set of substitutions
- The position of a rotor implements a substitution alphabet
- Spinning of a rotor implements a polyalphabetic cipher
- Stacking several rotors and spinning them at different times adds complexity to the cipher

The cipher key is:

- The set of rotors used
- The relative order of the rotors
- The position of the spinning ring
- The original position of all the rotors

Symmetrical (two-way) rotors allow decryption by "double encryption"

- Using a reflection disk (half-rotor)

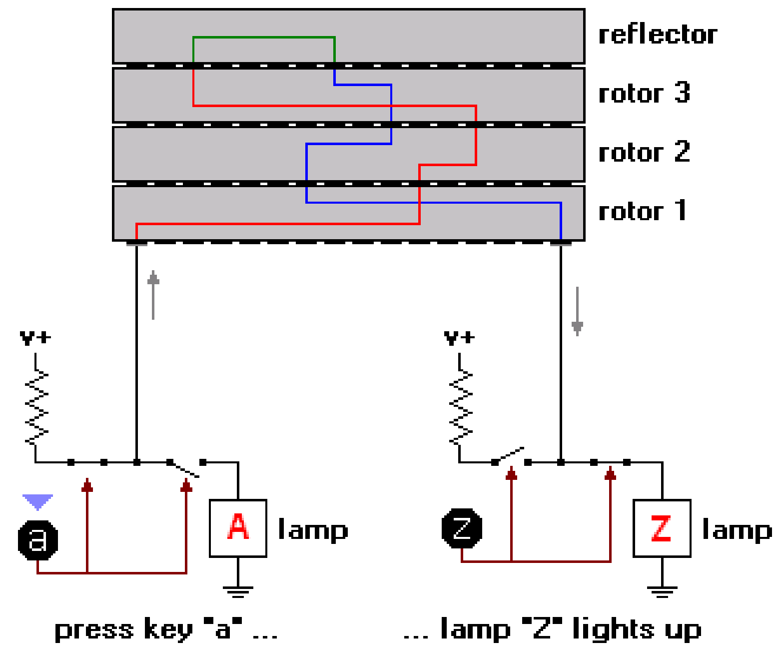


Sarah Witherby, www.flickr.com

Rotor Machines (3/3)

Reciprocal operation with reflector

- Sending operator types “A” as plaintext and gets “Z” as ciphertext, which is transmitted
- Receiving operator types the received “Z” and gets the plaintext “A”
- No letter could encrypt to itself !



RECIPROCAL OPERATION OF THE ENIGMA

Enigma

WWII German rotor machine

Initially presented in 1919

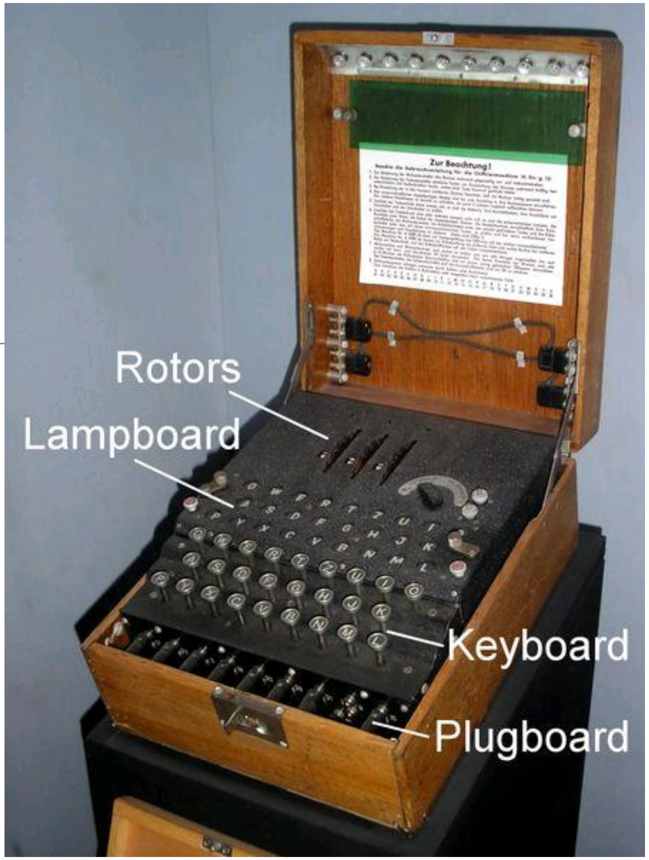
- Enigma I, with 3 rotors

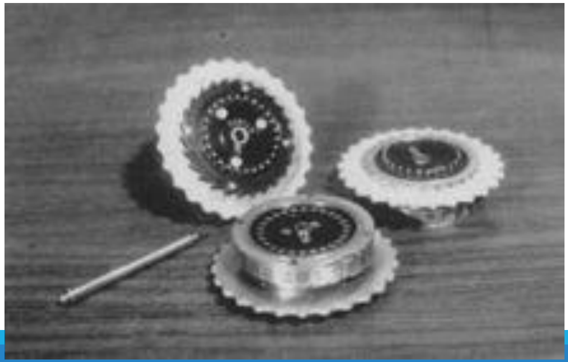
Several variants where used

- With different number of rotors
- With patch cord to permute alphabets

Key settings distributed in codebooks

https://observablehq.com/@tmcw/enigma-machine





Cryptography: theoretical analysis

Plaintext space

- Possible plaintext values (M)

Ciphertext space

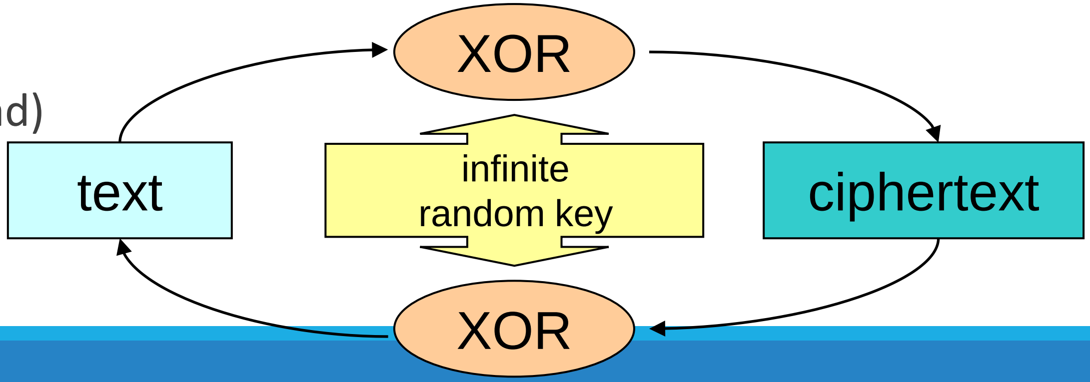
- Possible ciphertext values (C)

Key space

- Possible key values for a given algorithm (K)

Perfect (information-theoretical) security

- Given $c_j \in C$, $p(m_i, k_j) = p(m_i)$
- $\#K \geq \#M$
- Vernam cipher (one-time pad)



Cryptography: practical approaches (1/4)

Theoretical security vs. practical security

- Expected use != practical exploitation
- Defective practices can introduce vulnerabilities
  - Example: re-use of one-time pad key blocks

Computational security

- Security is measured by the computational complexity of break-in attacks
  - Using brute force
- Security bounds:
  - Cost of cryptanalysis
  - Availability of cryptanalysis infra-structure
  - Lifetime of ciphertext

Cryptography: practical approaches (2/4)

5 Shannon Criteria

1. The amount of offered secrecy
   e.g. key length
2. Complexity of key selection
   e.g. key generation, detection of weak keys
3. Implementation simplicity
4. Error propagation
   Relevant in error-prone environments
   e.g. noisy communication channels
5. Dimension of ciphertexts
   Regarding the related plaintexts

Cryptography: practical approaches (3/4)

Confusion: Complex relationship between the key, plaintext and the ciphertext

- Output bits (ciphertext) should depend on the input bits (plaintext + key) in a very complex way

Diffusion: Plaintext statistics are dissipated in the ciphertext

- If one plaintext bit toggles, then the ciphertext changes substantially, in an unpredictable or pseudorandom manner
- **Avalanche effect**

Cryptography: practical approaches (4/4)

Always assume the worst case

Cryptanalysts know the algorithm
- Security lies in the key

Cryptanalysts know/have many cryptogram samples produced with the same algorithm & key
- Cryptograms are not secret!

Cryptanalysts partially (or fully) knows original plaintexts
- As they have some idea of what they are looking for
- Know-plaintext attacks
- Chosen-plaintext attacks

Cryptographic robustness

The robustness of algorithms is their resistance to attacks

- No one can evaluate it precisely
  - Only speculate or demonstrate using some other robustness assumptions
- They are robust until someone breaks them
- There are public guidelines with what should/must not be used
  - Sometimes anticipating future problems

Public algorithms without known attacks are likely to be more robust

- More people looking for weaknesses

Algorithms with longer keys are likely to be more robust

- And usually slower …

Cryptographic robustness
Example: AES selection timeline

AES: Advanced Encryption Standard

1997: NIST launches a challenge for the next AES

- public knowledge and rights, symmetric, keys of 128, 192 ans 256 bits

1998: 15 candidates presented by researchers

- CAST-256, Crypton, DEAL, DFC, Frog, HPC, LOKI97, Magenta, MARS, RC6, Rijndael, Safer+, Serpent, Twofish
- Entire community tried to find problems in the candidates

1999: 5 proposals stayed secure

- MARS, RC6, Rijndael, Twofish
- Entire community tried to find problems, and to evaluate the performance

2001: Rijndael selected as the winner

- MARS reduced versions are broken, RC6 and Twofish are still secure

2002: Published as a FIPS PUB 197 and widely used

Stream Ciphers (1/2)

Mixture of a keystream with the plaintext or ciphertext

- Random keystream (Vernam's one-time pad)
- Pseudo-random keystream (produced by generator using a finite key)

Reversible mixture function

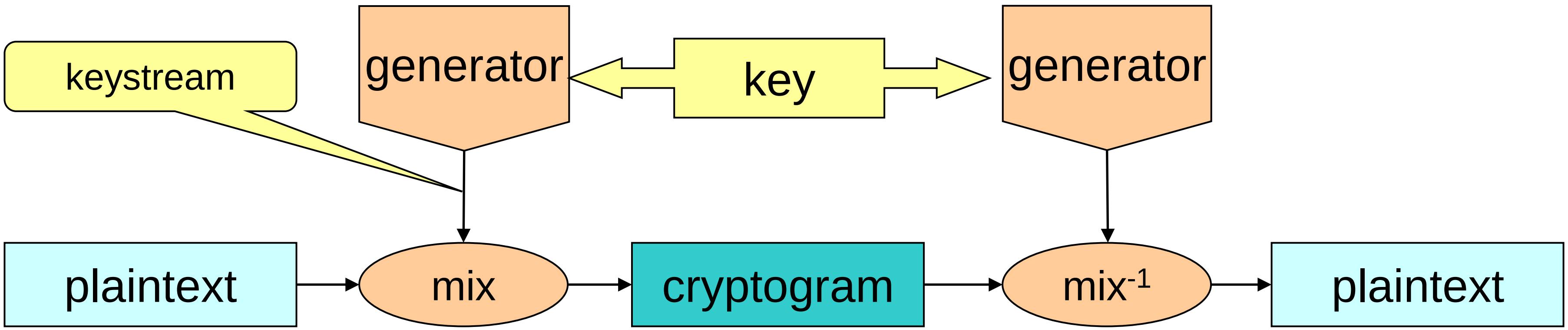
- e.g. bitwise XOR

$$C = P \oplus ks \quad P = C \oplus ks$$

Polyalphabetic cipher

- Each keystream symbol defines an alphabet

Stream Ciphers (1/2)



Stream Ciphers (2/2)

Keystream may be infinite but with a finite period

- The period depends on the generator

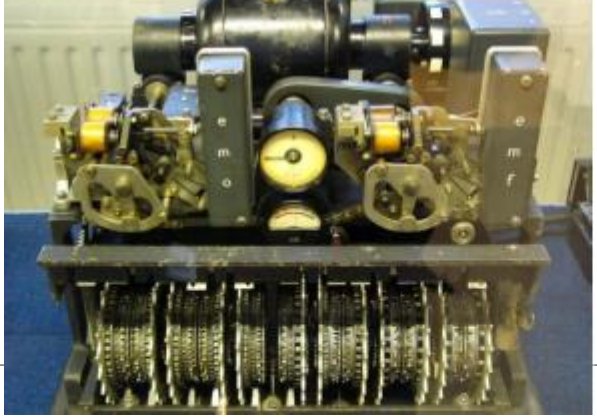
Practical security issues

- Each keystream should be used **only once**!
  - Otherwise, the sum of cryptograms yields the sum of plaintexts

$$C1 = P1 \oplus Ks,\ \ C2 = P2 \oplus Ks \ \rightarrow\ C1 \oplus C2 = P1 \oplus P2$$

- Plaintext length should be **smaller** than the keystream period
  - Keystream exposure is **total under known/chosen** plaintext attacks
  - Keystream cycles help cryptanalysts knowing plaintext samples
- Integrity control is **mandatory**
  - No diffusion! (only confusion)
  - Ciphertexts can easily be changed deterministically

Lorenz (Tunny)

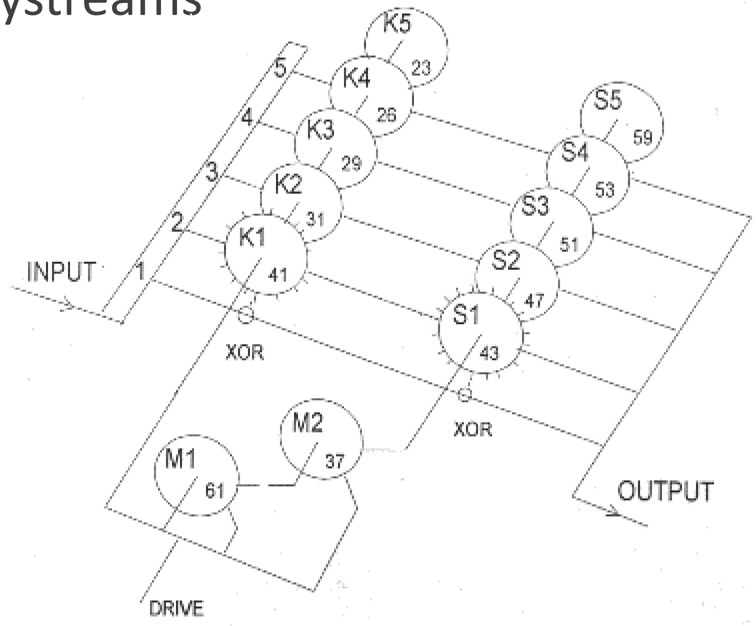


12-Rotor stream cipher

- Used by the German high-command during the 2nd WW
- Implements a stream cipher
- Each 5-bit character is mixed with 5 keystreams

Operation

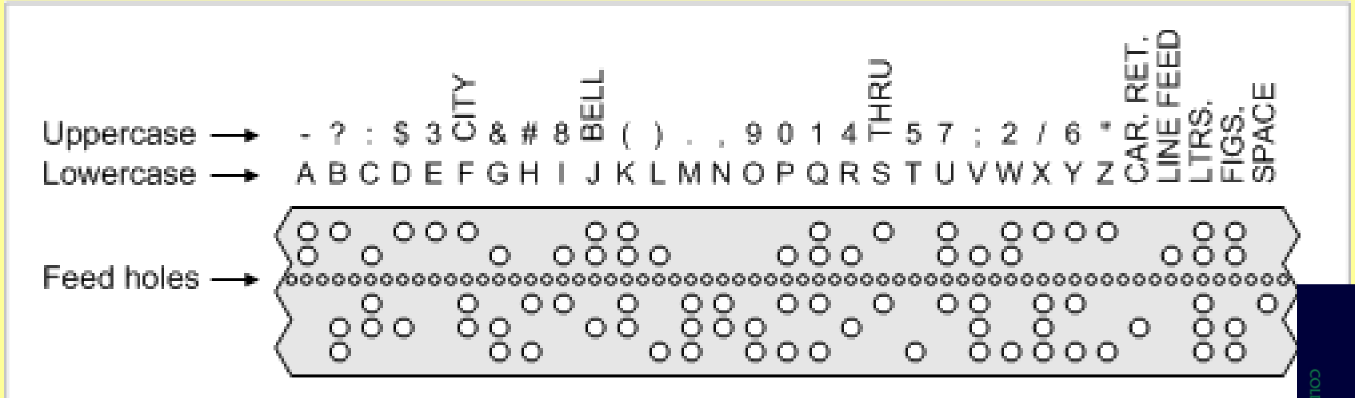
- 5 regularly stepped (χ) wheels
- 5 irregularly stepped (ψ) wheels
  - All or none stepping
- 2 motor wheels
  - For stepping the ψ wheels
- Number of steps in all wheels is relatively prime



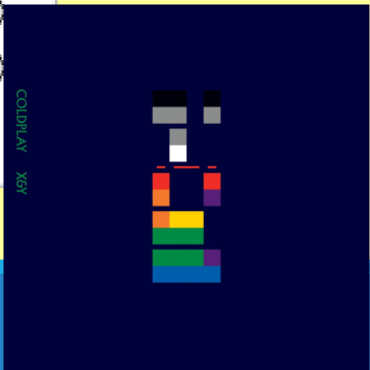
Cryptanalysis of Tunny in Bletchley Park (1/5)

They didn’t know Lorenz internal structure

- They observed one only at the end of the war
- They knew about them because they could get 5-bit encrypted transmissions
  - Using the 32-symbol Baudot code instead of Morse code



Paper tape showing the five-bit Baudot Code



Cryptanalysis of Tunny in Bletchley Park (2/5)

The mistake (30 August 1941)

- A German operator had a long message (~4,000) to send
- He set up his Lorenz and sent a 12 letter indicator (wheel setup) to the receiver
- After ~4,000 characters had been keyed, by hand, the receiver said "send it again“

The operator resets the machine to the same initial setup

- Same keystream! Absolutely forbidden!

The sender began to key in the message again (by hand)

- But he typed a **slightly different** message!

Cryptanalysis of Tunny in Bletchley Park (3/5)

$C0 = M0 \oplus Ks$

$C1 = M1 \oplus Ks$

$M1 = C0 \oplus C1 \oplus M0$ -> text variations

If you know part of the initial text (M0), you can find the variations

Cryptanalysis of Tunny in Bletchley Park (4/5)

Breakthrough

- Message began with a well known SPRUCHNUMMER — "msg number".
  - The first time the operator keyed in **S P R U C H N U M M E R**
  - The second time he keyed in **S P R U C H N R**
  - Thus, immediately following the **N** the two texts were different!

Both messages were sent to John Tiltman at Bletchley Park, which was able to fully decrypt them using an additive combination of the messages (Depths)

- The 2nd message was ~500 characters shorter than the first one
- Tiltman managed to discover the correct message for the 1st ciphertext

They got for the 1st time a long stretch of the Lorenz keystream

- They did not know how the machine did it, …
- … but he knew that this was what it was generating!

Cryptanalysis of Tunny in Bletchley Park (5/5): Colossus

The cipher structure was determined from the keystream

- But deciphering it required knowing the initial position of rotors

Germans started using numbers for the initial wheels' state

- Bill Tutte invented the double-delta method for finding that state
- The Colossus was built to apply the double-delta method

Colossus

- Design started in March 1943
- The 1,500 valve Colossus Mark 1 was operational in January 1944
- Colossus reduced the time to break Lorenz from weeks to hours

The Imitation Game, 2014, "describing" some activities at Bletchley Park

Modern ciphers: types

Concerning operation

- Block ciphers (mono-alphabetic)
- Stream ciphers (polyalphabetic)

Concerning their key

- Symmetric ciphers (secret key or shared key ciphers)
- Asymmetric ciphers (or public key ciphers)

Arrangements

	Block ciphers	Stream ciphers
Symmetric ciphers		
Asymmetric ciphers		DO NOT EXIST

Symmetric ciphers

Single secret key, shared by 2 or more peers

Allow

- Confidentiality among the key holders
- Limited authentication of messages
  - When block ciphers are used

Advantages

- Performance (usually very efficient)

Disadvantages

- N interacting peers, pairwise secrecy -> N x (N-1)/2 keys

Problems

- Key distribution

Symmetric block ciphers

Usual approaches

- Large bit blocks usually greater than 128 bits

Diffusion & confusion

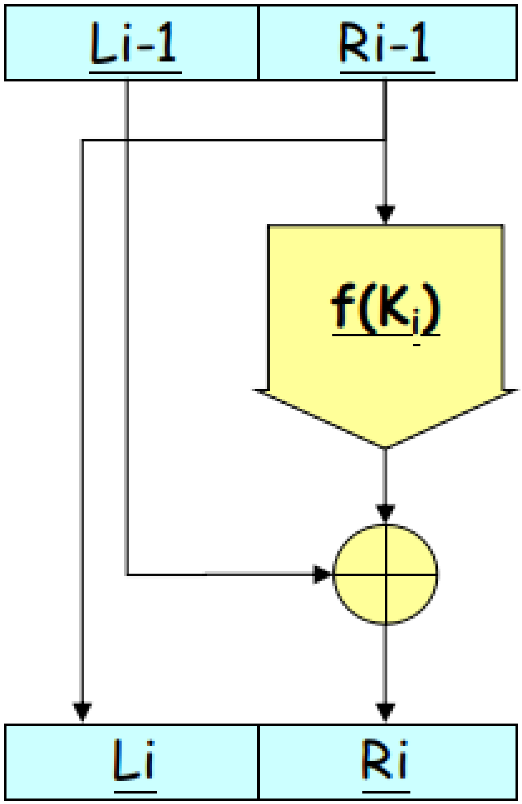
- Permutation, substitution, expansion, compression
- Feistel Networks with multiple iterations
  - $L_i = R_{i-1}$ $R_i = L_{i-1}\ f(R_{i-1} \oplus, K_i)$
- Or substitution-permutation networks

Most common algorithms

- DES (Data Enc. Stand.), D=64; K=56
- AES (Adv. Enc. Stand., aka Rijndael), D=128, K=128, 192, 256
- Other (Blowfish, CAST, RC5, etc.)

Feistel Network

$L_i = R_{i-1} \qquad R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$



Substitution Permutation Network

S-Box: Substitution - based on input, switches bits in the output

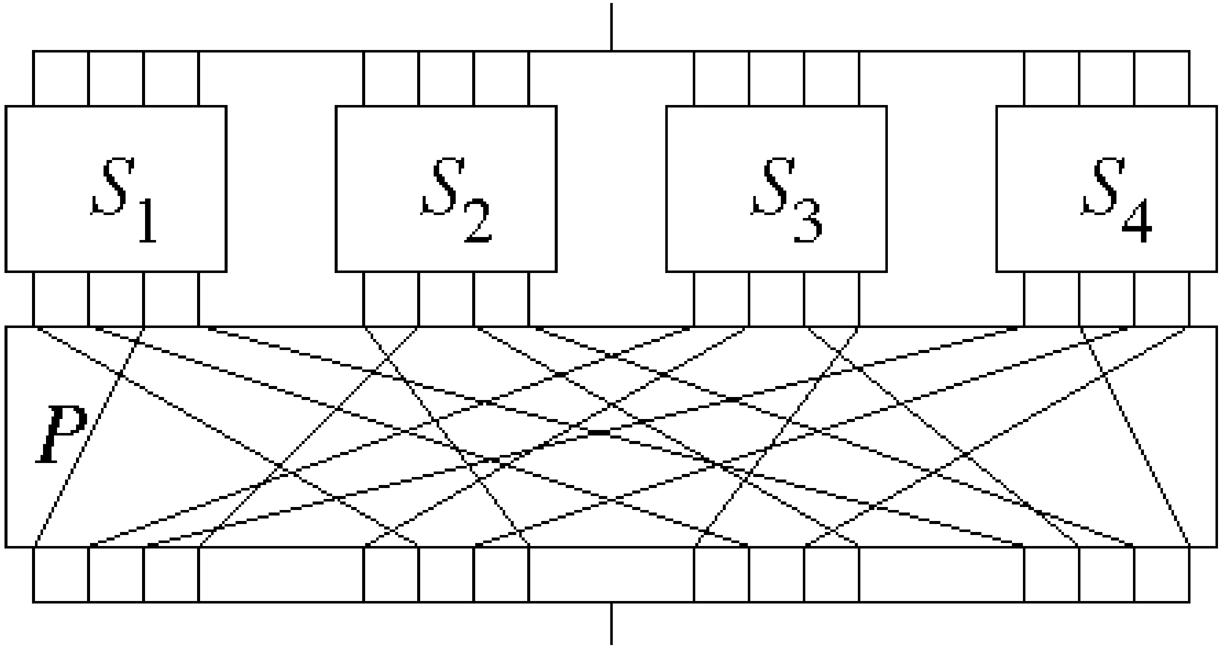
- not a 1 to 1 substitution
- ideal: all output bits depend on all input bits
- practical: at least half the output bits depend on a single input bit

P-Box: Permutation - permutates input bits to output bits

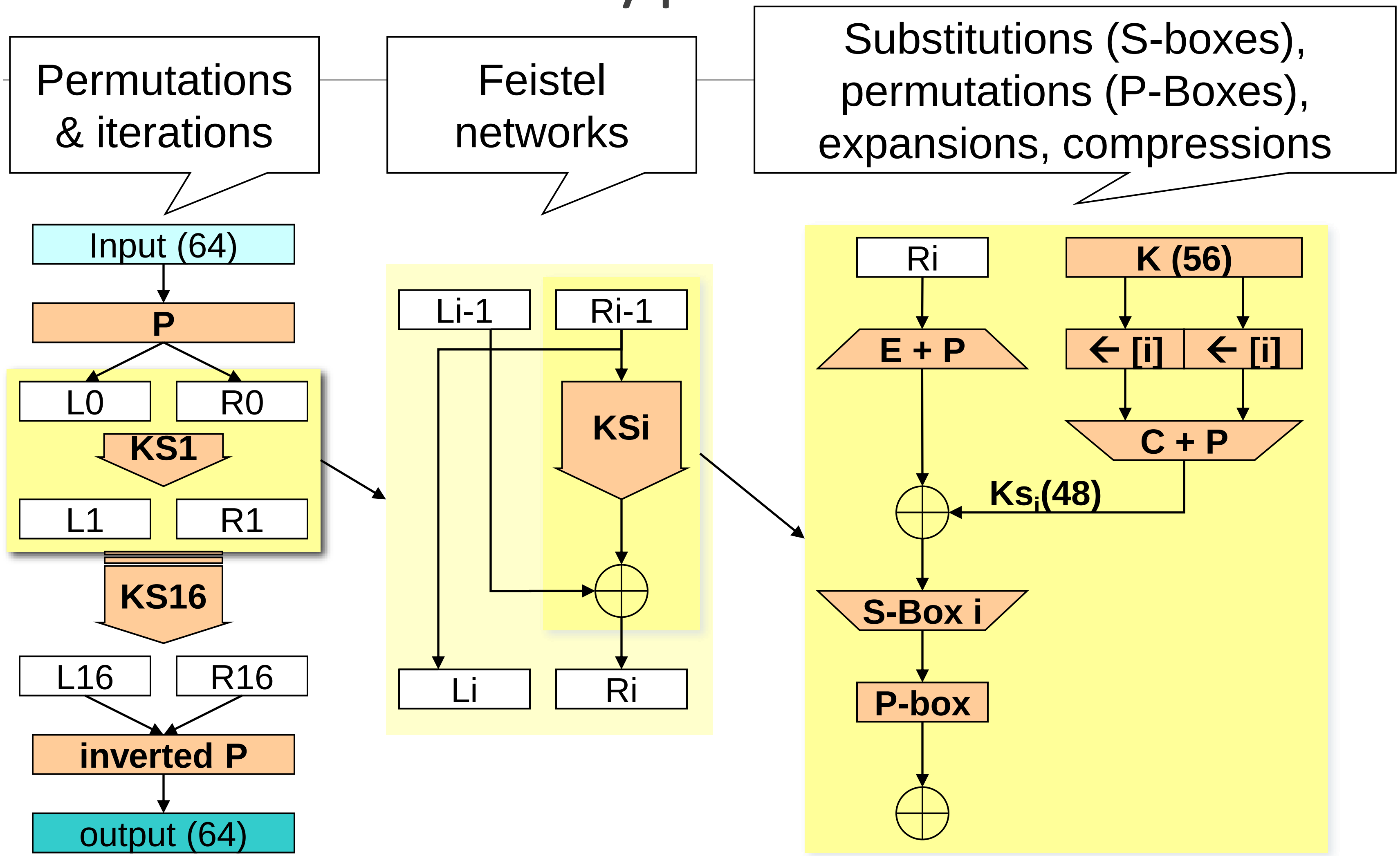
- ideal implementations permute all bits

Operation of both depends on the key

Substitution Permutation Network



DES: Data Encryption Standard



DES: security strength

Key selection

- Most 56 bit values are suitable keys
- 4 weak, 12 semi-weak keys, 48 possibly weak keys
  - Produce equal key schedules (one Ks, two Ks or four Ks)
  - Easy to spot and avoid

Known attacks

- Exhaustive key space search (practical with 56bits keys)

Solution: multiple encryption

- Double encryption is not (theoretically) more secure
- Triple encryption: 3DES (Triple-DES) or DES-EDE
  - With 2 or 3 keys
  - Equivalent key length of 112 or 168 bits
  - By using the same key, the 3DES is compatible with standard DES

(Symmetric) stream ciphers

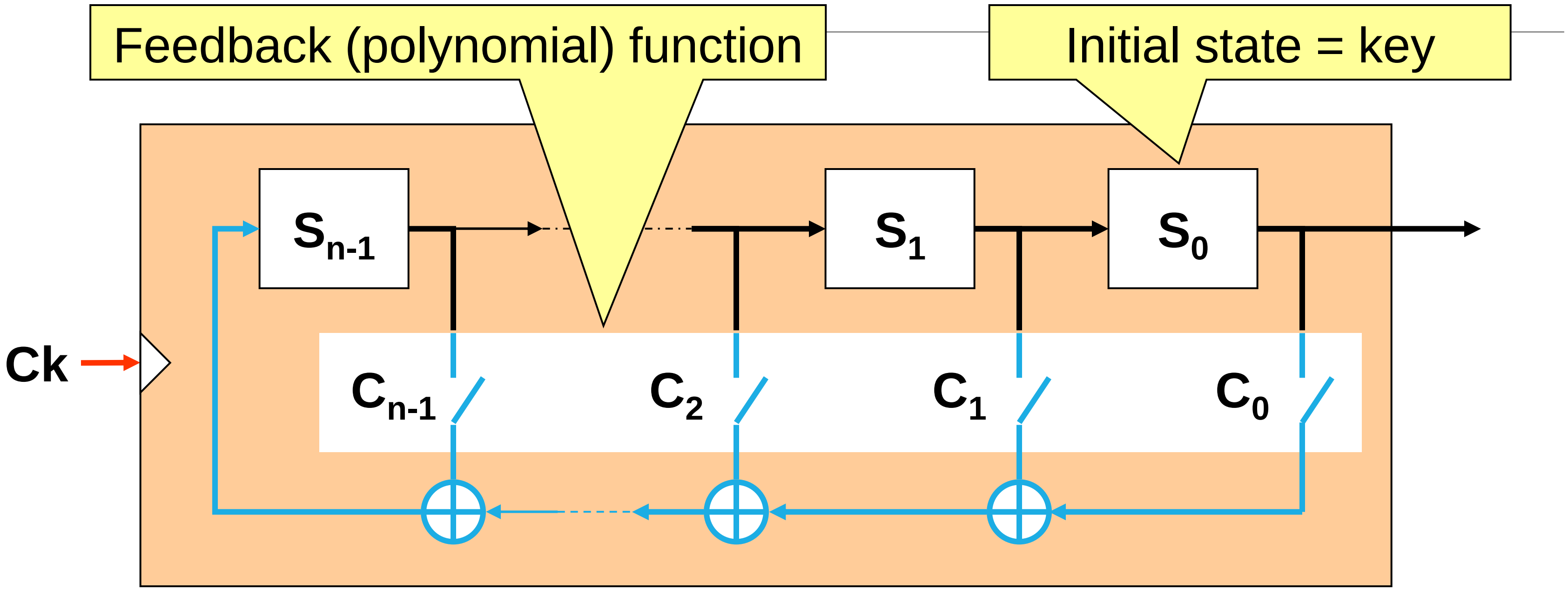
Approaches

- Cryptographically secure pseudo-random generators (PRNG)
  - Using linear feedback shift registers (LFSR)
  - Using block ciphers
  - Other (families of functions, etc.)
- Usually not self-synchronized
- Usually without uniform random access

Most common algorithms

- A5/1 (US, Europe), A5/2 (GSM)
- RC4 (802.11 WEP/TKIP, etc.)
- E0 (Bluetooth BR/EDR)
- SEAL (w/ uniform random access)
- Chacha20
- Salsa20

Linear Feedback Shift Register (LFSR)



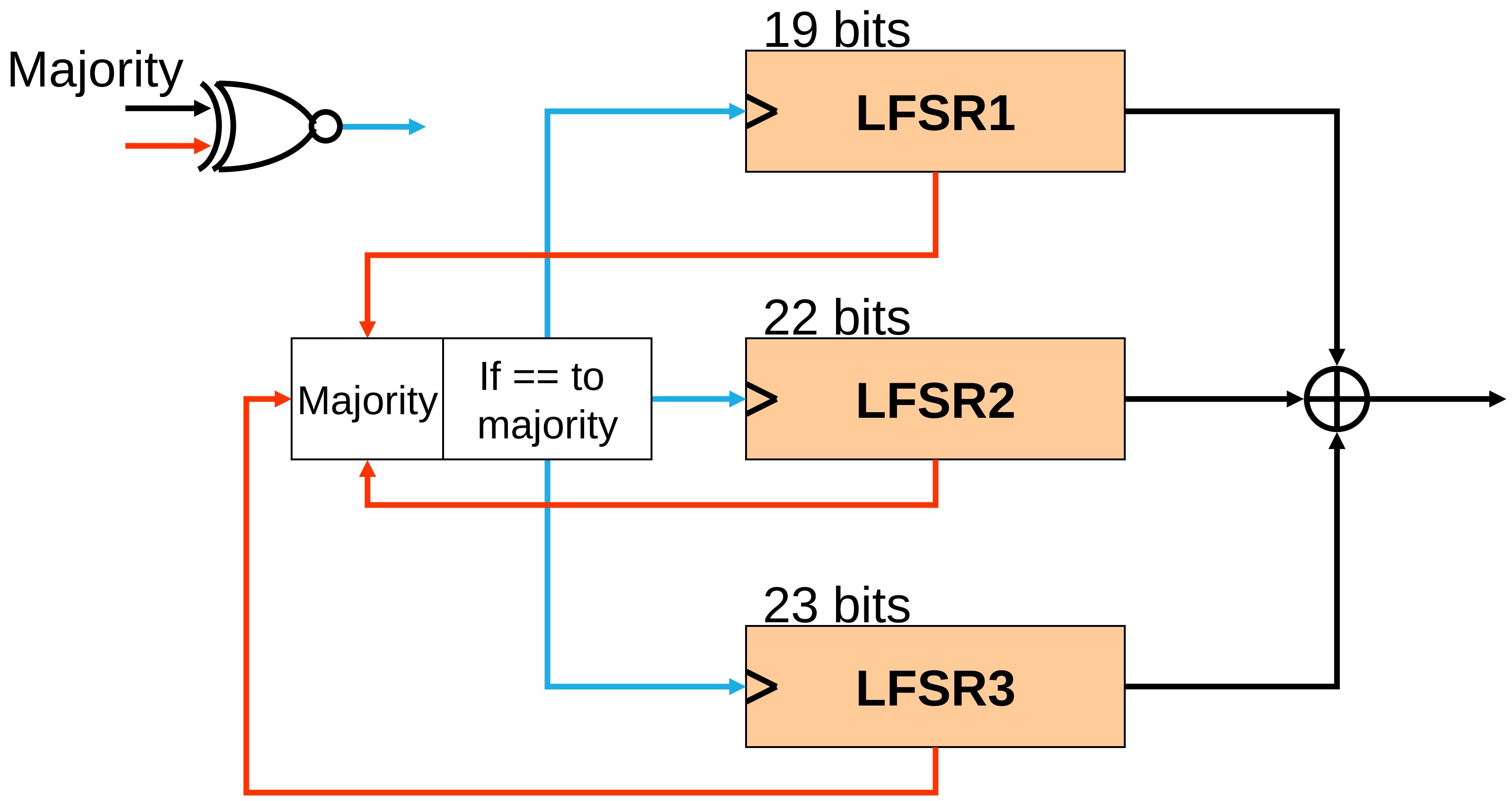
2^n-1 non-null sequences

- If one of them has a 2^n-1 period length, then they all have it

Primitive feedback functions (primitive polynomials)

- All non-null sequences have a 2^n–1 period length

Generators using many LFSR: A5/1 (GSM)



Symmetric Block Ciphers

Process text in blocks

- Text must be multiple of the blocksize
- In practice: size(cryptogram) >= size(plaintext)

Can apply both confusion and diffusion

- Inside the block
- ... but can be used as a stream cipher

Most common encryption methods

- Especially when dealing with discrete objects (files, documents, data chunks)

Most popular cipher: AES

Deployment of (symmetric) block ciphers: Cipher modes

Initially proposed for DES

- ECB (Electronic Code Book)
- CBC (Cipher Block Chaining)
- OFB (Output Feedback Mode)
- CFB (Cipher Feedback Mode)

Can be used with other block ciphers

- In principle ...

Some other modes do exist

- CTR (Counter Mode)
- GCM (Galois/Counter Mode)
- Tweaks

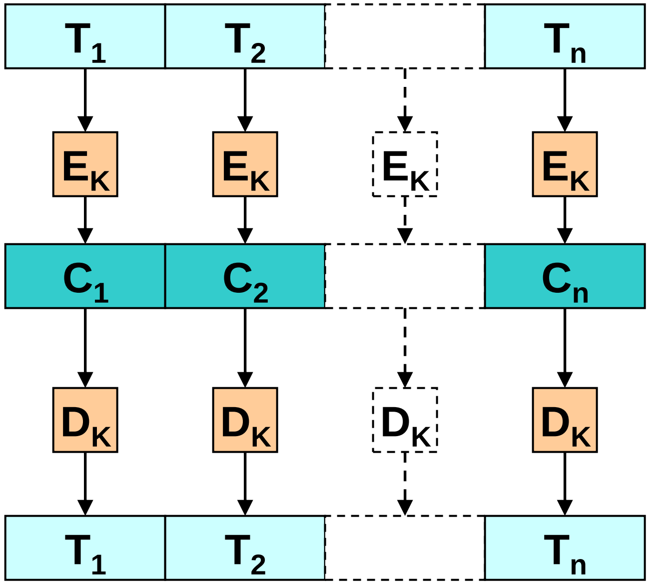
Cipher Modes: Electronic Code Book (ECB)

Direct encryption of each block: $C_i = E_K(T_i)$

Direct decryption of each block: $T_i = D_K(C_i)$

Blocks are processed independently

- No Feedback mechanisms



Problem:

If $T_1 = T_2$ then $C_1 = C_2$

Cipher Modes: Cipher Block Chaining (CBC)

Encrypt each block Ti with feeback from Ci-1

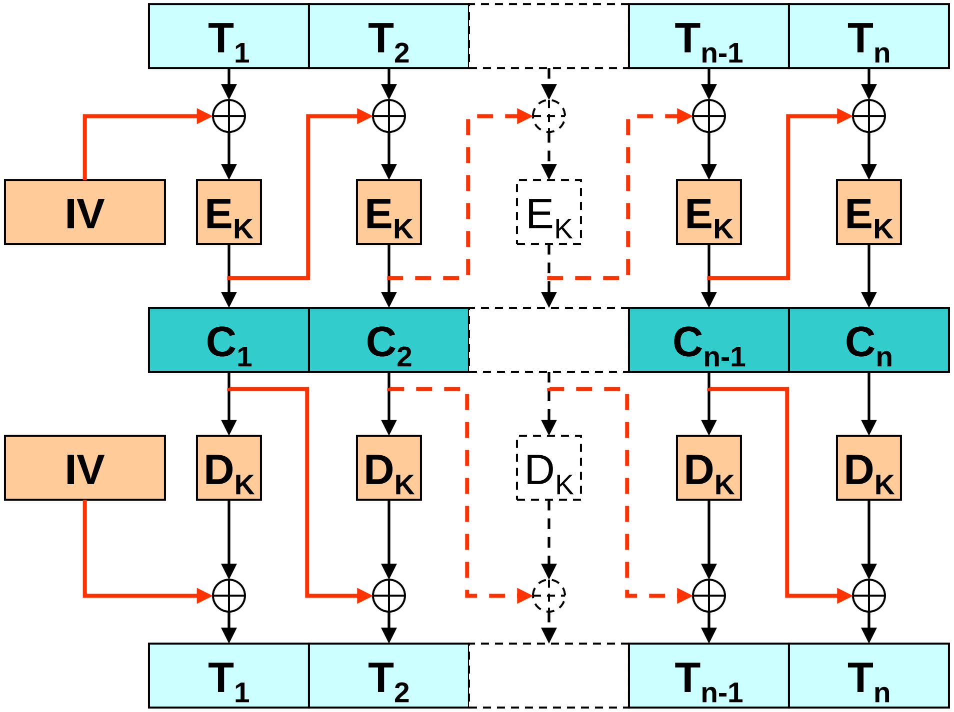
- $C_i = E_K(T_i \oplus C_{i-1})$

Decrypt each block Ci with feedback from Ci-1

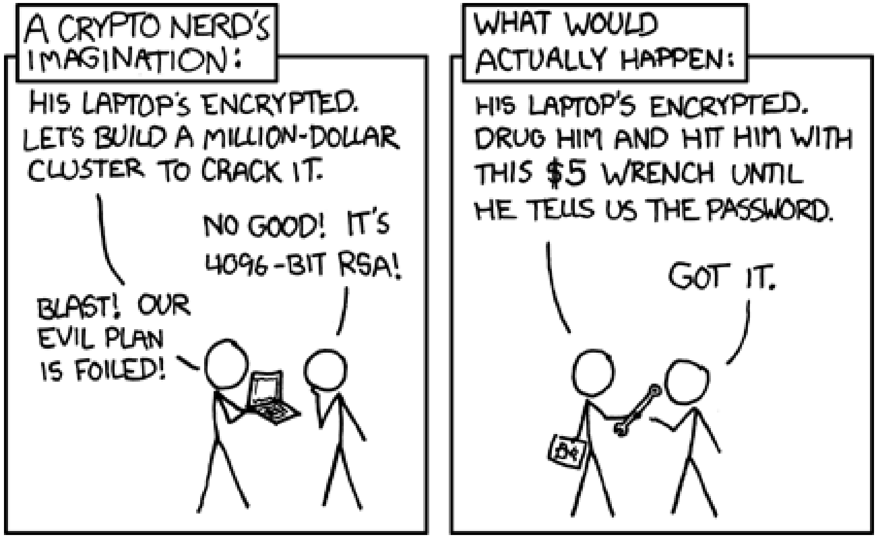
- Decryption: $T_i = D_K(C_i) \oplus C_{i-1}$

First block uses an IV

- IV: Initialization Vector
- Random value
- Never reused for the same key
- May be sent in clear

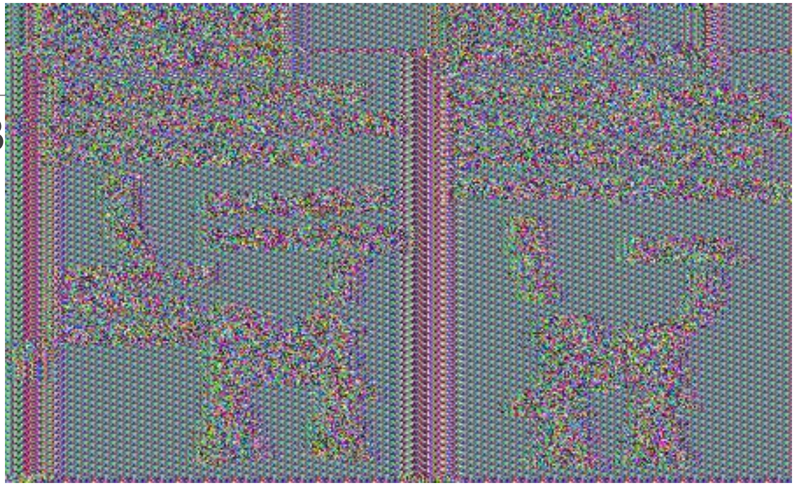


ECB vs CBC: Pattern propagation

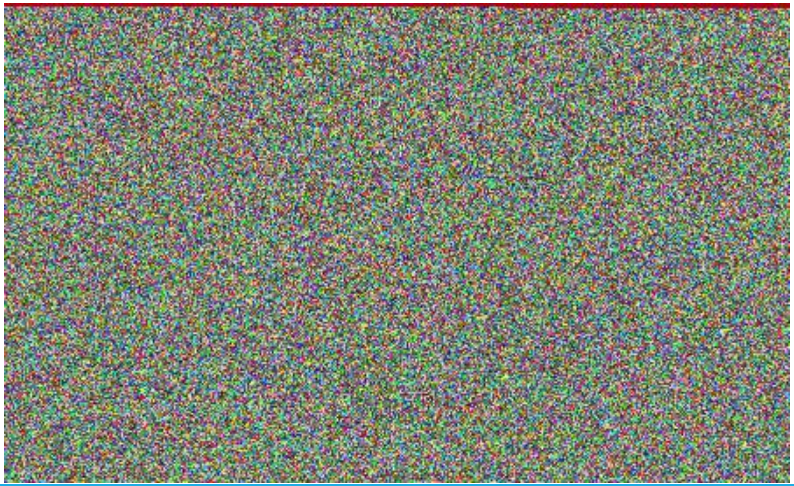


https://xkcd.com/538/

ECB



CBC



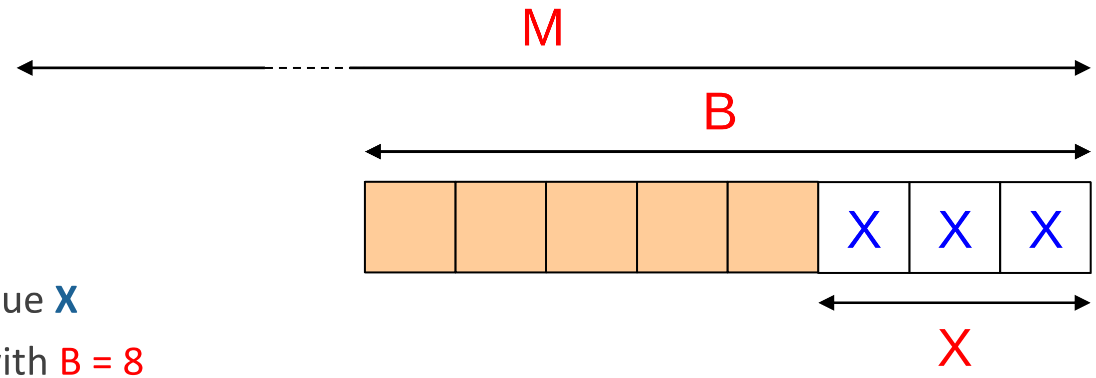
ECB/CBC cipher modes: Trailing sub-block issues

Block cipher modes ECB and CBC require block-aligned inputs

- Trailing sub-blocks need special treatment

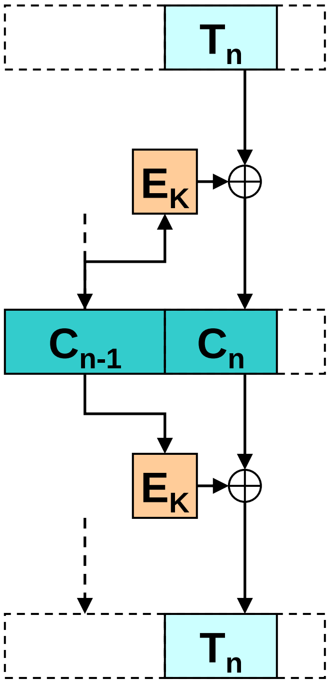
Alternatives

- Padding
  - Of last block, identifiable
  - PKCS #7
    - **X = B – (M mod B)**
    - **X** extra bytes, with the value **X**
  - PKCS #5: Equal to PKCS #7 with B = 8
- Different processing for the last block
  - Adds complexity



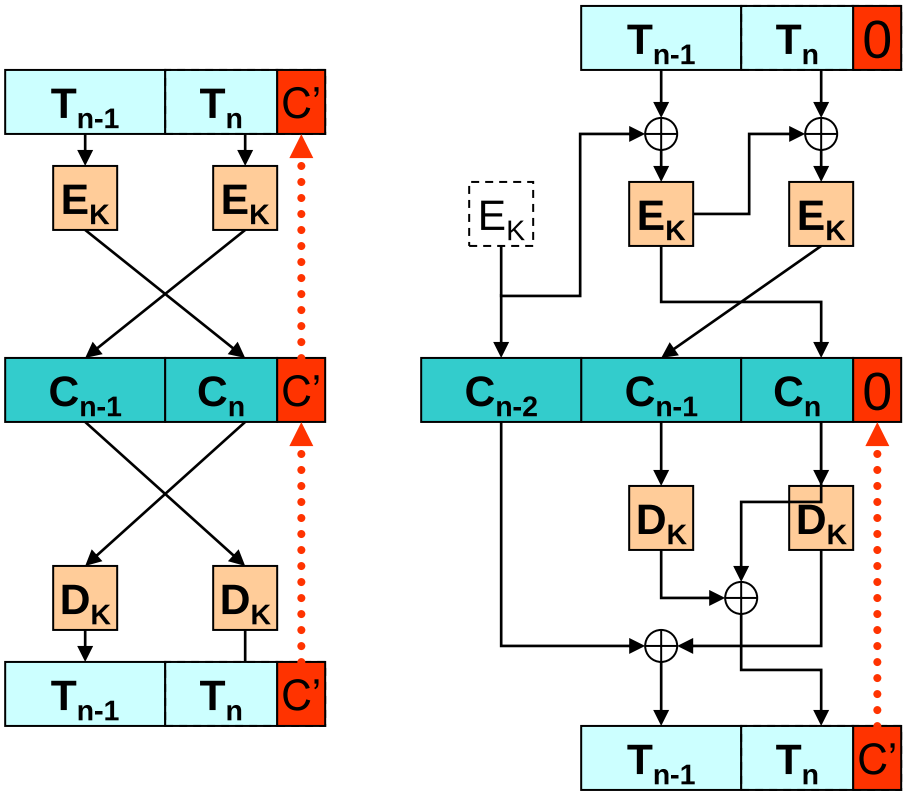
ECB/CBC cipher modes: Handling trailing sub-blocks

Sort of stream cipher



ECB/CBC cipher modes: Handling trailing sub-blocks

Ciphertext stealing



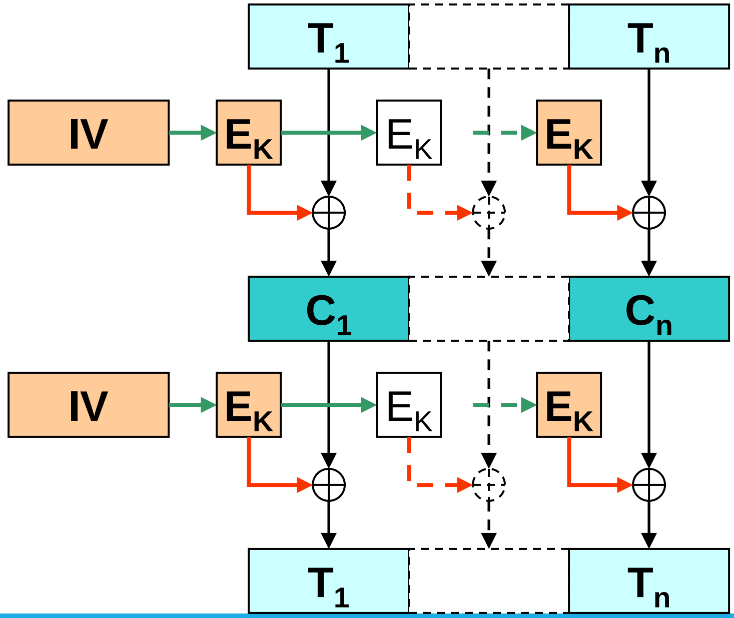
Cipher modes: n-bit OFB (Output Feedback)

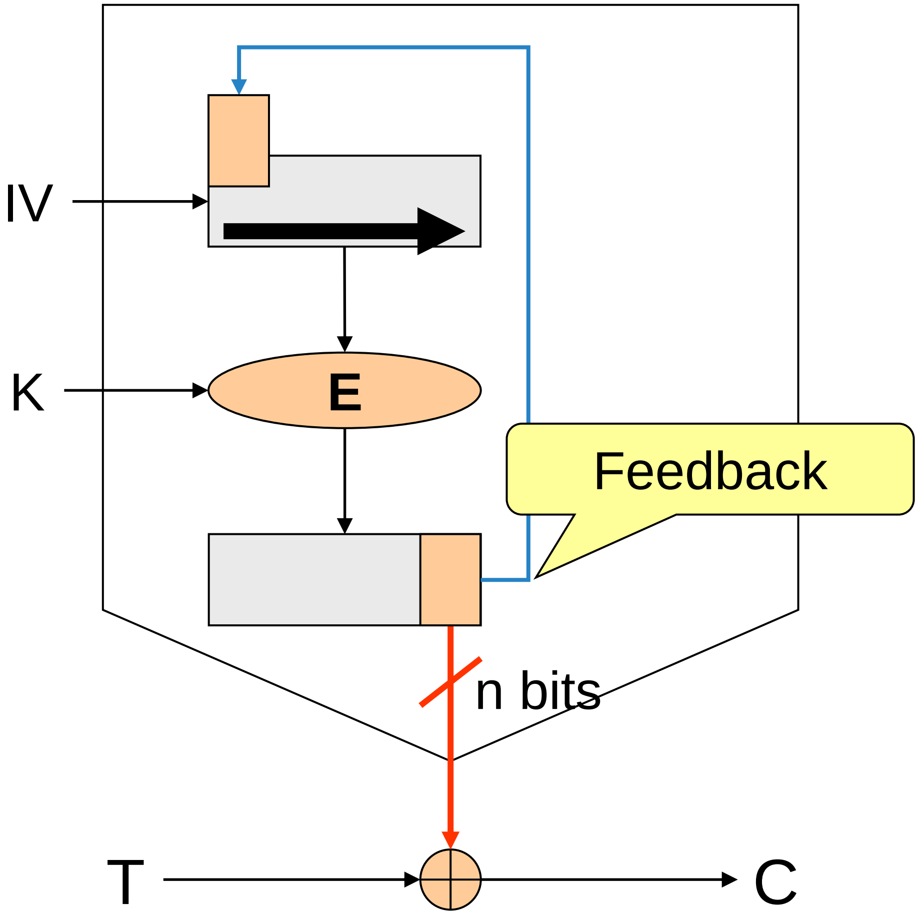
$C_i = T_i \oplus E_K(S_i)$

$T_i = C_i \oplus E_K(S_i)$

$S_i = f(S_{i-1}, E_K(S_{i-1}))$

$S_0 = IV$





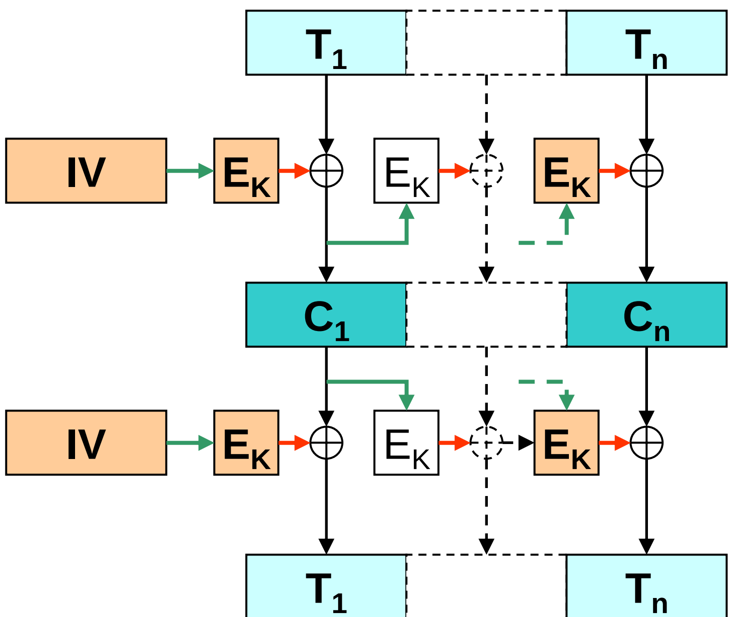
Cipher modes:
n-bit CFB (Ciphertext Feedback)

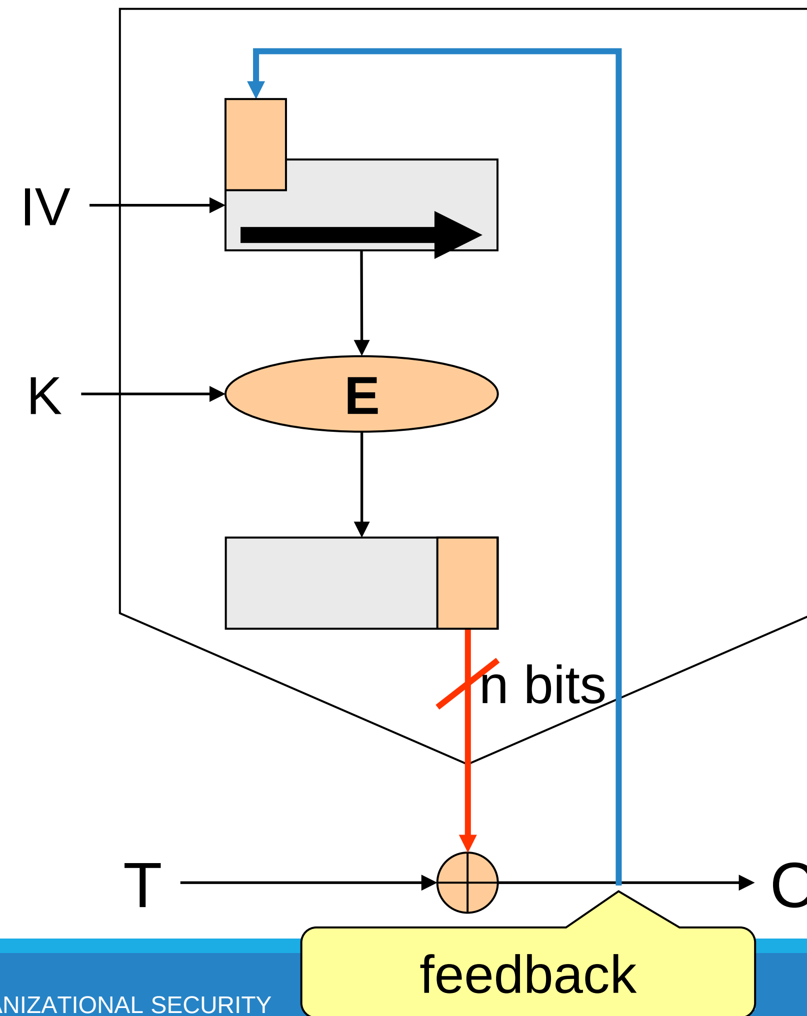
$C_i = T_i \oplus E_K(S_i)$

$T_i = C_i \oplus E_K(S_i)$

$S_i = f(S_{i-1}, C_i)$

$S_0 = IV$





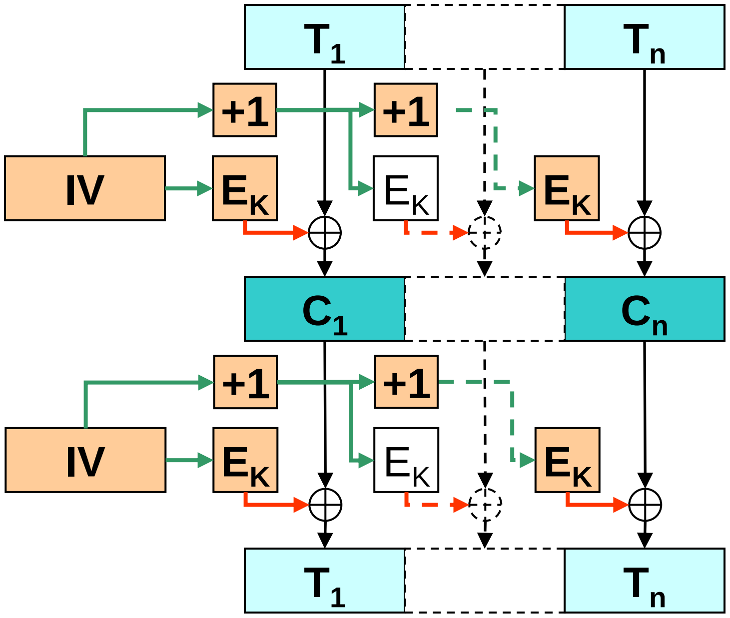
Cipher modes: n-bit CTR (Counter)

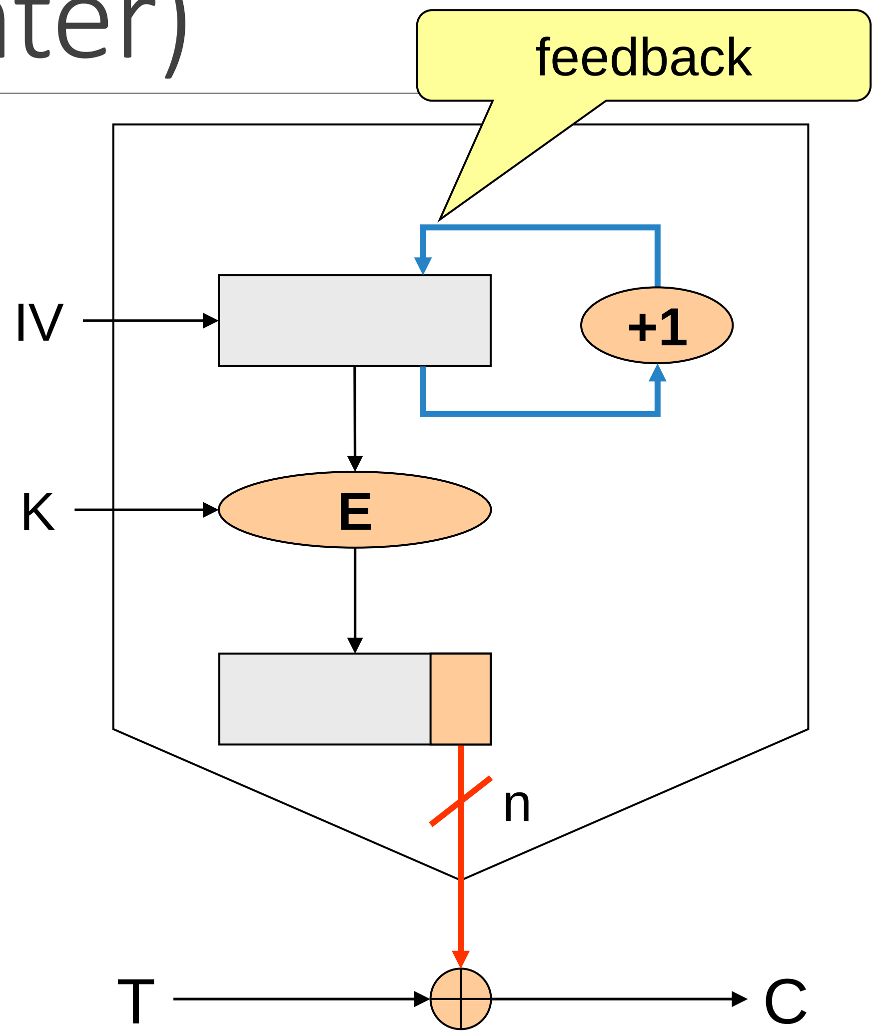
$C_i = T_i \oplus E_K(S_i)$

$T_i = C_i \oplus E_K(S_i)$

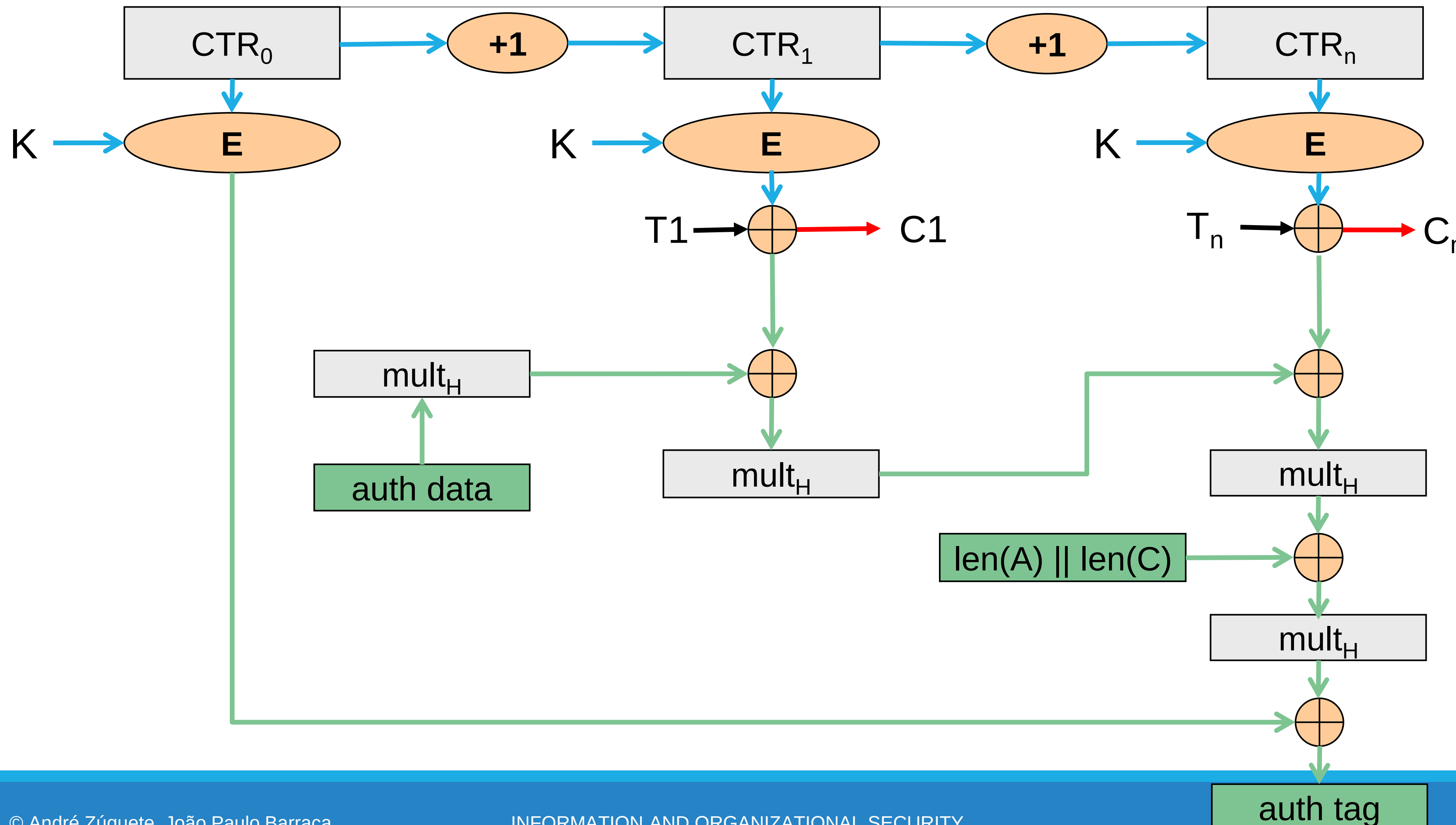
$S_i = S_{i-1}+1$

$S_0 = IV$





Cipher modes:
Galois with Counter Mode (GCM)



Cipher Modes: Comparison

	Block		Stream			
	ECB	CBC	OFB	CFB	CTR	GCM
Input pattern hiding		✓	✓	✓	✓	✓
Confusion on the cipher input		✓		✓	Secret Counter	Secret Counter
Same key for different messages	✓	✓	Other IV	Other IV	Other IV	Other IV
Tampering difficulty	✓	✓ (...)				✓
Pre-processing			✓		✓	✓
Parallel processing **Uniform Random Access**	✓	decrypt	With pre-proc	Decrypt	✓	✓
Error Propagation		Next Block		Next bits		detected
Capacity to recover from losses	Lost blocks	Lost blocks		lost multiple n-bits		detected

Cipher modes:Security reinforcement

Multiple Encryption

Double encryption

- Breakable with a meet-in-the-meddle attack in 2^{n+1} attempts
  - with 2 or more known plaintext blocks
  - Using 2^n blocks stored in memory ...
- No secure enough (theoretically)

Triple encryption (EDE)

- $C_i = E_{K1}(D_{K2}(E_{K3}(T_i)))$ $Pi = D_{K3}(E_{K2}(D_{K1}(C_i))$
- Usually $K_1=K_3$
- If $K_1=K_2=K_3$, then we get simple encryption

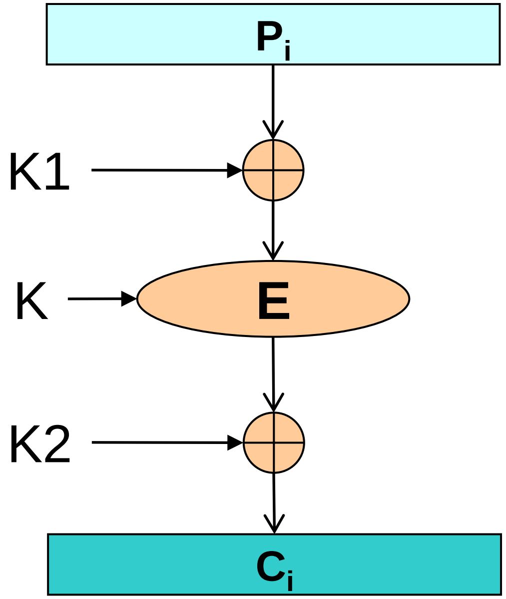
Cipher modes:Security reinforcement

Whitening (DESX)

Simple and efficient technique to add confusion

$C_i = E_K(K_1 \oplus T_i) \oplus K_2$

$T_i = K_1 \oplus D_K(K_2 \oplus C_i)$



Asymmetric (Block) Ciphers

Use key pairs

- One private key (personal, not transmittable)
- One public key, available to all

Allow

- Confidentiality without any previous exchange of secrets
- Authentication
  - Of contents (data integrity)
  - Of origin (source authentication, or digital signature)

Asymmetric (Block) Ciphers

Disadvantages

- Performance (usually very inefficient and memory consuming)

Advantages

- N peers requiring pairwise, secret interaction -> N key pairs

Problems

- Distribution of public keys (must be done before data is exchanged)
- Lifetime of key pairs (must expire)

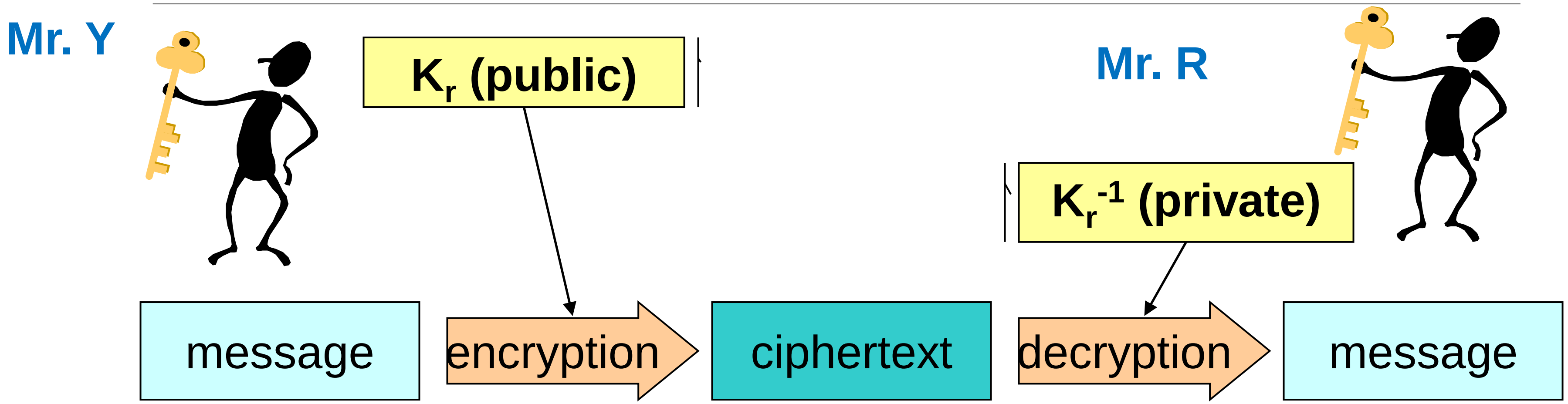
Asymmetric (block) ciphers

Approaches: complex mathematic problems
- Discrete logarithms of large numbers
- Integer factorization of large numbers
- Knapsack problems

Most common algorithms
- RSA
- ElGamal
- Elliptic curves (ECC)

Other techniques with asymmetric key pairs
- Diffie-Hellman (key agreement)

Confidentiality w/ asymmetric ciphers



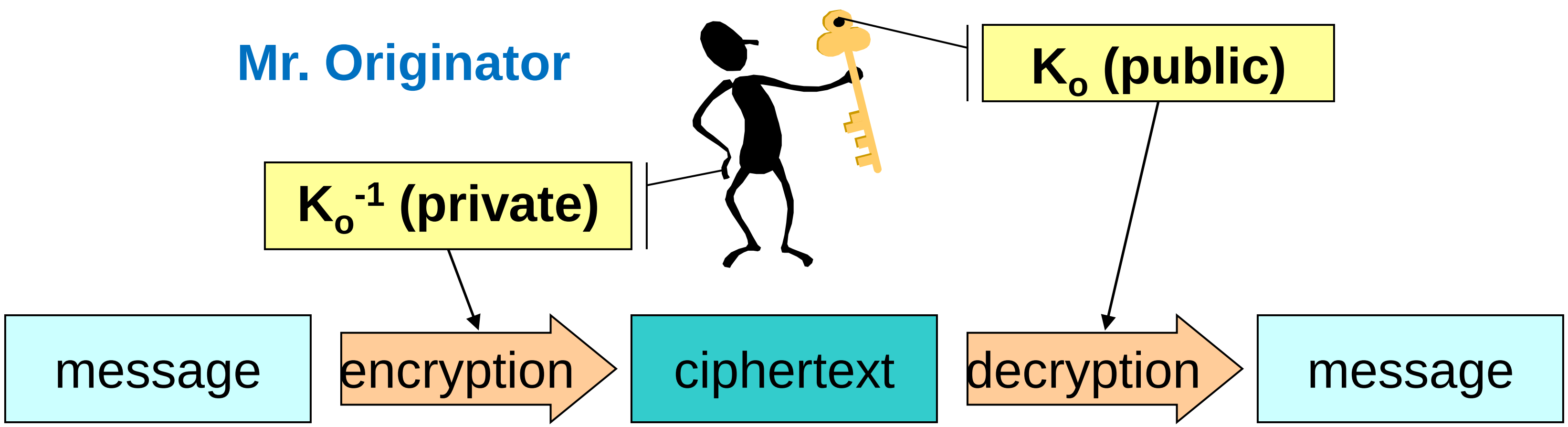
Only uses the keypair of the recipient

- C = E(K, P) P = D(K^{-1}, C)
- Sending a confidential message to **R**, requires **Y** knowing **R** public key (**K_r**)

No authentication

- **R** has no means to know who produced the ciphertext
- If **K_r** is really public, then everybody can do it

Source authentication w/ asymmetric ciphers



Only uses the keypair of the originator

- $C = E(K^{-1}, P)$ $P = D(K, C)$;
- Only **O** knows K_o^{-1}, which was used to produce the ciphertext

There is no confidentiality

- Knowing K_o (public) allows to decrypt the ciphertext
- If K_o is really public, everybody can do it

RSA (Rivest, Shamir, Adelman) 1978

coprime -> gcd(a, b) = 1
× -> multiplication
mod -> modulo operation
≡ -> modular congruence

Computational complexity

- Discrete logarithm
- Integer factoring

Key selection

- Large n (hundreds or thousands of bits)
- n = p × q with p and q being large (secret) prime numbers
- Chose an e co-prime with (p-1) × (q-1)
- Compute d such that e × d ≡ 1 mod ((p-1) × (q-1))
- Discard p and q
- The value of d cannot be computed out of e and n
    - Only from p and q

RSA Example

p = 5 q = 11 (prime numbers)

- **n** = p x q = 55
- (p-1) x (q-1) = 40

e = 3 (public key = e,n)

- Coprime of 40

d = 27 (private key = d,n)

- e x d ≡1 mod (40) -> (d x e) mod (40) = 1, (27 x 3) mod 40 = 1

For T = 26 (notice that T, C∈[0, n-1])

$$C = T^e \bmod n = 26^3 \bmod 55 = 31$$

$$T = C^d \bmod n = 31^{27} \bmod 55 = 26$$

ElGamal - 1984

Similar to RSA, but using only the discrete logarithm complexity

A variant is used for digital signatures (DSA and DSS)

Operations and keys (for signature handling)

- $\beta = \alpha^x \bmod p$ $K = (\beta, \alpha, p)$ $K^{-1} = (x, \alpha, p)$
- k random, $k \cdot k^{-1} \equiv 1 \bmod (p-1)$
- Signature of **M**: (γ,δ) $\gamma = \alpha^k \bmod p$ $\delta = k^{-1} (M - x\gamma) \bmod (p-1)$
- Validation of signature over M: $\beta^\gamma \gamma^\delta \equiv \alpha^M \pmod p$

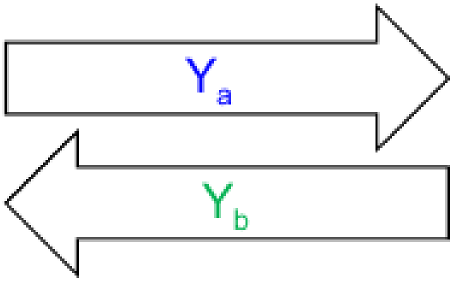
Problem

- Knowing k reveals x out of δ
- k must be randomly generated and remain secret

Diffie-Hellman Key Agreement

q (large prime)

α (primitive root mod q)

a = random		**b = random**
$Y_a = \alpha^a \bmod q$		
		
		$Y_b = \alpha^b \bmod q$
$K_{ba} = {Y_b}^a \bmod q$		$K_{ab} = {Y_a}^b \bmod q$

$$K_{ba} = K_{ab}$$

Diffie-Hellman Key Agreement: MitM attack

a = random

$Y_a = \alpha^a \bmod q$

$\rightarrow Y_a$

$\leftarrow Y_c$

$K_{ca} = Y_c^{\,a} \bmod q$

c = random

$Y_c = \alpha^c \bmod q$

$K_{ac} = Y_a^{\,c} \bmod q$

$K_{cb} = Y_b^{\,c} \bmod q$

$\leftarrow Y_b$

$\rightarrow Y_c$

b = random

$Y_b = \alpha^b \bmod q$

$K_{cb} = Y_c^{\,b} \bmod q$

Randomization of asymmetric encryptions

Non-deterministic (unpredictable) result of asymmetric encryptions

- **N** encryptions of the same value, with the same key, should yield **N** different results
- **Goal:** prevent the trial & error discovery of encrypted values

Approaches

- Concatenation of value to encrypt with two values
  - A fixed one (for integrity control)
  - A random one (for randomization)

Randomization of asymmetric encryptions:OAEP
(Optimal Asymmetric Encryption Padding)

lHash: Digest over Label

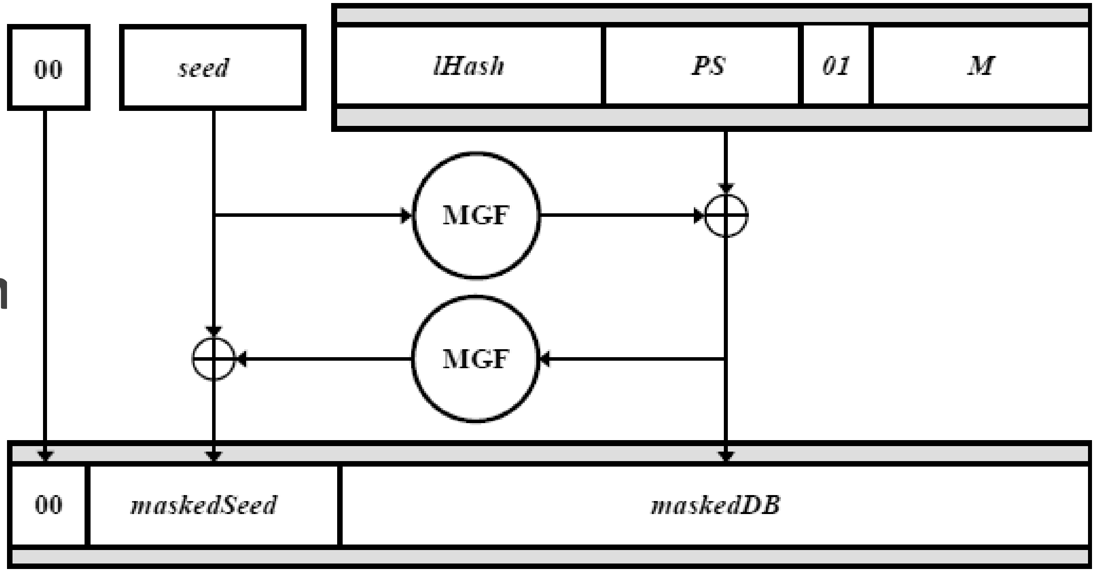
seed: Random

PS: zeros

M: Plain Text

MGF: Mask Generation Function

- Similar to Hash, but with variable size



Performance Increase: Hybrid cipher

Combine Symmetric with Asymmetric Cipher

- Use the best of both worlds, while avoiding problems
- Asymmetric cipher: Uses public keys (but it's slow)
- Symmetric cipher: Fast (but with weak key exchange methods)

Method:

1. Obtain K_{pub} from the receiver
2. Generate a random K_{sim}
3. Calculate $C_1 = E_{sim}(K_s, T)$
4. Calculate $C_2 = E_{asim}(K_{pub}, K_s)$
5. Send $C_1 + C_2$
    - C1 = Text encrypted with symmetric key
    - C2 = Symmeric key encrypted with the receiver public key
        - May also contain the IV

Digest functions

Give a fixed-length value from a variable-length text

- Sort of text “fingerprint”

Produce very different values for similar texts

- Cryptographic one-way hash functions

Relevant properties:

- Preimage resistance
  - Given a digest, it is unfeasible to find an original text producing it
- 2nd-preimage resistance
  - Given a text, it is unfeasible to find another one with the same digest
- Collision resistance
  - It is unfeasible to find any two texts with the same digest
  - Birthday paradox

Digest functions: size

Considering the similar, yet different texts:

- T1: “Hello User_A!”, T2:“Hello User_B!”, T3:“Hello User_XY!”

Different algorithms will result in values with different dimension, but independent of the dimension of the text

- MD5:
  - T1: 70df836fdaf02e0dfc990f9139762541
  - T3: a08313b553d8bf53ca7457601a361bea
- SHA-1:
  - T1: f591aa1eabcc97fb39c5f422b370ddf8cb880fde
  - T3: c28b0520311e471200b397eaa55f1689c8866f25
- SHA-256:
  - T1: 9649d8c0d25515a239ec8ec94b293c8868e931ad318df4ccd0dffd67aff89905
  - T3: 8fc49cde23d15f8b9b1195962e9ba517116f45661916a0f199fcf21cb686d852

Digest functions: size

Considering the similar, yet different texts:

- T1: “Hello User_A!”, T2:“Hello User_B!”, T3:“Hello User_XY!”

A small change in the text (1 bit) results in a completely different result

- MD5:
  - T1: 70df836fdaf02e0dfc990f9139762541
  - T2: c32e0f62a7c9c815063d373acac80c37
- SHA-1:
  - T1: f591aa1eabcc97fb39c5f422b370ddf8cb880fde
  - T2: bab31eb62f961266758524071a7ad8221bc8700b
- SHA-256:
  - T1: 9649d8c0d25515a239ec8ec94b293c8868e931ad318df4ccd0dffd67aff89905
  - T2: e663a01d3bec4f35a470aba4baccece79bf484b5d0bffa88b59a9bb08707758a

Digest functions

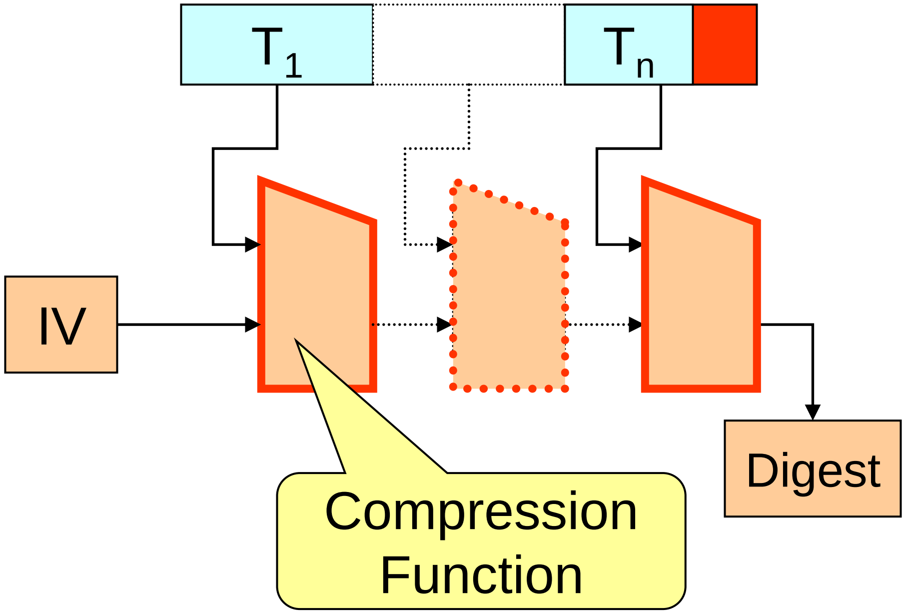
Approaches

- Collision-resistant, one-way compression functions
- Merkle-Damgård construction
  - Iterative compression
  - Length padding

Most common algorithms

- MD5 (128 bits)
  - No longer secure! It's easy to find collisions!
- SHA-1 (Secure Hash Algorithm, 160 bits)
  - Also no longer secure … (collisions found in 2017)
- **SHA-2, aka SHA-256/SHA-512, SHA-3**

Digest functions



Message Integrity Code (MIC)

Provide the capability to detect changes by devices

- Communication/storage errors
- From a random process or without control

Send: Calculate MIC and send T + MIC

- T = Text, MIC = digest(T)

Receive: Receive data (T') and check if D(T) = MIC

- Calculate S'=digest(T')
- Validate if S(T') = MIC

Doesn't protect from planned changes to the text

- Attacker can manipulate T into T'' and calculate a new MIC''

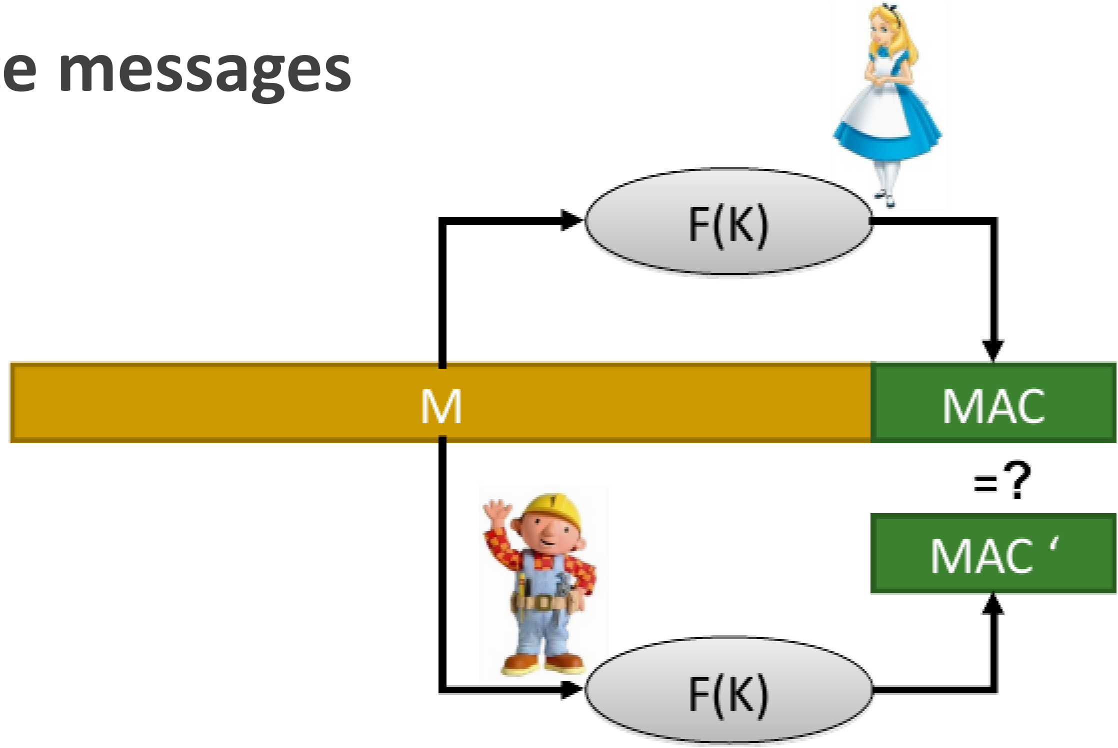
Message Authentication Codes (MAC)

Hash, or digest, computed with a key

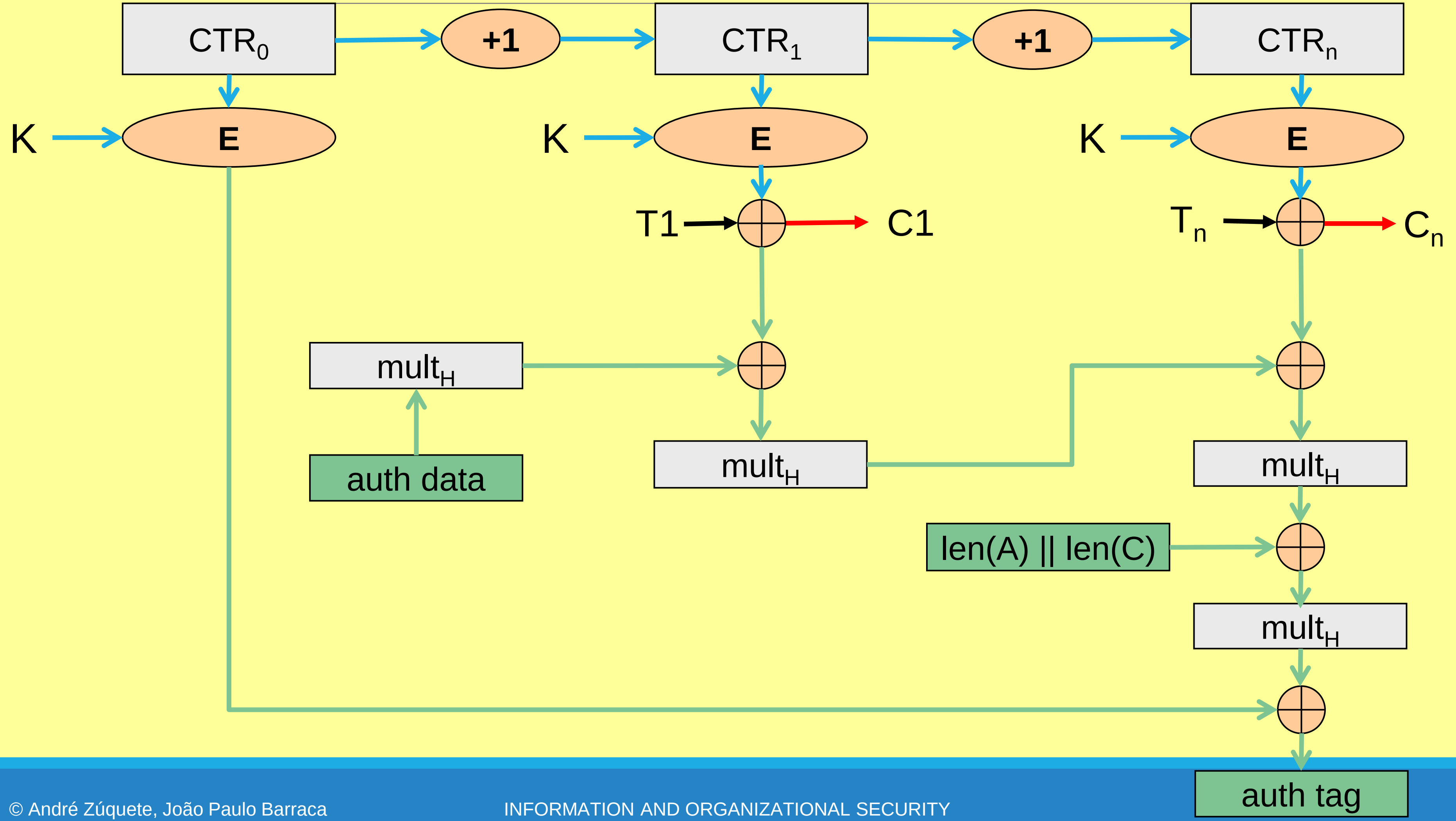
- Only key holders can generate/validate the MAC

Used to authenticate messages

- M' = M | MAC(M)



Example: GCM



CTR0
+1
CTR1
+1
CTRn
K
E
K
E
K
E
T1
C1
Tn
Cn
multH
auth data
multH
multH
len(A) || len(C)
multH
auth tag

Message Authentication Codes (MAC):Approaches

Encryption of an ordinary digest

- Using, for instance, a symmetric block cipher

Using encryption with feedback & error propagation

- ANSI X9.9 (or DES-MAC) with DES CBC (64 bits)

Adding a key to the hashed data

- Keyed-MD5 (128 bits)
  - MD5(K, keyfill, text, K, MD5fill)
- HMAC (output length depends on the function H used)
  - H(K, opad, H(K, ipad, text))
  - ipad = 0x36 B times   opad = 0x5C B times
    - HMAC-MD5, HMAC-SHA, etc.

Encryption + Authentication

Encrypt-then-MAC: MAC is computed from cryptogram
- Allows verifying integrity before (the longer) decryption

BAD

Encrypt-and-MAC: MAC is computed from plaintext
- MAC is not encrypted
- May give information regarding original text (if similar to other)

MAC-then-Encrypt: MAC is computed from plaintext
- MAC is encrypted
- Requires full decryption before MAC is validated

Digital Signatures

Authenticate the contents of a document

- Ensure its integrity (It was not changed)

Authenticate its author

- Ensure the identity of the creator/originator

Prevent repudiation of creating a content

- Genuine authors cannot deny authorship
  - Only the author could have generated a given signature
  - Note: Author is the creator of a content, not who sends the content

Digital Signatures

Approaches

- Asymmetric encryption
- Digest functions (only for performance)

Signing: $A_x(doc) = info + E(K_x^{-1}, digest(doc + info))$

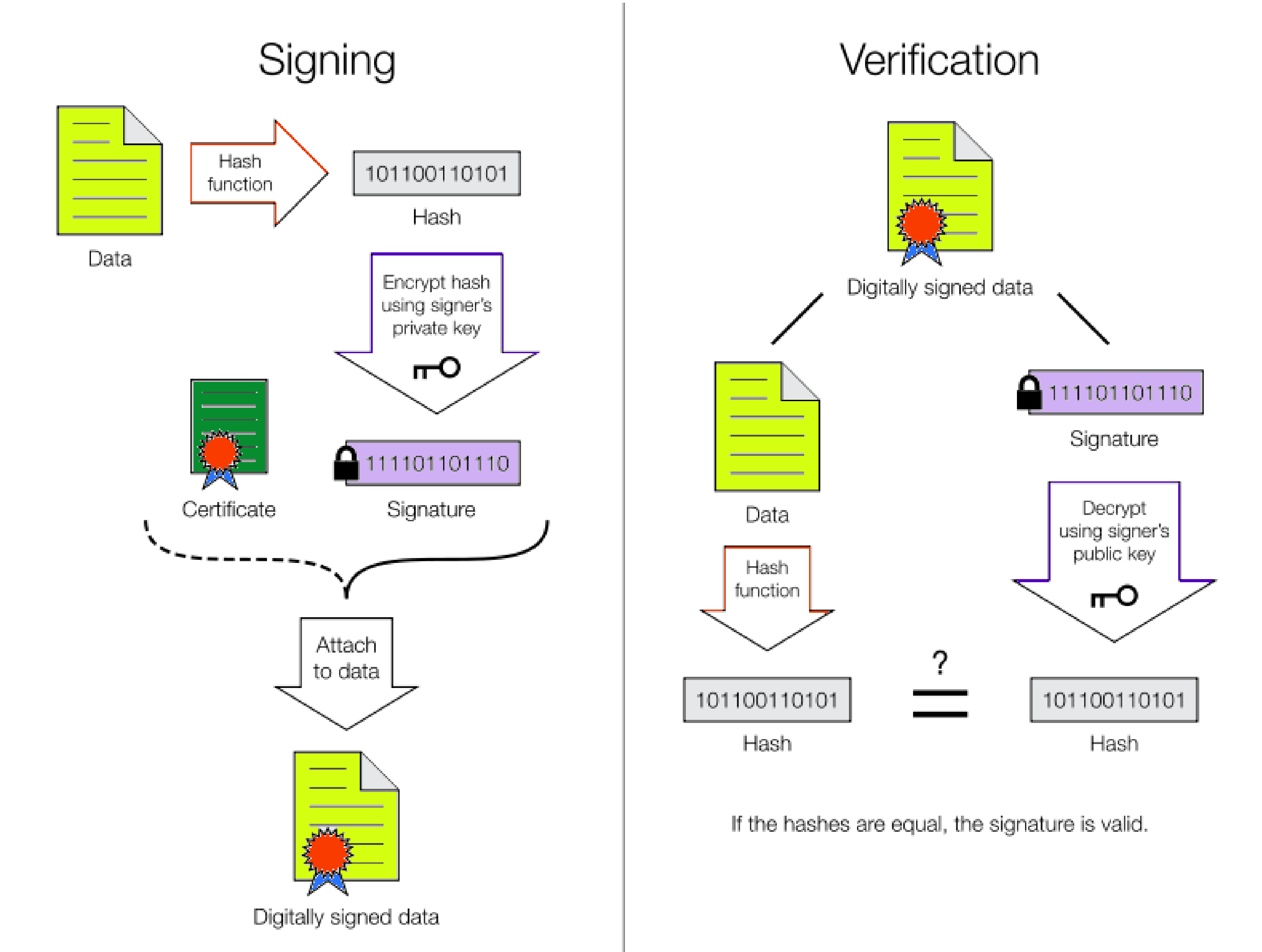
info associated with K_x

Verification:

$D(K_x, A_x(doc)) \equiv digest(doc + info)$

wikipedia, http://en.wikipedia.org/wiki/Digital_signature

Signing / verification diagrams



Digital signature on a mail: Multipart content, signature w/ certificate

From - Fri Oct 02 15:37:14 2009
[…]
Date: Fri, 02 Oct 2009 15:35:55 +0100
From: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Reply-To: andre.zuquete@ua.pt
Organization: IEETA / UA
MIME-Version: 1.0
To: =?ISO-8859-1?Q?Andr=E9_Z=FAquete?= <andre.zuquete@ua.pt>
Subject: Teste
Content-Type: multipart/signed; protocol="application/x-pkcs7-signature"; micalg=sha1; boundary="------------ms050405070101010502050101"

This is a cryptographically signed message in MIME format.

--------------ms050405070101010502050101
Content-Type: multipart/mixed;
boundary="------------060802050708070409030504"

This is a multi-part message in MIME format.
--------------060802050708070409030504
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable

Corpo do mail

--------------060802050708070409030504—
--------------ms050405070101010502050101
Content-Type: application/x-pkcs7-signature; name="smime.p7s"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="smime.p7s"
Content-Description: S/MIME Cryptographic Signature

MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAoIIamTCC
BUkwggSyoAMCAQICBAcnIaEwDQYJKoZIhvcNAQEFBQAwdTELMAkGA1UEBhMCVVMxGDAWBgNV
[…]
KoZIhvcNAQEBBQAEgYCofks852BV77NVuww53vSxO1XtI2JhC1CDlu+tcTPoMD1wq5dc5v40
Tgsaw0N8dqgVLk8aC/CdGMbRBu+J1LKrcVZa+khnjjtB66HhDRLrjmEGDNttrEjbqvpd2QO2
vxB3iPTlU+vCGXo47e6GyRydqTpbq0r49Zqmx+IJ6Z7iigAAAAAAAA==
--------------ms050405070101010502050101--

Blind signatures

Signatures made by a “blinded” signer

- Signer cannot observe the signed contents
- Similar to a handwritten signature on an envelope containing a document and a carbon-copy sheet

Useful for ensuring anonymity of the signed information holder, while the signed information provides some extra functionality

- Signer X knows someone who requires a signature (Y)
- X blinds T_1 into T_2, and Y afterwards transforms it into a signature over T_2
  - **T_2 = blind(b_x, T_1)**
- Requester Y can present T_2 signed by X
  - But it cannot change T_2
  - X cannot link T_2 to the T_1 that it observed when blinding

Chaum Blind Signatures w/ RSA

Blinding

- Random blinding factor K
- $k \times k^{-1} \equiv 1 \pmod N$
- $m' = k^e \times m \bmod N$

Ordinary signature (encryption w/ private key)

- $Ax(m') = (m')^d \bmod N$

Unblinding

- $A_x(m) = k^{-1} \times A_x(m') \bmod N$

Key Derivation

Cipher algorithms require fixed dimension keys

- 56, 128, 256… bits

It's needed to derive keys from multiple sources

- Shared secrets
- Passwords generated by humans
- PIN codes and small length secrets

Original source may have low entropy

- Reduces the difficulty of a brute force attack
- Although we must have some strong relation into a useful key

Sometimes it's needed to generate multiple keys from the same material

- While not allowing to find the material (a password) from the key

Key Derivation - Purposes

Key reinforcement: Increase the security of a password

- Usually defined by humans
- Making dictionary attacks impractical

Key expansion: Increase the dimension of a key

- Expansion to a size that suits the algorithm
- Eventually derive other related keys for other algorithms (MAC)

Key Derivation

Key derivation requires the existence of:

- A salt which makes the derivation unique
- A difficult problem
- A chosen level of complexity

Computational difficulty: Transformation will require the use of relevant computational resources

Memory difficulty: Transformation allows relevant storage resources

- Limits attacks using dedicated hardware accelerators

Key Derivation: PKBDF2

Password Based Key Derivation Function 2

Produces a key from a password, with a chosen difficulty

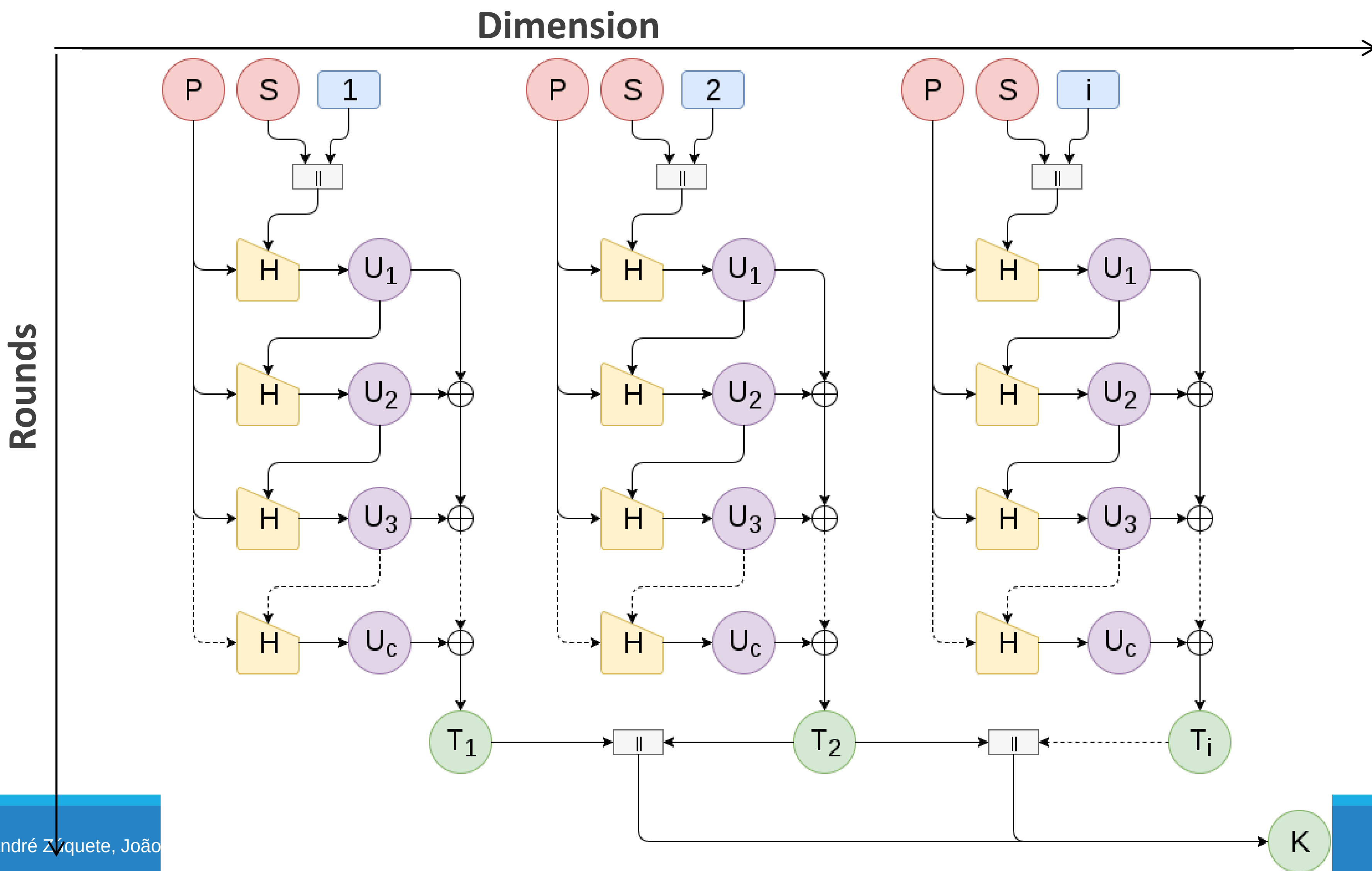
K = PBKDF2(PRF, Salt, rounds, password, dim)

- PRF: Pseudo-Random-Function: A digest
- Salt: a random value
- Rounds: the computational cost (tens or hundreds of thousands)
- Dim: the size of the result required

Operation: calculates ROUNDS x DIM operations from the PRF using the SALT and PASSWORD

- Larger number of rounds will increase the cost

Key Derivation: PBKDF2



Key Derivation: scrypt

Produces a key with a chosen storage cost

K = scrypt(password, salt, n, p, dim, r, hLen, Mflen)

- Password: a secret
- Salt: a random value
- N: the cost parameter
- P: the parallelization parameter. $p \leq (2^{32} - 1) * hLen / MFLen$
- Dim: the size of the result
- R: the size of the blocks to use (default is 8)
- hLen: the size of the digest function (32 for SHA256)
- Mflen: bytes in the internal mix (default is 8 x R)

Key Derivation: scrypt