Co-funded by the Erasmus+ Programme of the European Union

ISSES

ISSES – Information Security Services Education in Serbia

Supported by the Erasmus+ Capacity Building in the field of Higher Education (CBHE) grant N° 586474-EPP-1-2017-1-RS-EPPKA2-CBHE-JP

HACKING CRYPTOGRAPHIC SYSTEMS

Information Security Services Education in Serbia (ISSES), Erasmus+ Key Action 2 – Capacity Building in the field of Higher Education (CBHE), University of Novi Sad, 2017 – 2020

ISSES 2017-2020
Novi Sad, Serbia

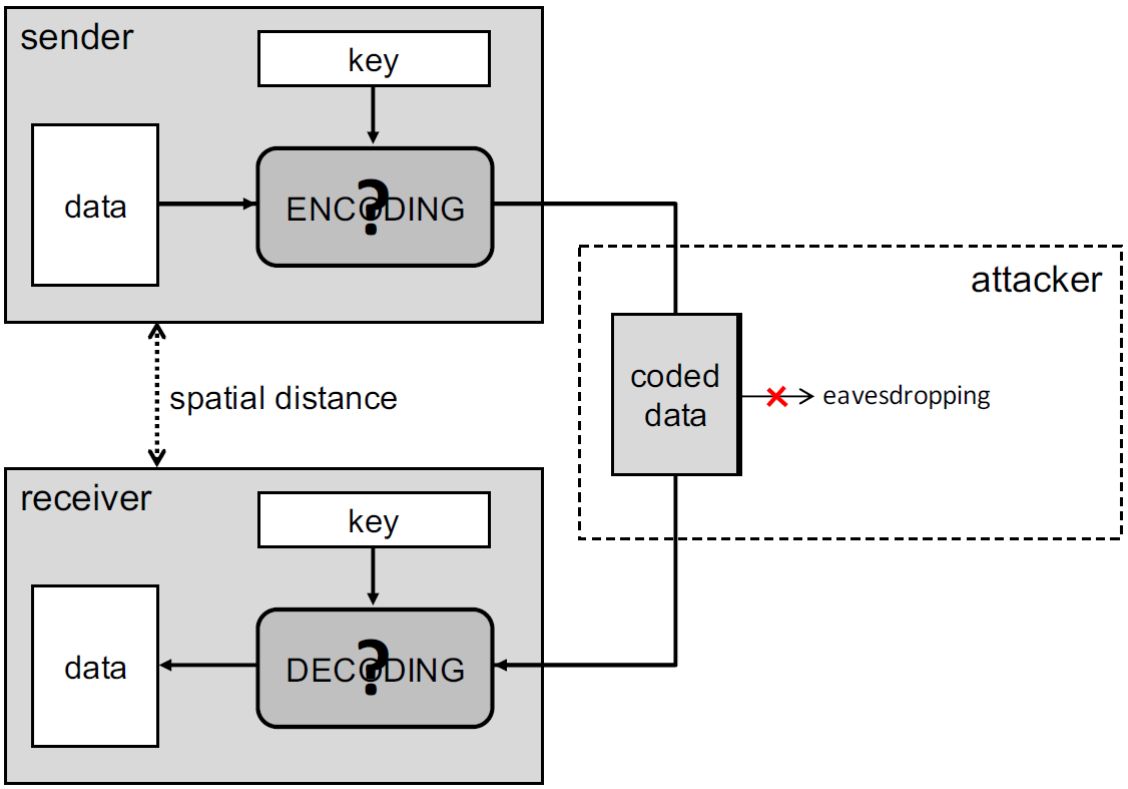
Vision

- Goal: From Zero to Hero in crypto hacking
- Cryptography intro and history
  - Stream ciphers
  - Block ciphers
  - Public key crypto
- Black-box hacking
- Breaking the 4-square cipher

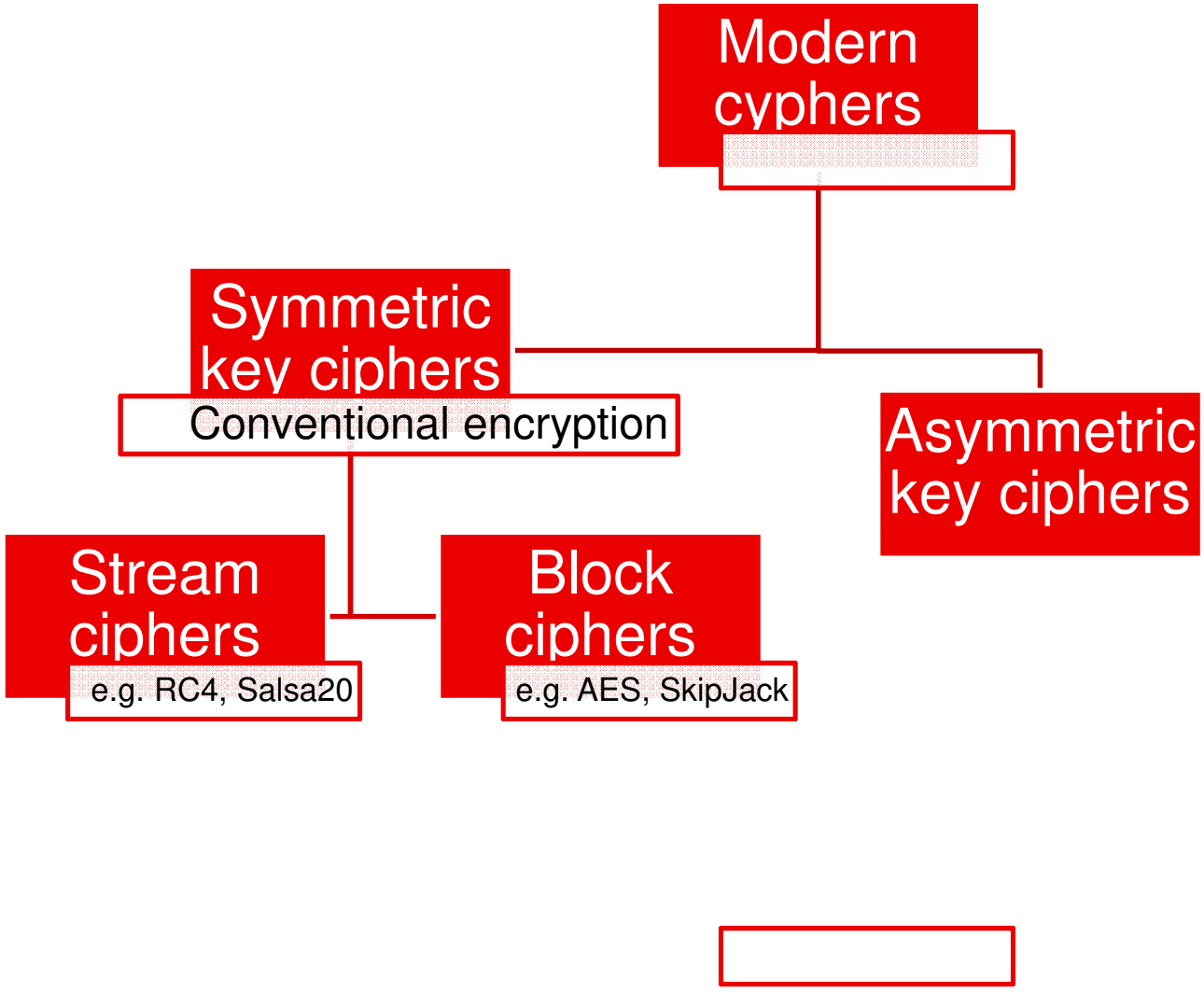
Cryptography intro

- **DEF:** Cryptographic systems consist of a set of algorithmic tools to solve security problems
  - encryption → confidentiality
  - message authentication codes (MAC) → integrity and authenticity
  - digital signature → integrity, authenticity, non-repudiation (of origin)
- Often, cryptography is the only possible or viable solution
- but be careful…
  - "If you think cryptography is the answer to your problem, then you don't know what your problem is…", Peter G. Neumann
  - "… or you don't understand cryptography." Bruce Schneier
- The proper application of cryptography is an engineering problem

Basic model of cryptographic systems



Modern ciphers



A BIT OF CRYSTORY…

Caesar cypher

- Substitution cipher (replaces letters of the plaintext) used by Julius Caesar (more than 2000 years ago!)
- Each letter is replaced by the letter at some fixed number of positions (e.g., 3) down in the alphabet

plain: **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**

cipher: **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**

example: **BRUTUS → EUXWXV**

Caesar cypher disc

Monoalphabetic substitution

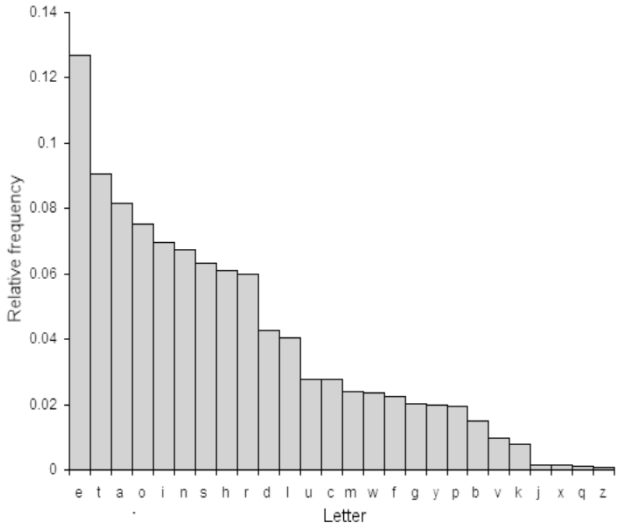
- Mono-alphabetic substation is a generalization of the Caesar cipher
- The replacement of letters is determined by a permutation

  plain: **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**

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

  coding example: **BRUTUS → TVXPXF**
- The key is the permutation and the key space is very large: $26! = 1.56 * 2^{88}$
  - Time left until the next ice age ..... $2^{39}\ sec$
  - Time left until the Sun becomes a supernova ..... $2^{55}\ sec$
  - The age of the Earth ..... $2^{55}\ sec$
  - The age of the Universe ..... $2^{59}\ sec$

Breaking monoalphabetic substitutions



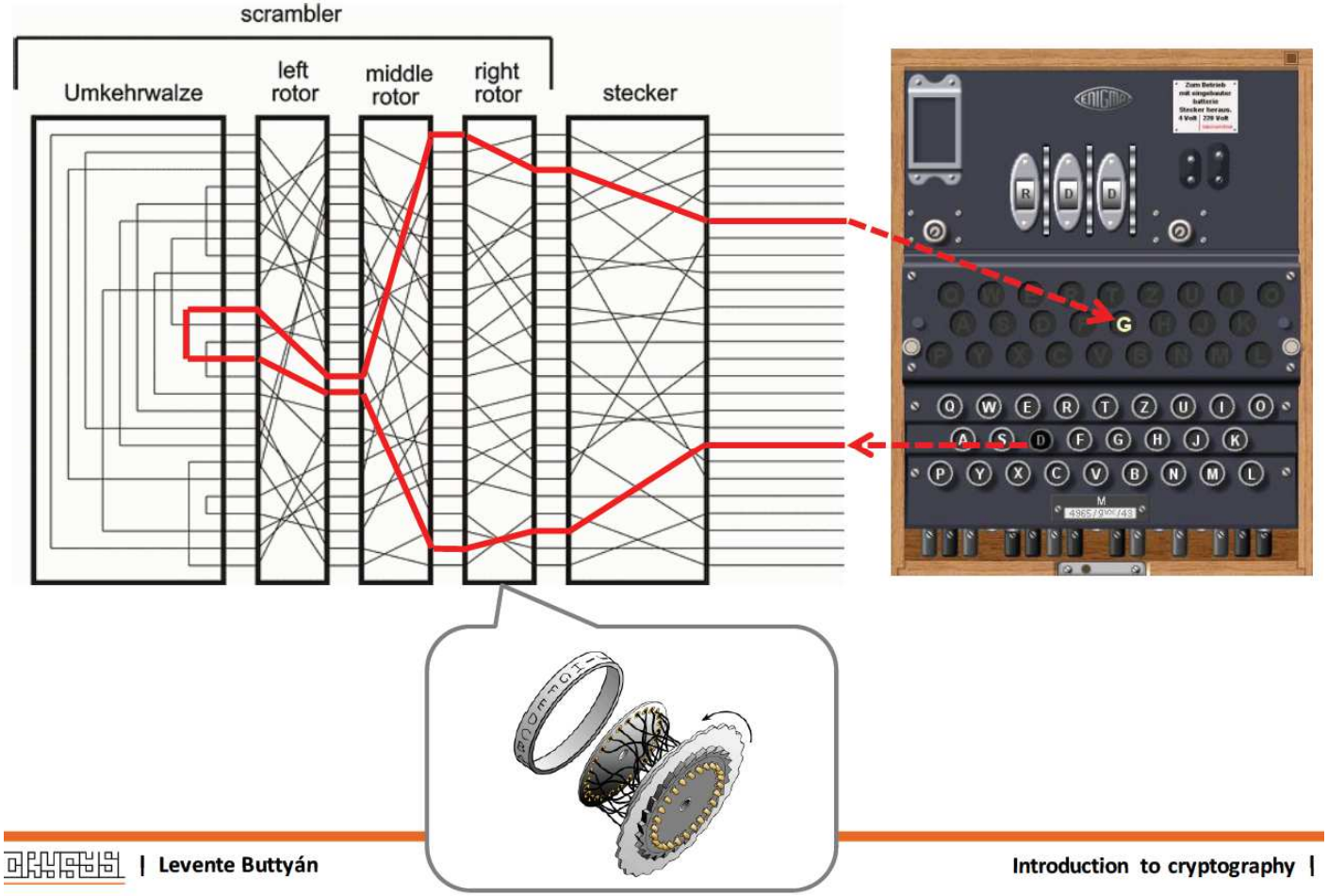
- Every language has its own letter statistics
  - Letter frequencies are independent of the actual text
  - There are letters that are more frequent than others…
    - e → 12.7%, t → 9.1% in the English language
  - … and letters that are less frequent:
    - z → 0.1%, j → 0.2% in in the English language
- In case of monoalphabetic substitution, the ciphertext preserves the letter statistics of the original plaintext!
  - After decoding the most frequent and least frequent letters, the rest of the text can be figured out much like solving a crossword puzzle

Enigma

- The Enigma machine was an electro-mechanical cipher
- Adopted by the Wehrmacht in 1926
- Used heavily by Germans in WWII

- Let's try it out → Enigma emulator: http://people.physik.hu-berlin.de/~palloks/js/enigma/enigma-u_v25_en.html

Enigma in action



Breaking the Enigma

STREAM CIPHERS

Properties of stream ciphers

- Stream ciphers are usually very efficient
  - Fast (especially in hardware)
  - Require limited memory space
- The ciphertext always has the same length as the plaintext
- Synchronization is needed between the sender and the receiver
  - Loss of synchrony needs to be detected and addressed
- Stream ciphers do not provide any integrity protection !!!
  - An attacker can make changes to selected ciphertext characters and know exactly what effect these changes have on the plaintext
  - The receiver may not notice these changes (!)

The XOR operator

- XOR (+ or ▪)
  - 0+0 = 0; 0+1 = 1; 1+0 = 1; 1+1 = 0

- XOR of bit vectors (words)
  - we XOR each corresponding bit pairs, e.g., 0011 + 1010 = 1001

- three main properties:

  1. X + 0 = 0 + X = X
  2. X + X = 0
  3. if A + B = C, then A = B + C (and B = A + C)

Simple XOR cipher

- represent the plaintext as a sequence of bytes
- take a password and repeat it many times to get a byte string as long as the plaintext
- obtain the ciphertext by XOR-ing together the plaintext and the password string

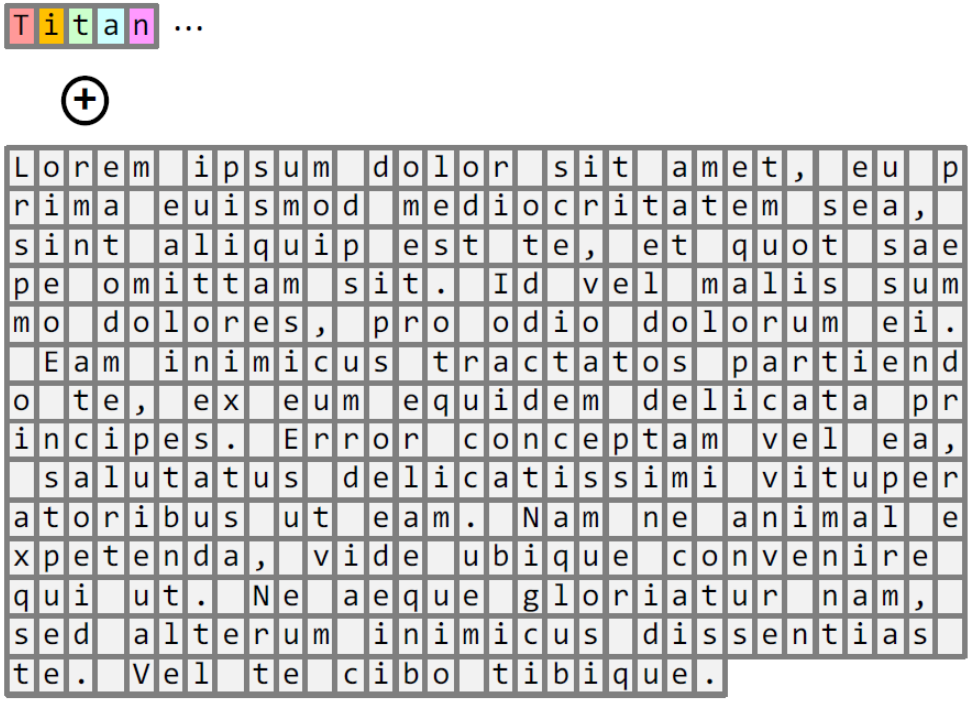
Lorem ipsum dolor sit amet, eu p
rima euismod mediocritatem sea,
sint aliquip est te, et quot sae
pe omittam sit. Id vel malis sum
mo dolores, pro odio dolorum ei.
Eam inimicus tractatos partiend
o te, ex eum equidem delicata pr
incipes. Error conceptam vel ea,
salutatus delicatissimi vituper
atoribus ut eam. Nam ne animal e
xpetenda, vide ubique convenire
qui ut. Ne aeque gloriatur nam,
sed alterum inimicus dissentias
te. Vel te cibo tibique.

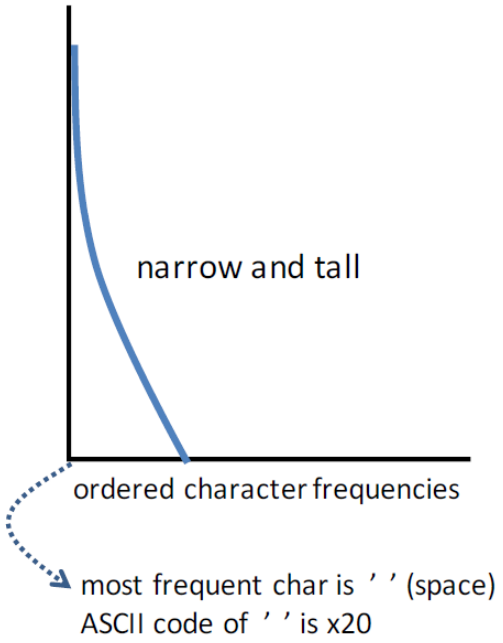
XOR

TitanTitanTitanTitanTitanTitanTi
tanTitanTitanTitanTitanTitanTita
nTitanTitanTitanTitanTitanTitanT
itanTitanTitanTitanTitanTitanTit
anTitanTitanTitanTitanTitanTitan
TitanTitanTitanTitanTitanTitanTi
tanTitanTitanTitanTitanTitanTita
nTitanTitanTitanTitanTitanTitanT
itanTitanTitanTitanTitanTitanTit
anTitanTitanTitanTitanTitanTitan
TitanTitanTitanTitanTitanTitanTi
tanTitanTitanTitanTitanTitanTita
nTitanTitanTitanTitanTitanTitanT
itanTitanTitanTitanTitanT

Decryption → XOR the same password string to the ciphertext to recover the plaintext

Breaking the simple XOR cipher – Step 1 of 5

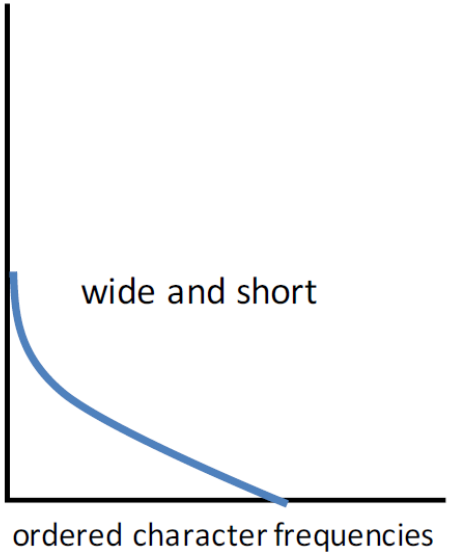




Breaking the simple XOR cipher – Step 2 of 5

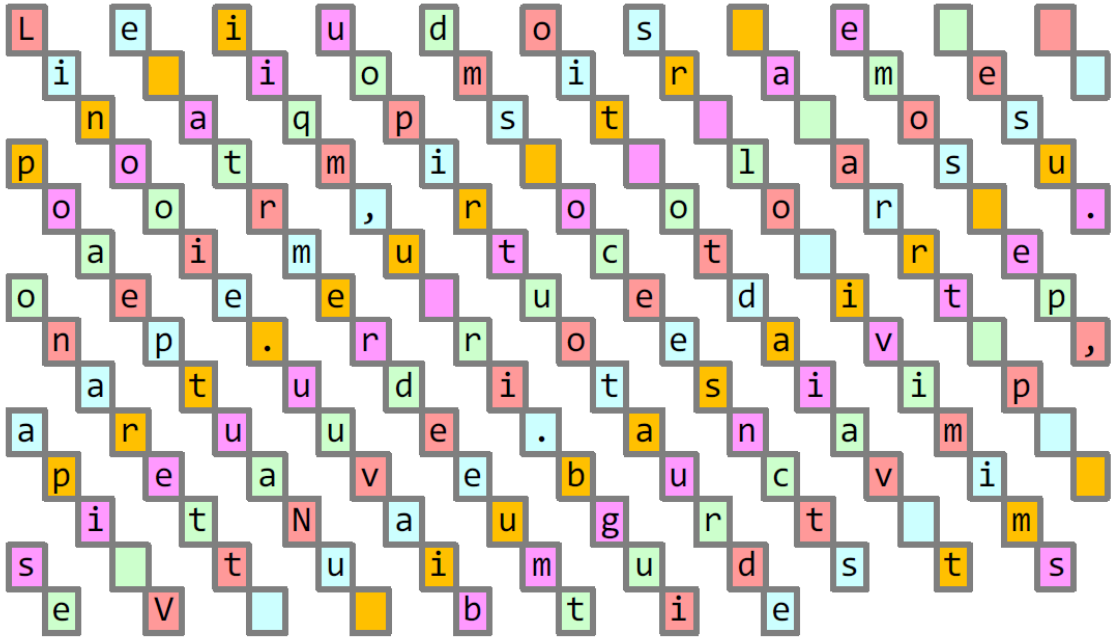
Let's determine the length of the key ...

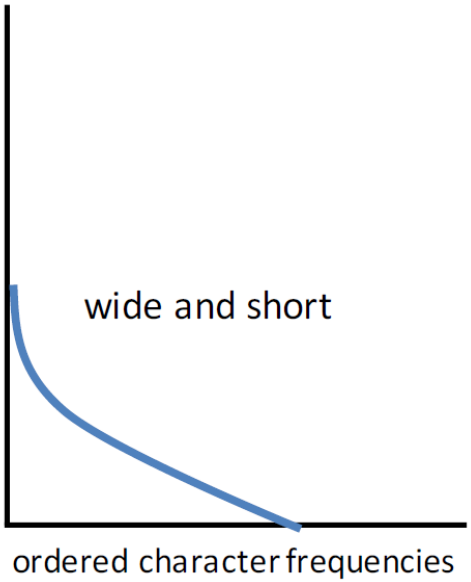
```
Lorem ipsum dolor sit amet, eu p
rima euismod mediocritatem sea,
sint aliquip est te, et quot sae
pe omittam sit. Id vel malis sum
mo dolores, pro odio dolorum ei.
 Eam inimicus tractatos partiend
o te, ex eum equidem delicata pr
incipes. Error conceptam vel ea,
 salutatus delicatissimi vitupera
toribus ut eam. Nam ne animal e
xpetenda, vide ubique convenire
qui ut. Ne aeque gloriatur nam,
sed alterum inimicus dissentias
te. Vel te cibo tibique.
```



Breaking the simple XOR cipher – Step 3 of 5

Attempt #1: Keep every 3rd letter of the coded message…

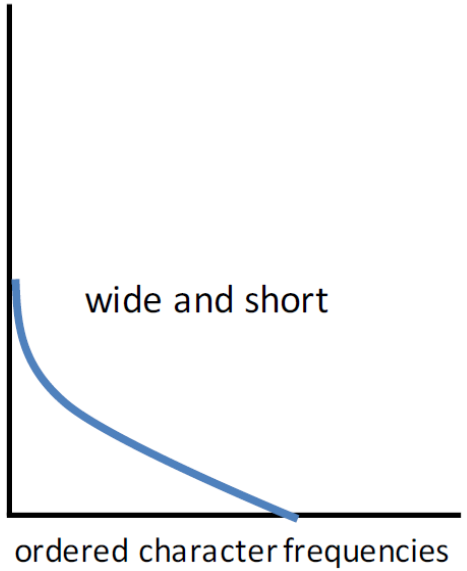




Breaking the simple XOR cipher – Step 4 of 5

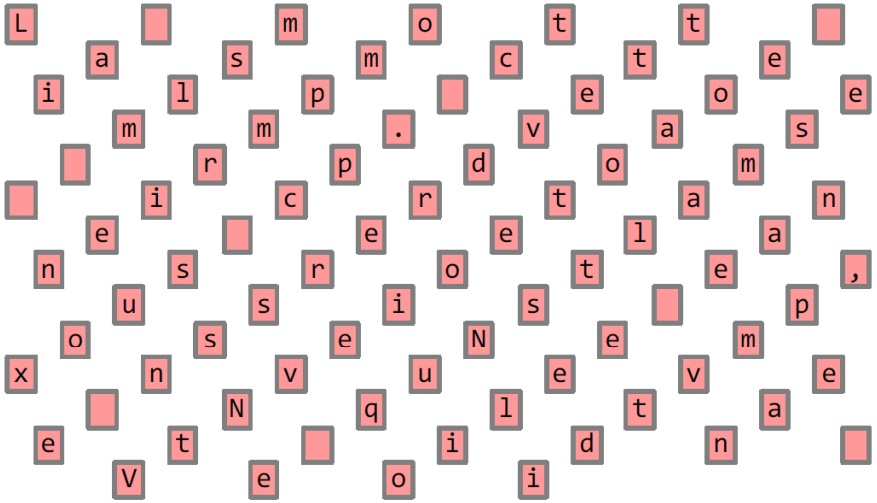
Attempt #2: Keep every 4th letter of the coded message…

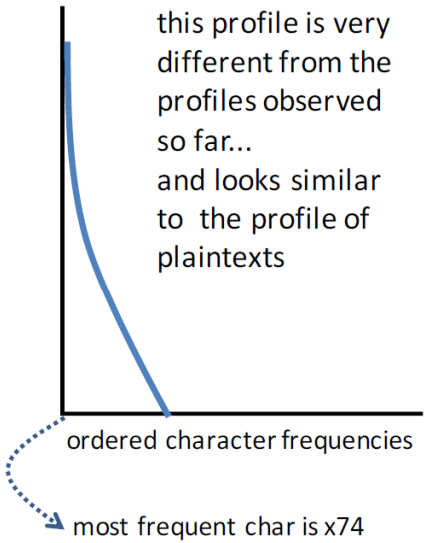
L	m	s	d	r	t	e	e
r		s		i	i	e	e
s	m	q	i	I	e	q	
p	o	a	p	o	t	a	
m		e	s	a		o	i
	,	m		i	p	p	
o	p		o	o	s	i	a
i	u	u	e	a			
	i		e		e	a	u
a	e	,	d	b	r	n	a
x	u	N	e			u	i
q	a	r	i	i	q	s	a
s	V	t	i	t			i
t							



Breaking the simple XOR cipher – Step 5 of 5

Attempt #3: Keep every 5th letter of the coded message…





Let's determine the characters of the key …

' ' + ▯ = x20 + ▯ = x74

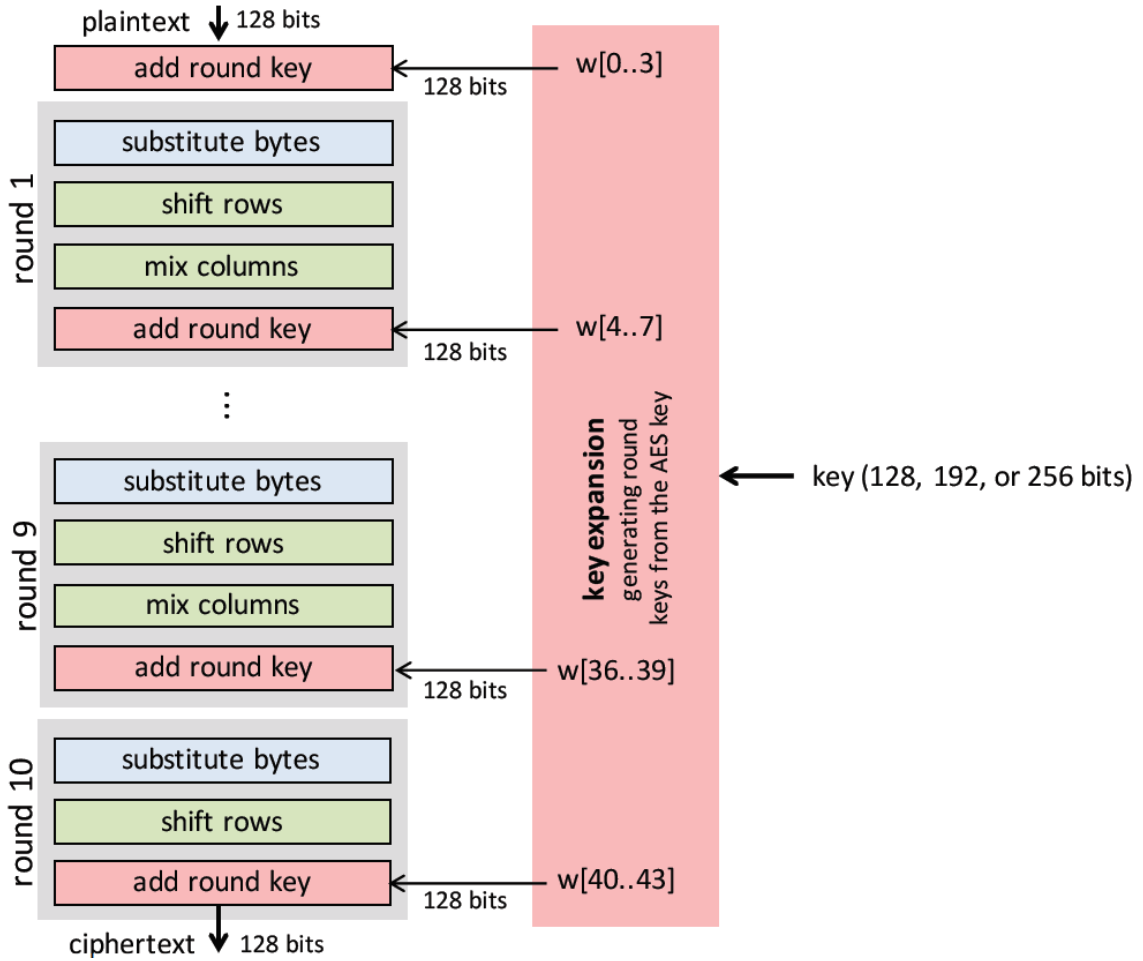
▯ = x74 + x20 = x54 = 'T'

BLOCK CIPHERS

Block ciphers

- Block ciphers operate on blocks of bits (typical block size is n = 128 bits)
- They are stateless (unlike stream ciphers)
- They cannot be efficiently distinguished from a random permutation
  - if K is unknown, the output is unpredictable (even parts of it, and even when some input-output pairs are known)
- Notation:
  - E(K, X) or EK(X) for encryption
  - $E_K^{-1}(Y)$ or DK(Y) for decryption
- Terminology
  - X – plaintext block (bit vector of length n)
  - Y – ciphertext block (bit vector of length n)
  - K – key (bit vector of length k)
  - E – encryption/encoding algorithm
  - D – decryption/decoding algorithm
- Examples: AES, DES (3DES), RC5, Twofish, Skipjack, ...

Advanced Encryption Standard (AES)



Block encryption modes

Basic modes

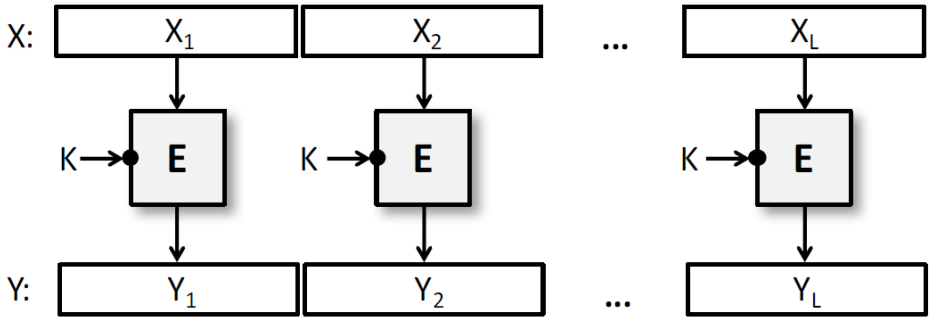
- Electronic Codebook (ECB) mode
- Cipher Block Chaining (CBC) mode
- Cipher Feedback (CFB) mode
- Output Feedback (OFB) mode
- Counter (CTR) mode

Other

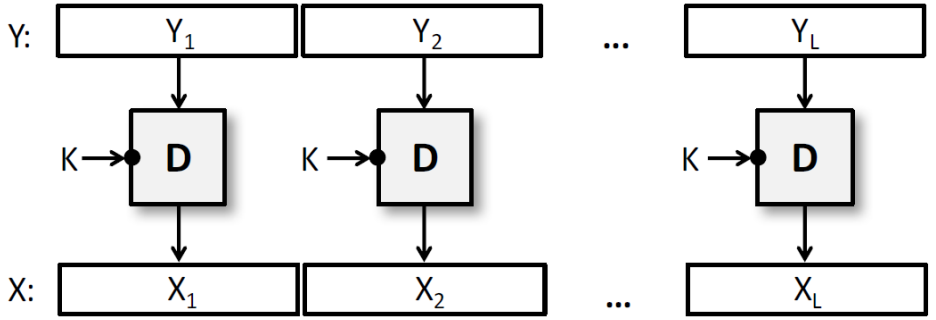
- Some special modes
  - XCBC
  - CBC with Ciphertext Stealing (CTS)
- Authenticated encryption modes
  - CCM: CTR + CBC MAC
  - GCM: Galois CTR mode
  - OCB: Offset Codebook Mode

ECB mode

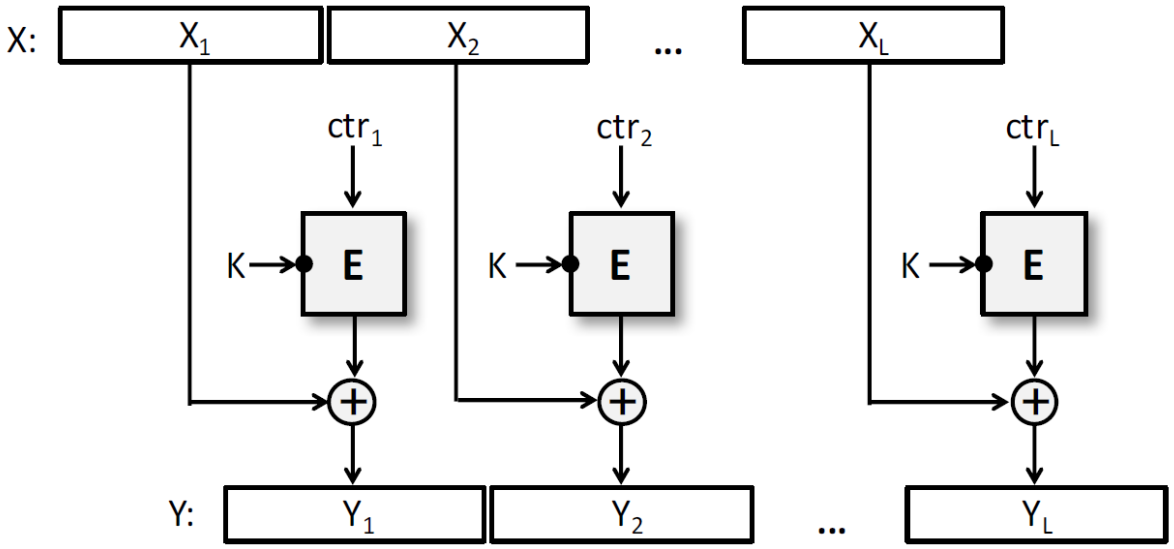
- encrypt:



- decrypt:

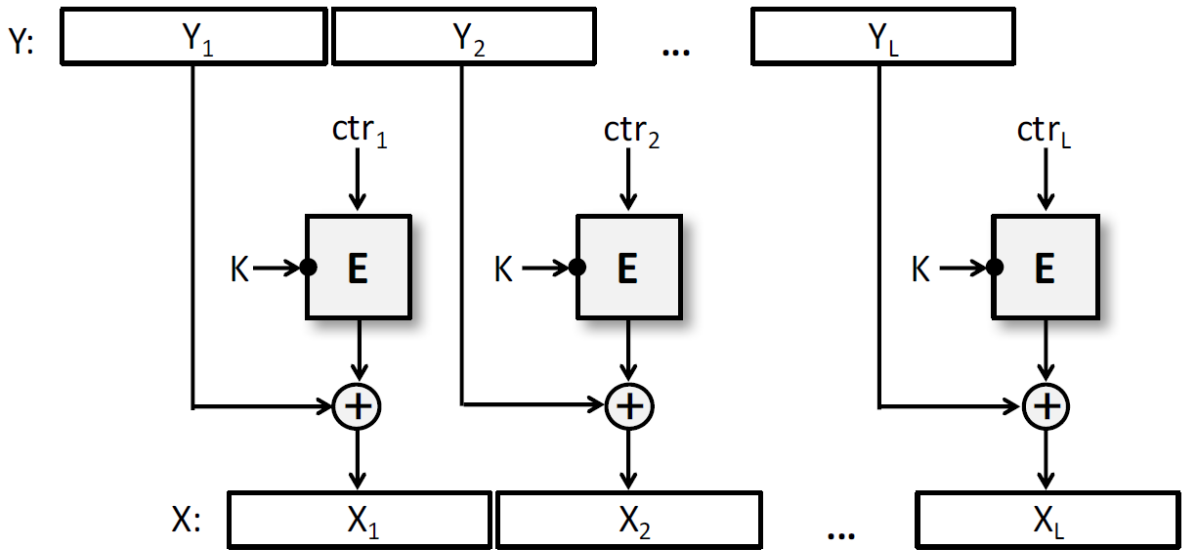


CTR mode (encryption)



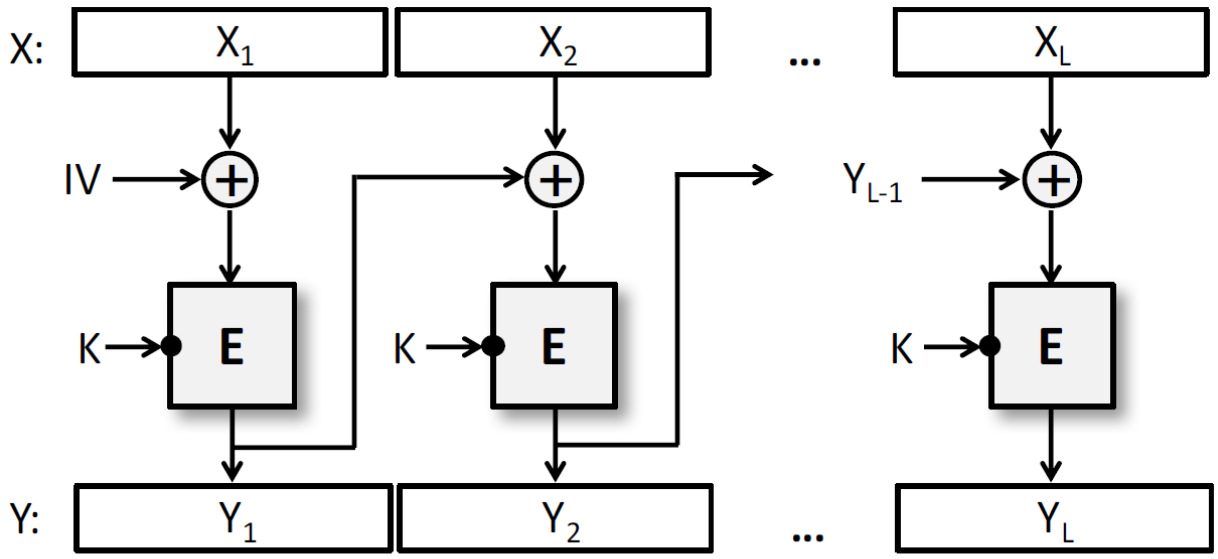
$$Y_i = X_i + E_K(ctr_i)$$
$$ctr_{i+1} = ctr_i + 1$$

CTR mode (decryption)



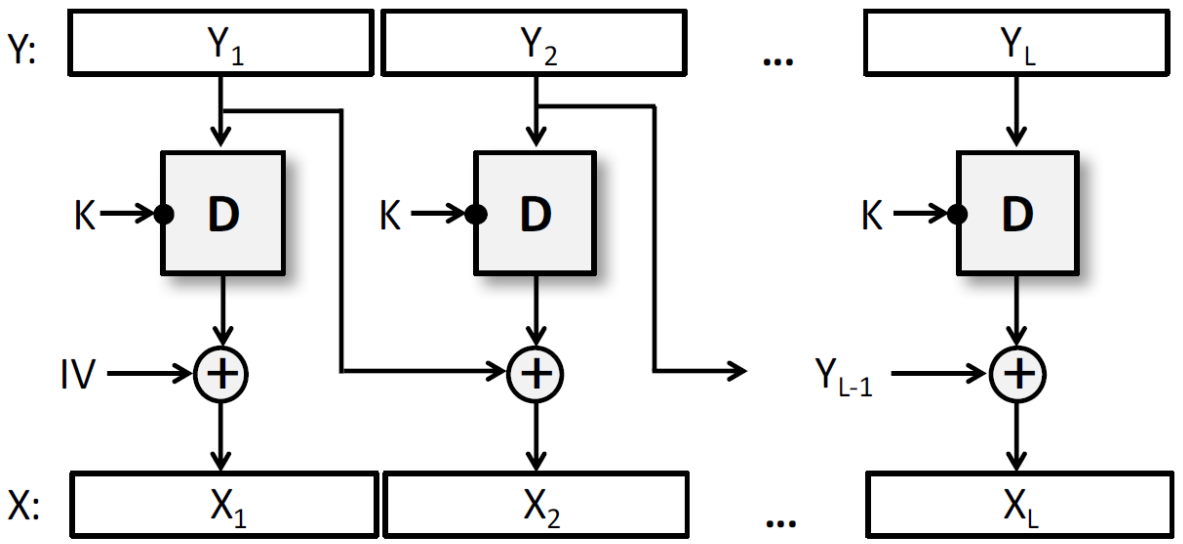
$$X_i = Y_i + E_K(ctr_i)$$
$$ctr_{i+1} = ctr_i + 1$$

CBC mode (encryption)



$$Y_i = E_K(X_i + Y_{i-1})$$

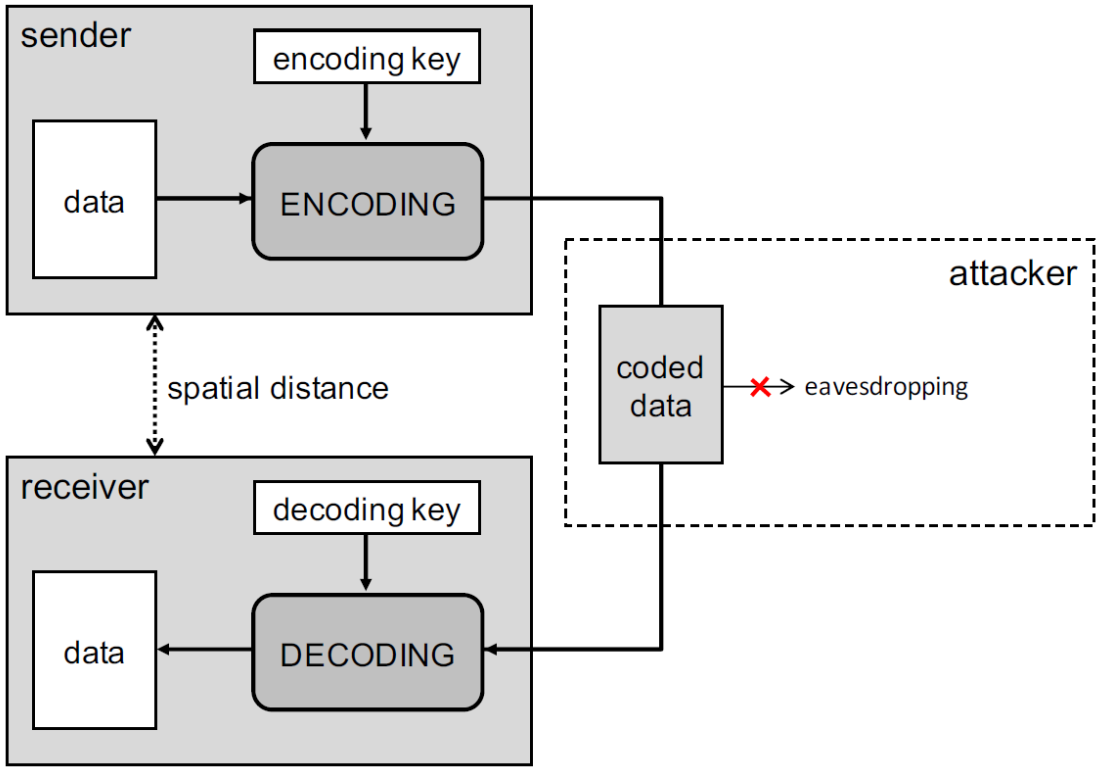
CBC mode (decryption)



$$X_i = D_K(Y_i) + Y_{i-1}$$

PUBLIC KEY CRYPTOGRAPHY

Model of asymmetric key encryption



decoding key ą **encoding key**

- decoding key is secret
- computing the decoding key from the encoding key is hard
- encoding key can be made **public** (public-key crypto)

Public-key encryption schemes

- Functions (algorithms) and terminology:
  - Key-pair generation function G( ) = (K+, K-)
    - K+ – public key
    - K- private key
  - Encryption function E(K+, X) = Y
    - X – plaintext
    - Y – ciphertext
  - decryption function D(K-, Y) = X
- Typically, the plaintext (and the ciphertext) consists of a few thousands bits → similar to block ciphers
- Examples: RSA, ElGamal

The (textbook) RSA cryptosystem

key-pair generation algorithm

- choose two large primes p and q (easy)
- n = pq, f(n) = (p-1)(q-1) (easy)
- choose e, such that 1 < e < f(n) and gcd(e, f(n)) = 1 (easy)
- compute the inverse d of e mod f(n), i.e., ed mod f(n) = 1 (easy if p and q are known)
- output public key: (e, n) (made public after key-pair generation)
- output private key: d (and p, q) (kept secret after key-pair generation)

Encryption & decryption

- encryption algorithm:
  - represent the plaintext message as an integer m Î [0, n-1]
  - compute the ciphertext c = me mod n
- decryption algorithm:
  - compute the plaintext from the ciphertext c as m = cd mod n
  - this works, because cd mod n = med mod n = mkf(n)+1 mod n = m mod n = m

Security of RSA

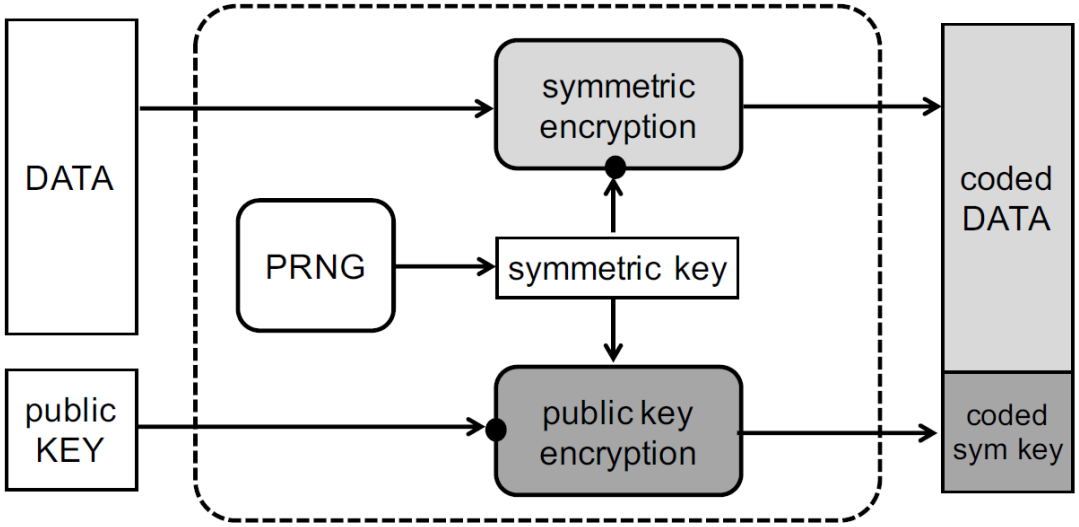
- factoring integers is believed to be a hard problem
  - given a composit integer n, find its prime factors
  - true complexity is unknown
  - it is believed that no polinomial time algortihm exists to solve it
- computing d from (e, n) is equivalent to factoring n
- computing m from c and (e,n) may not be equivalent to factoring n (this is known as the RSA problem)
  - if the factors p and q of n are known, then one can easily compute d, and using d, one can also compute m from c
  - we don't know if one could factor n, given that he can efficiently compute m from c and (e,n)
  - nevertheless, the RSA problem is believed to be a hard problem
- textbook RSA is not semantically secure (encryption is deterministic) and malleable (due to its homomorphic property)
  - in practice, textbook RSA needs to be extended with message formatting (PKCS #1)

RSA in practice – homomorphic property

- if m1 and m2 are two plaintext messages and c1 and c2 are the corresponding ciphertexts, then the encryption of m1m2mod n is c1c2 mod n
  - (m1m2)eş m1 e m2 e ş c1c2 (mod n)
- this leads to an adaptive chosen-ciphertext attack on RSA
  - assume that the attacker wants to decrypt c = me mod n
  - assume that an oracle will decrypt arbitrary ciphertext for the attacker, except c
  - the attacker can select a random number r and submit c × remod n to the oracle for decryption
  - since (c × re)d ş cd × red şm × r (mod n), the attacker will obtain m × r mod n
  - He/she then computes m by multiplication with r-1 (mod n)
- we say that textbook RSA is ***malleable***
  - can be circumvented by imposing some structural constraints on plaintext messages → see PKCS #1 formatting

Hybrid encryption

- public key crypto is slower than symmetric key crypto and require longer (e.g. 2048 bits) keys for similar security
- the sp
  encryp



Semantic security

- an adversary should not be able to choose two plaintexts X and X’ and later distinguish between the encryptions EK(X) and EK(X’) of these messages
  - note: symmetric-key block ciphers have this property
  - the problem with public-key encryption is that the adversary can compute EK(X) and EK(X’) using the public key K and trivially determine that EK(X) is the encryption of X and EK(X’) is the encryption of X’
- the solution is probabilistic encryption
  - computation of the ciphertext uses some random input àeven when the same message is encrypted twice, the outputs will be different
  - some public-key encryption schemes are probabilistic by design (e.g. ElGamal)
  - others need pre-formatting of messages which involves the addition of some randomness (e.g., RSA uses PKCS #1 formatting)

BLACK BOX HACKING

Black box hacking intro

- A little bit of theory: https://en.wikipedia.org/wiki/Black_box
- Access the Avatao platform on www.avatao.com
- Create a new account (if not created earlier)
- Find the “Cryptography” path in Avatao by searching for “crypto”
- Start the path
- Start the first challenge named “Black box crypto”

Black box hacking – Solution – 1

- Read the (Avatao-)suggested reading on Wikipedia: https://en.wikipedia.org/wiki/Black_box
- Download netcat from this link: https://eternallybored.org/misc/netcat/
- Connect to the remote server via raw TCP (IP addresses ar subject to change):

```
nc 63.32.77.56 32815
```

- Interact with the cryptographic system by manually entering plaintext messages and inspecting the responses:
  - Possible messages: 1, 11, 2, 22, A, AA, B, BB, “This is the plaintext”, T, Th, Thi, This, etc.

Black box hacking – Solution – 2

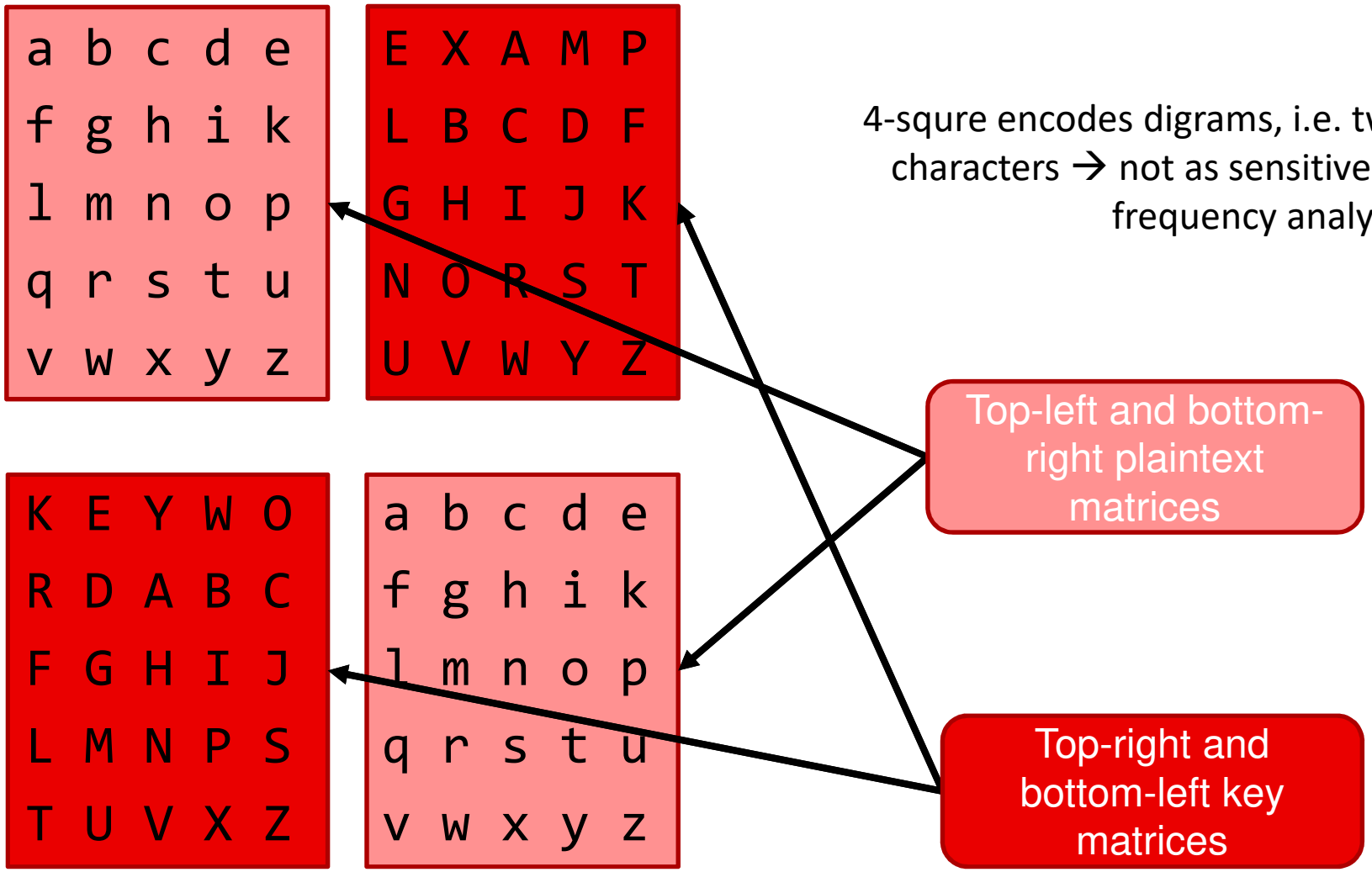
- Step #1: Notice that there is some padding going on by sending A, AA, AAA (or similarly formatted other messages)
- Step #2: Notice the pairwise character flipping by sending these messages: AAAA, BAAA, AAAB, AACA
- Step #3: Find out the encoding method for each 2-byte character by writing them down in binary and inspecting in binary the responses received from the remote server, e.g. A = 10000001, B = 10000010
- Step #4: Manually decode the first 4-5 characters
- Step #5: Write Python code to programmatically decode entire messages

Black box hacking – Solution in Python

```
cipher =
'4b795162425d5e1e1e757544726f6669797f7c63756f4249461e4f464d4475
4f425e534f5975585f75197d797e1b62696e6f5e751b424d4475595a5f4b754
8755e434b754b424b424d755e4f1b75155e2a57'

# Turn the ciphertext into bytes
cipher_bytes = ''
for i in range(  len(cipher) // 2  ):
    cipher_bytes += chr(int(cipher[i*2:i*2+2], 16));

# Decrypt the ciphertext
plaintext = ''
for i in range(len(cipher_bytes)//2):
    plaintext += chr(ord(cipher_bytes[i*2+1])^42)
    plaintext += chr(ord(cipher_bytes[i*2])^42)

print(plaintext)
```

Breaking the 4-square cipher

Code ‘cryptography’ with 4-square – The key!



Encode ‘cr’

a	b	c	d	e
f	g	h	i	k
l	m	n	o	p
q	r	s	t	u
v	w	x	y	z

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

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

a	b	c	d	e
f	g	h	i	k
l	m	n	o	p
q	r	s	t	u
v	w	x	y	z

Step 1: Row of plaintext ‘c’ in the top-left plaintext square and column of ‘r’ from the bottom-right plaintext square

‘cr’ → ‘XN’

Step 2: Row of plaintext ‘r’ in the bottom-right plaintext square and column of ‘c’ from the top-left plaintext square

Encode ‘yp’

a	b	c	d	e
f	g	h	i	k
l	m	n	o	p
q	r	s	t	u
v	w	x	y	z

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

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

a	b	c	d	e
f	g	h	i	k
l	m	n	o	p
q	r	s	t	u
v	w	x	y	z

Step 3: Row of plaintext ‘y’ in the top-left plaintext square and column of ‘p’ from the bottom-right plaintext square

‘yp’ → ‘ZI’

Step 4: Row of plaintext ‘p’ in the bottom-right plaintext square and column of ‘y’ from the top-left plaintext square

4-square task

- The inspector of the military camp gets a letter from the second in command:
  Uqzwhppahp pzvwp tq outcdv pawaqqzk zaqzaaa. Luweaav ptn oivxazzovupwq aaq slv pqzxwq az rtn snpxqwpxspv. Ptn obovqppwz xp upan xqzensr saxqzi ee nrxywkvqvv vk sli pwvqtlpqearlp. Cl pq vq weywvwp, ttnz rtn negeywywp zqpqrosvlaa qrwqv upxwvl oe vqienv lz rtn etcwvlaa vwixqvvwp ttn „Giwnyzi" vliivvy cslc. Slo vawqeznosq ls xzq imvcgiizv ptnnepsne slv yvuv vp ievlunqne yq yweonouzywvuo.B

- The inspector knows that the encrypted text was generated with the four-square cipher. Unfortunately he lost his four-square matrices which could decrypt it. Help him to solve this problem!
- Hints:
  - Wikipedia direct link: https://en.wikipedia.org/wiki/Four-square_cipher
  - The plaintext squares contain the standard alphabet (letters are in the right
    order) and both i and j are in the same location (to reduce the alphabet size
    to 25). You may also find this 4square-help.pdf useful.

ADDITIONAL CHALLENGES

Avatao

- The “Cryptography” path → first two challenges available with a free subscription
- The “Mastering Cryptographic Engineering” with 26 challenges
- Elements of the “CrySyS Student Exercises” path
- Elements of the “CrySyS SecChallenge 2014” path – freely available!
- Elements of the “CrySyS SecChallenge 2015” path – freely available!
- Elements of the “CrySyS SecChallenge 2019” path

Summary

- Cryptography intro and history
  - Stream ciphers
  - Block ciphers
  - Public key crypto
- Black-box hacking
- Breaking the 4-square cipher

Thank you for your attention!