Introduction to Cryptography

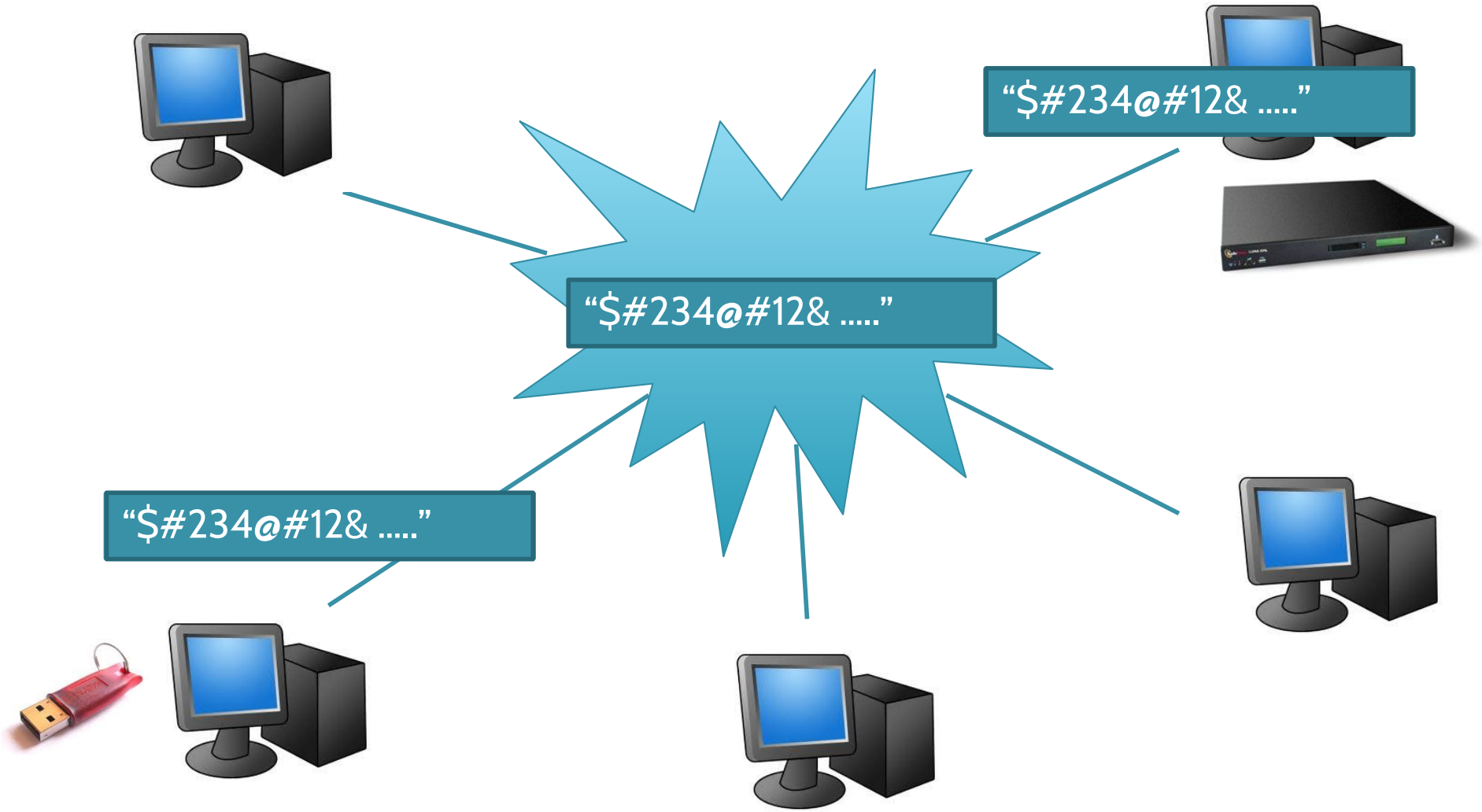
Riccardo Focardi

Università
Ca'Foscari
Venezia

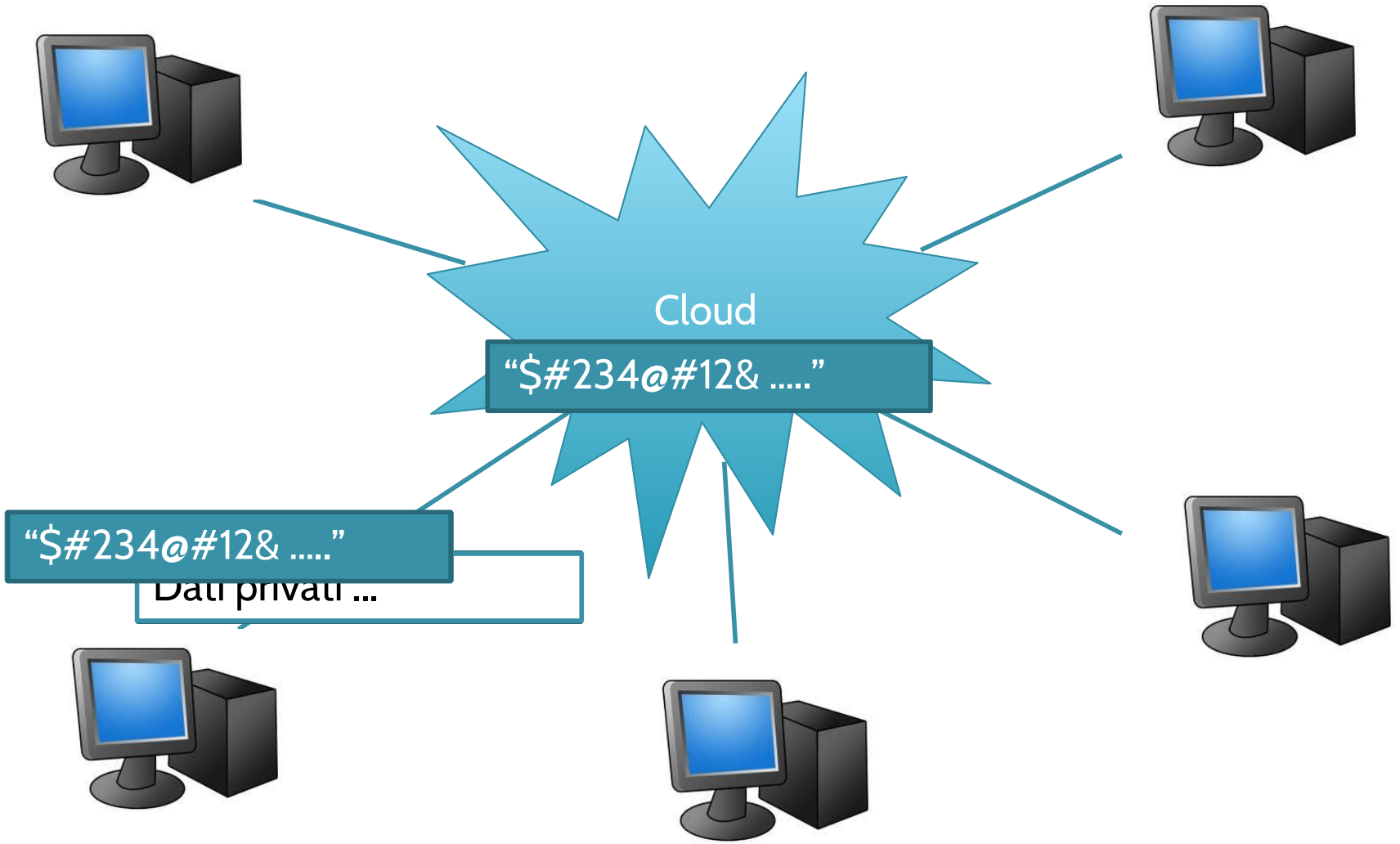
Università Ca' Foscari Venezia
www.dais.unive.it/~focardi
secgroup.dais.unive.it

Cryptosense
cryptosense.com

End to end security



Secure storage in the cloud



Classic cryptography

A message is transformed so to make it hard to understand it

Caesar cipher: every letter is replaced by the one which is three positions next in the alphabet

Brute force and cryptanalysis

Caesar cipher only has 26 possible variants: we can **try them all!**

Moreover, equal letters are **encrypted in the same way**: it is easy, for example, to spot vowels and double letters.

Example 1: Caesar cipher

RPTHPGRXEWTGDCANWPHILTCINHXMKPGXPCIHIWP
ILTRPCTPHXANQGJITUDGRT

Brute forcing

We try all the 26 possibilities until we find something that makes sense in English:

```
for k in range(26):
  chiave = alfabeto[k:] + alfabeto[:k]
  print (k, d('RPTHPGRXEWTGDCANWPHILTCINHXMKPGXPCIHIWPILTRPCTPHXANQGJITUDGRT',chiave))
...
0 RPTHPGRXEWTGDCANWPHILTCINHXMKPGXPCIHIWPILTRPCTPHXANQGJITUDGRT
1 QOSGOFQWDVSFCBZMVOGHKSBHMGWLJOFWOBHGHVOHKSQOBSOGWZMPFIHSTCFQS
...
15 CAESARCIPHERONLYHASTWENTYSIXVARIANTSTHATWECANEASILYBRUTEFORCE
...
25 SQUIQHSYFXUHEDBOXQIJMUDJOIYNLQHYQDJIJXQJMUSQDUQIYBORHKJUVEHSU
```

Example 2: substitution cipher

HMBFSFNCXZEFNBHCWPSHMZFXCNMPNFSYPWWPNKHMODXWSKQMFEFNZHMHFZOCNMMPOHKLWCDQZFLSNCBBPKHC
DKKFYPNOFWPSHPNKHMWPQSKWCQKLZFLSFICLHFZIMMNBPKHHPEIDKFWKMQWCQKLZFLSBPKHCDKFWPSHKHMICLSAF
QMNKSHFZKCXZHPEKHFKKHMCXZEFNBFSNCBZMWPNPKMXLFNZWPNFXXLSFXFCBHPTHPSKHMBCQSKWCQECWDNXDTYLF
NZKHMICLHFZOCNMFKKHMPQCQZMQSPNFNCKHMQICFKBHPTHTFDOHKKHQMMOCCZWPSHKHMWPQSKBMMYPKEFZMKH
MICLSFZKCSMMKHMCXZEFNTCEMPNMFTHZFLBPKHHPSSYPWWMEAKLFNZHMFXBFLSBMNKZCBNKCHMXAHPETFQQLMPKH
MQKHMTCPXMZXPNMSCQKHMOFWWFNZHFQACCNFNZKHMSFPXKHFKBFSWDQXMZFQCDNZKHMEFSKKHMSFPXBFSAFKTH
MZBPKHWXCDQSFTYSFNZWDQXMZPKXCCYMZXPYMKHMWXFOCWAMQEFNMNKZMWMFKKHMCXZEFNBFSKHPNFNZOFDN
KBPKHZMMABQPNYXMSPNKHMIFTYCWHPSNMTYKHMIQCBNIXCKTHMSCWKHMIMNMJCXMNKSYPNTFNTMQKHMSDNIQPNO
SWQCEPKSQMWXMTKPCNCNKHMKQCAPTSMFBMQMCNHPSTHMMYSKHMIXCKTHMSQFNBMXXZCBNKHMSPZMSCWHPSWF
TMFNZHPSHFNZSHFZKHMZMMATQMFSMZSTFQSWQCEHFNZXPNOHMFJLWPSHCNKHMTCQZSIDKNCNMCWKHMSMSTFQSB
MQMWQMSHKHMLBMQMFSCXZFSMQCSPCNSPNFWPSHXMSSZMSMQKMJMQLKHPNOFICDKHPEBFSCXZMVTMAKHPSMLMS
FNZKHMLBMQMKHMSFEMTCXCQFSKHMSMFFNZBMQMTHMMQWDXFNZDNZMWMFKMZSFNKPFOCKHMICLSFPZKCHPEFSK
HMLTXPEIMZKHMIFNYWQCEBHMQMKHMSYPWWBFSHFDXMZDAPTCDXZOCBPKHLCDFOFPNBMJMEFZMSCEMECNMLKHMC
XZEFNHFZKFDOHKKHMICLKCWPSHFNZKHMICLXCJMZHPENCKHMCXZEFNSFPZLCDQMBPKHFXDTYLICFKSKFLBPKHKHMEIDK
QMEMEIMQHCBLCDBMNKMPOHKLSMJMNZFLSBPKHCDKWPSHFNZKHMNBMTFDOHKIPOCNMSMJMQLZFLWCQKHQMMBMM
YSPQMEMEIMQKHMCXZEFNSFPZPYNCBLCDZPZNCKXMFJMEMIMTFDSMLCDZCDIKMZPKBFSAFAFEFZMEMXMFJMPFEFICLFNZ
PEDSKCIMLHPEPYNCBKHMCXZEFNSFPZPKPSRDPKMNCQEFXHMHFSNKEDTHWFPKHNCKHMCXZEFNSFPZIDKBMHFJMHFJMN
KBMLMSKHMICLSFPZTFNPCWWMQLCDFIMMQCNKHMKMQQFTMFNZKHMNBMXXKFYMKHMSKDWWHCEMBHLNCKKHMCXZE
FNSFPZIMKBMMNWPSHMQEMNKHMLSFKCNKHMKMQQFTMFNZEFNLCWKHMWPSHMQEMNEFZMWDNCWKHMCXZEFNFNZ
HMBFSNCKFNOQLCKHMQSCWKHMCXZMQWPSHMQEMNXCCYMZFKHPEFNZBMQMSFZIDKKHMLZPZNCKSHCBPKFNZKHMLS
ACYMACXPKMXLFICDKKHMTDQQMNKFNZKHMZMAKHSKHMLHFZZQPWKMZKHMPQXPNMSFKFNZKHMSKMFZLOCCZBMFKH
MQFNZCWBHFKKHMLHFZSMMNKHMSDTTMSSWDXWPSHMQEMNCWKHFKZFLBMQMFXQMFZLPNFNZHFZIDKTHMQMZKHM
PQEFQXPNCDKFNZTFQQPMZKHMEXFPZWDXXXMNOKHFTQCSSKBCAXFNYSBPKHKBCEMN

Cryptanalysis of substitution ciphers

Substitution ciphers use random **alphabet permutations**

ABCDEFGHIJKLMNOPQRSTUVWXYZ

KZBARCQHSMNIUWVPJGEOTFDXLY

Since there are

$26! = 403291461126605635584000000 \approx 2^{88}$

permutations, we cannot try all of them.

However we can break the cipher through statistical analysis and a dictionary. Try here.

What is a cipher?

A cipher is defined through two functions

1. **Encryption**: given a **plaintext** and a **key** returns a **ciphertext**

$$E_{K1}(X) = Y$$

2. **Decryption**: given a **ciphertext** and a **key** returns a **plaintext**

$$D_{K2}(Y) = X$$

Symmetric and asymmetric ciphers

Decrypting the encryption of X we obtain X:

$$D_{K2}(E_{K1}(X)) = X$$

When K1=K2 we have a **symmetric key cipher**

When K1≠K2 we have an **asymmetric key cipher**

Security: it should be **unfeasible** to compute **X** or **K2** from Y even knowing other pairs (X_1,Y_1), … , (X_n,Y_n)

Modern cryptography: AES

Modern ciphers are very complex and use keys of at least 128 bits:

about **$3.40282367 \times 10^{38}$** different keys

Example: we can use openSSL to experiment:

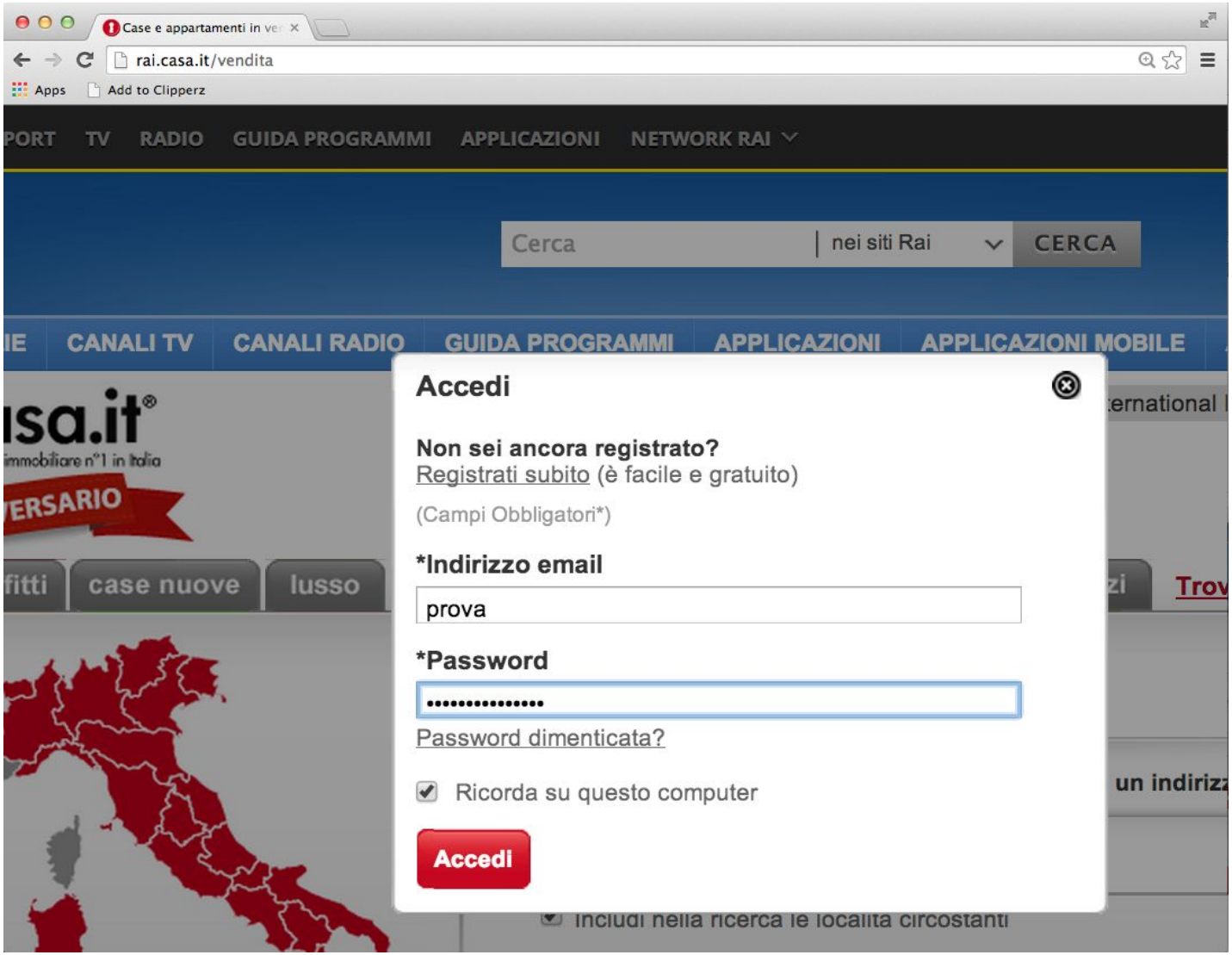
```
$ openssl rand -hex 16
ca8b7f7e66ab27302f7527df300f0fdf
$  ca8b7f7e66ab27302f7527df300f0fdf | hexdump
0000000 e3 d4 af 79 69 fa 02 31 db 58 2a f5 e3 33 13 1e
0000010
```

Key size

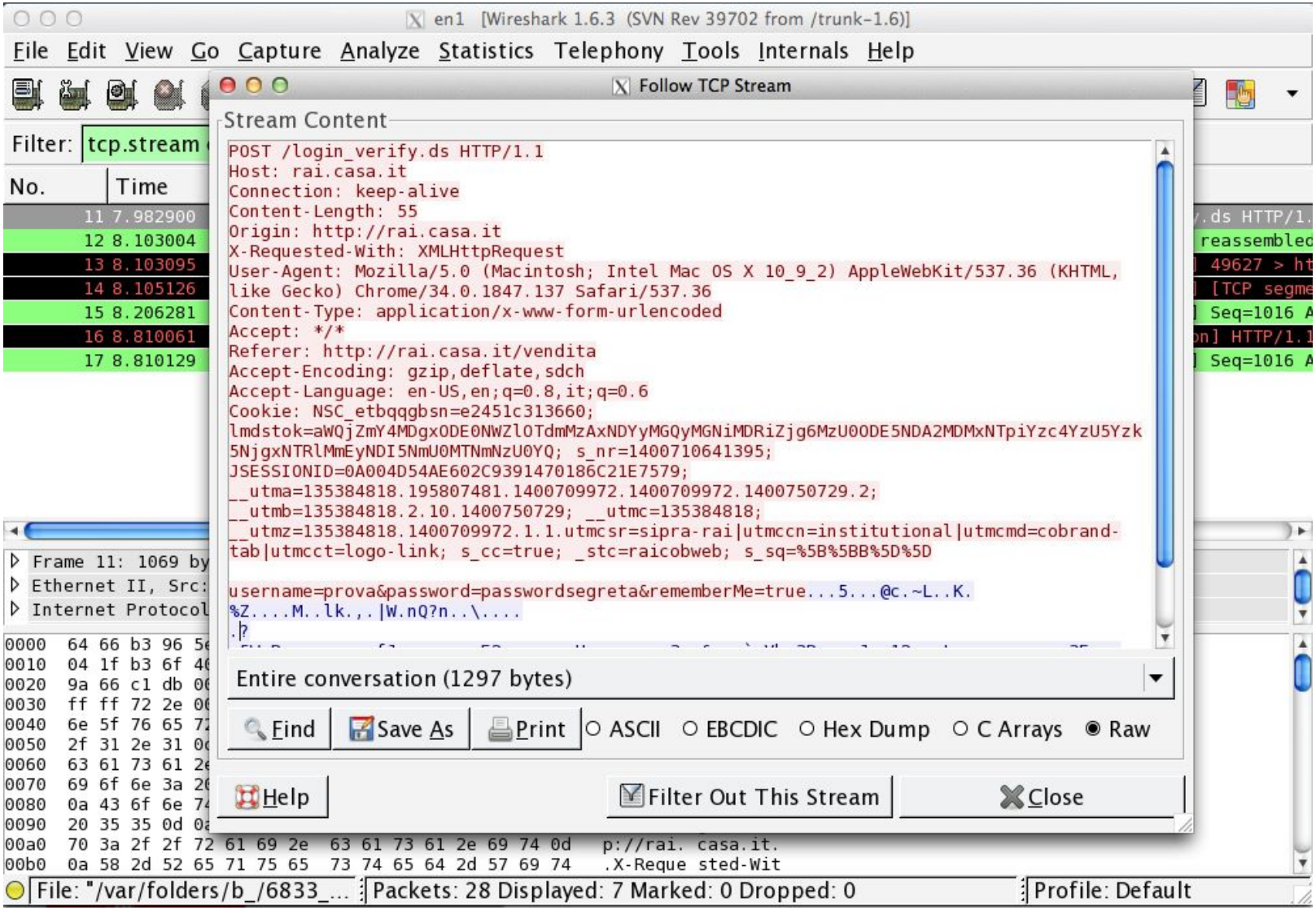
In the ENISA report 2014 we find the following:

Cipher	legacy	near term	long term
Symmetric key	80	128	256
RSA	1024	3072	15360

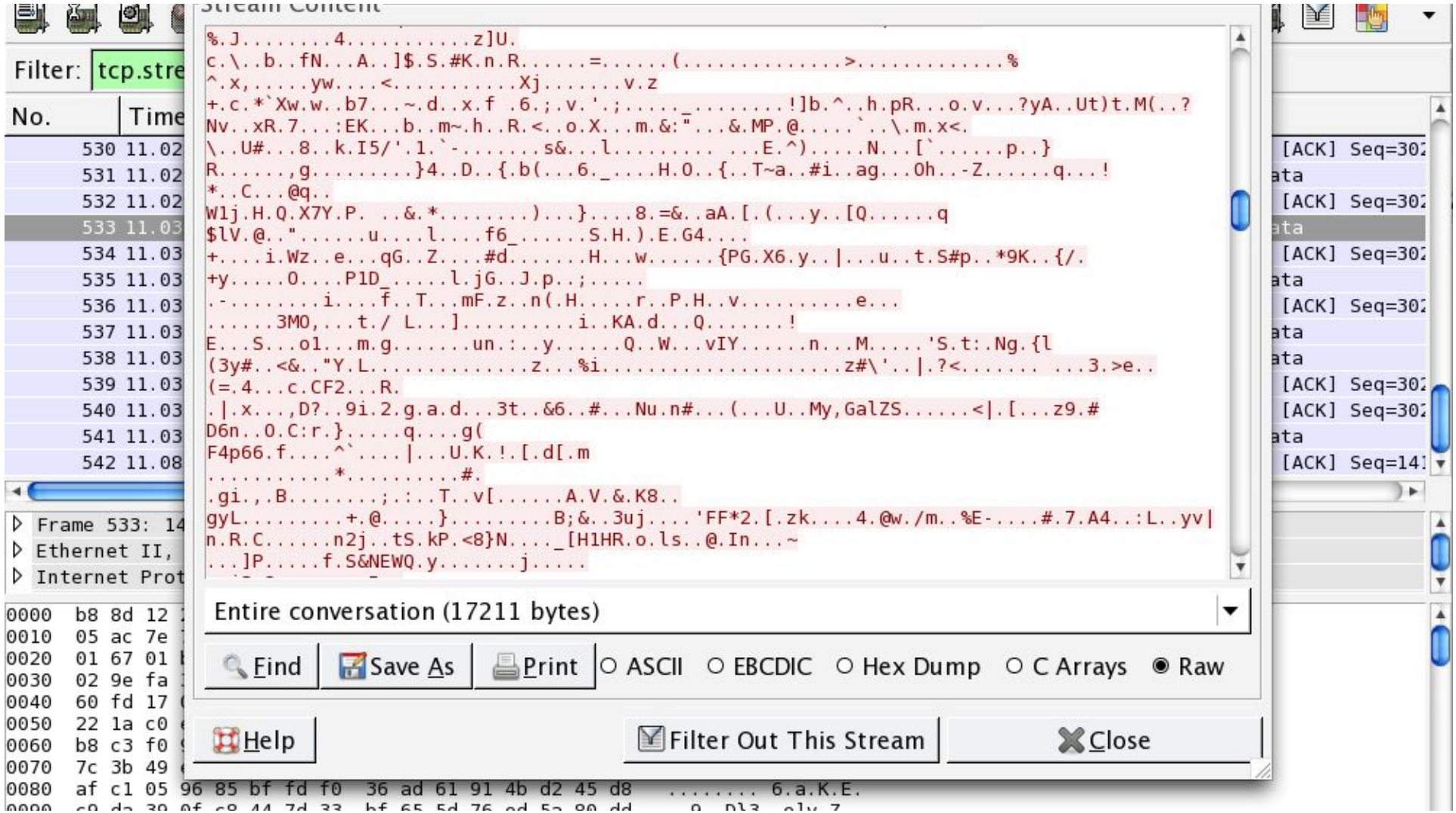
Cryptography on the web



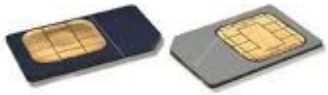
http: no protection!

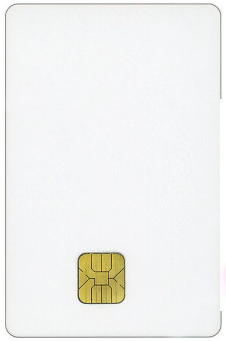


https: communication is encrypted



Cryptography ... of things!





Cryptography in banks

Payments, ATMs, money transfers, ...

Hardware Security Module (HSM)

Costs about 5k-20k € for a market of 200M € a year

… but things can go wrong

Many attacks on cryptographic systems in the last years:

- R. Verdult, F. D. Garcia and B. Ege. **Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer**. USENIX Security 2013
- R. Bardou, R. Focardi, Y. Kawamoto, L. Simionato, G. Steel, J. Tsay. **Efficient Padding Oracle Attacks on Cryptographic Hardware**. CRYPTO 2012
- M. Bortolozzo, M. Centenaro, R. Focardi, G. Steel. **Attacking and fixing PKCS#11 security tokens**. ACM CCS 2010
- F. D. Garcia, P. van Rossum, R. Verdult and R. Wichers Schreur. **Wirelessly Pickpocketing a Mifare Classic Card**. IEEE S&P 2009

Smartcards and crypto tokens

Brand	Device Model	Supported Functionality s	as	cobj	chan	w	ws	Attacks found wd	rs	ru	su	Tk
Aladdin	eToken PRO	✓	✓	✓	✓	✓	✓	✓				wd
Athena	ASEKey	✓	✓	✓								
Bull	Trustway RCI	✓	✓	✓	✓	✓	✓	✓				wd
Eutron	Crypto Id. ITSEC		✓	✓								
Feitian	StorePass2000	✓	✓	✓	✓	✓	✓	✓	✓	✓		rs
Feitian	ePass2000	✓	✓	✓	✓	✓	✓	✓	✓	✓		rs
Feitian	ePass3003Auto	✓	✓	✓	✓	✓	✓	✓	✓	✓		rs
Gemalto	SEG		✓		✓							
MXI	Stealth MXP Bio	✓	✓		✓							
RSA	SecurID 800	✓	✓	✓	✓				✓	✓	✓	rs
SafeNet	iKey 2032	✓	✓	✓		✓						
Sata	DKey	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	rs
ACS	ACOS5	✓	✓	✓	✓							
Athena	ASE Smartcard	✓	✓	✓								
Gemalto	Cyberflex V2	✓	✓	✓		✓	✓	✓				wd
Gemalto	SafeSite V1		✓		✓							
Gemalto	SafeSite V2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	rs
Siemens	CardOS V4.3 B	✓	✓	✓		✓				✓		ru

Real attacks!

20 February 2013

35 000 000 € stolen from ATMs in **less than 10 hours**

People think crypto look like this …

… but it is more like this!

16th Century, Citadel of Dinant, Belgium.

Photo © *2016* Ben Heine

Cryptographic vulnerabilities

Crypto mechanisms **are not** equally secure

Vulnerabilities in applications can reveal keys or downgrade to less secure mechanisms

Improvements in **technology** and **cryptanalysis** require better crypto

The configuration and management of cryptographic systems is complex and error prone

Heartbleed

Vulnerability in OpenSSL, the protocol underneath https

An *over-read* allows for accessing process memory where **server keys** are stored

Once those keys are leaked it is possible to mount a **MITM attack** and intercept the whole Web session

http://heartbleed.com/

Modes of operation

Needed when:

- Data is bigger than the block size
- We need to encrypt a stream with a block cipher

Example: AES ECB

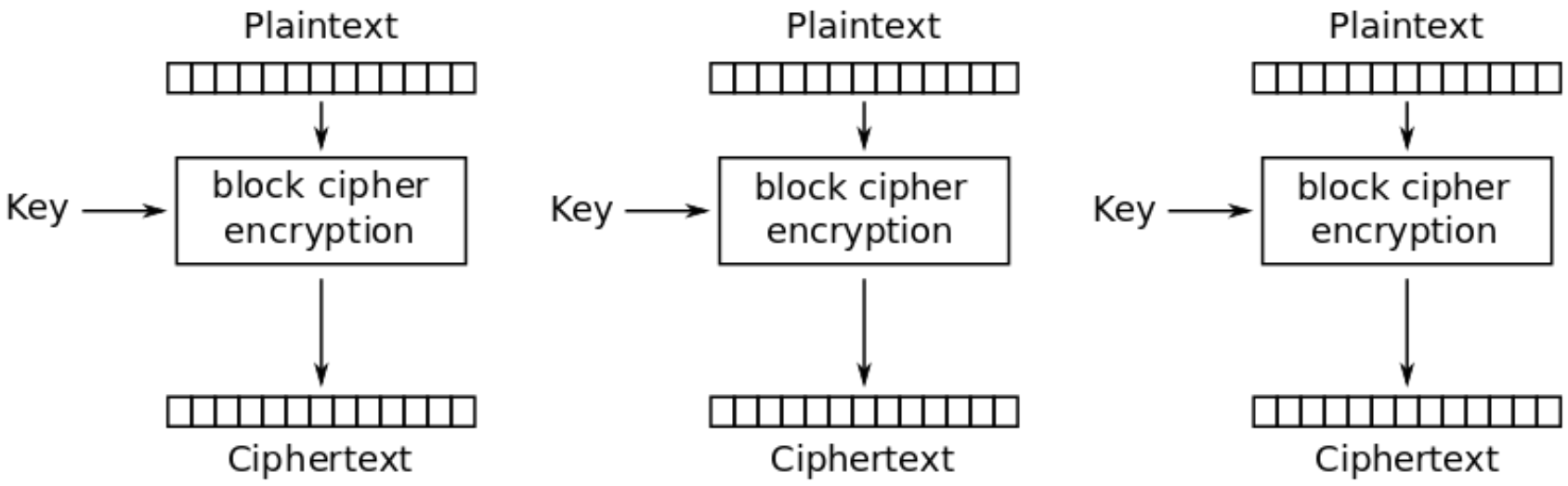
ECB is a mode of operation that splits long messages into blocks of 16 bytes (the size of AES block)

Blocks are encrypted **independently** under the same key

... not so different from substitution ciphers!

```
openssl enc -e  -aes-128-ecb  -K deccbe6da45d0d0fc57aad310d934ffe -in
LogoBig-tail.ppm -out LogoBig-tail-enc.ppm
```

ECB



Electronic Codebook (ECB) mode encryption

Poor confidentiality and integrity

The unsatisfactory result

Università
Ca'Foscari
Venezia

plaintext

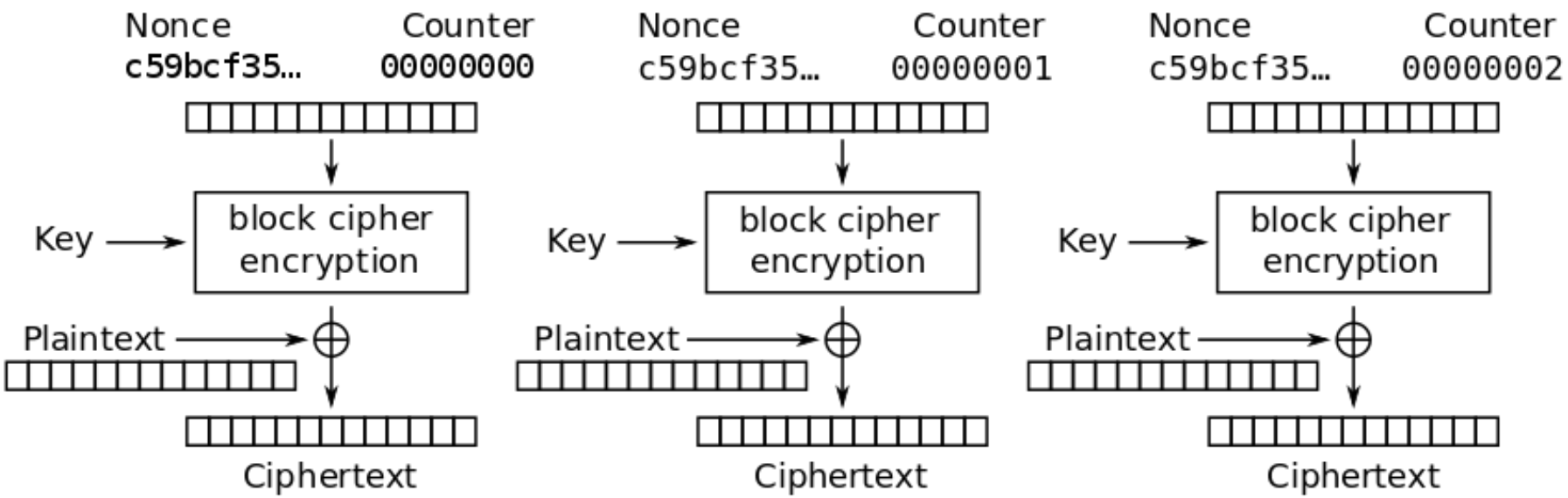
ciphertext

Chosen plaintext attack in ECB

If an attacker can prepend arbitrary prefix to the plaintext he can bruteforce blocks byte after byte

- prepend 15 known bytes
- bruteforce byte 16
- iterate over all bytes

CTR: stream ciphers



Counter (CTR) mode encryption

Nonce is fundamental for security!

Challenge

ciphertext 1:

```
8f079a817d1dfa5bb2b1e069b0f4027abc65db6d130e6f3c154611d
165d66b0a23424734790df0769cc3c4f4f289e784ac0cc5cab7e47c
5c1a
```

ciphertext 2:

```
9f0a92807d33fb1ab7a9ad36e5cd4064a320da7a56122e21004c42c
46d93214b28595b777612e46c9dc3c4eefedde88ee31c97c1b1e834
135c
```

Leaked plaintext:

```
Dear Graham, I'll be happy to participate in the
training
```

A CTR with fixed nonce has been used

... how would you break the other ciphertext?

Solution

P1, P2 plaintexts and C1, C2 corresponding ciphertext

Same nonce means same key K

$P1 \oplus K = C1$

$P2 \oplus K = C2$

thus

$P1 \oplus P2 = C1 \oplus C2$

$P2 = P1 \oplus C1 \oplus C2$

Padding oracle attacks

An attack that exploits **padding errors**

We have a padding oracle when

1. an application **exhibits padding errors** while decrypting a ciphertext
2. the **attacker can choose the ciphertext** (chosen ciphertext attack)

Example: key unwrapping in security devices

PKCS5 / PKCS7 Padding

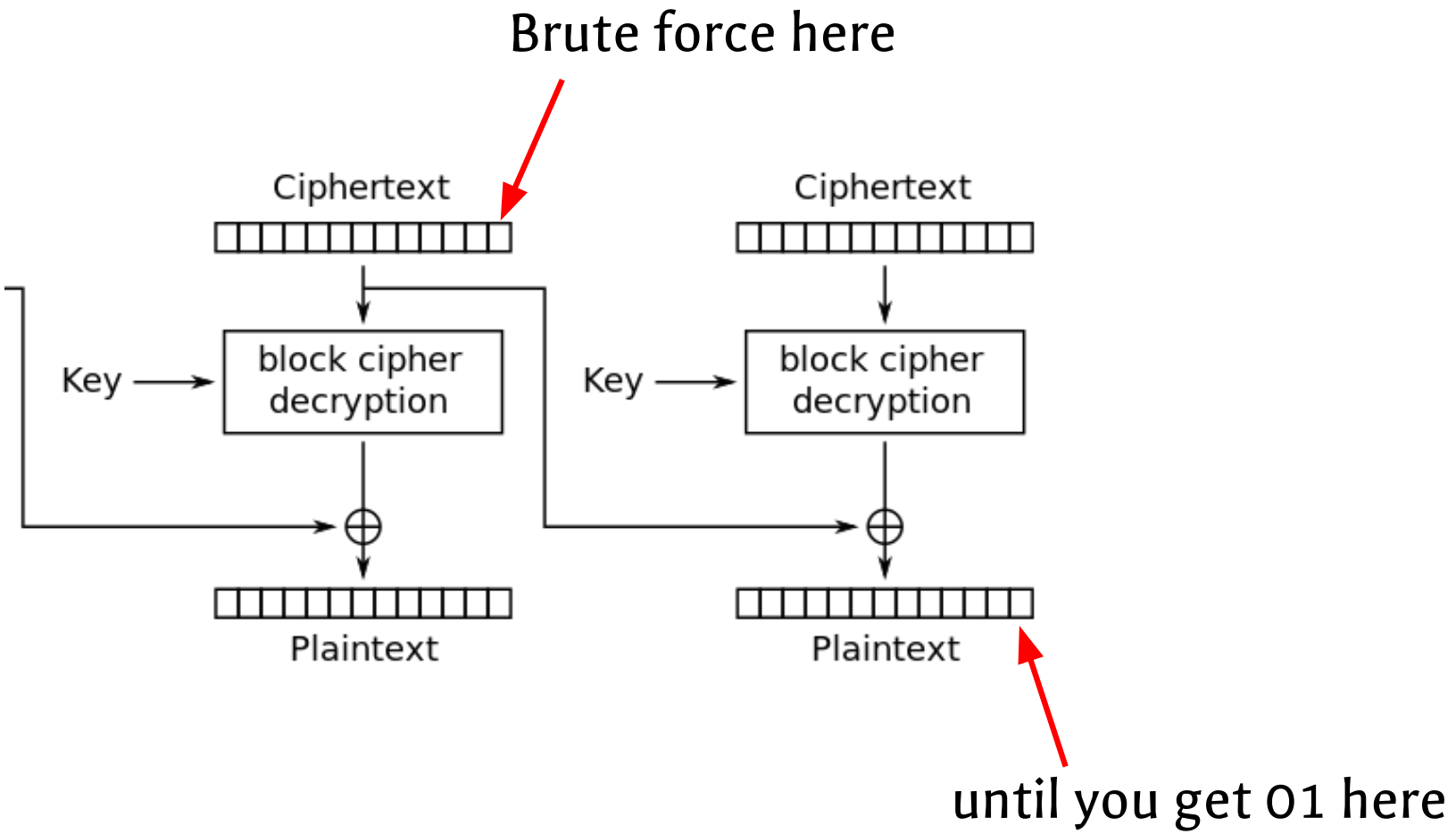
if we need 5 bytes we add `05 05 05 05 05`

Possible paddings:

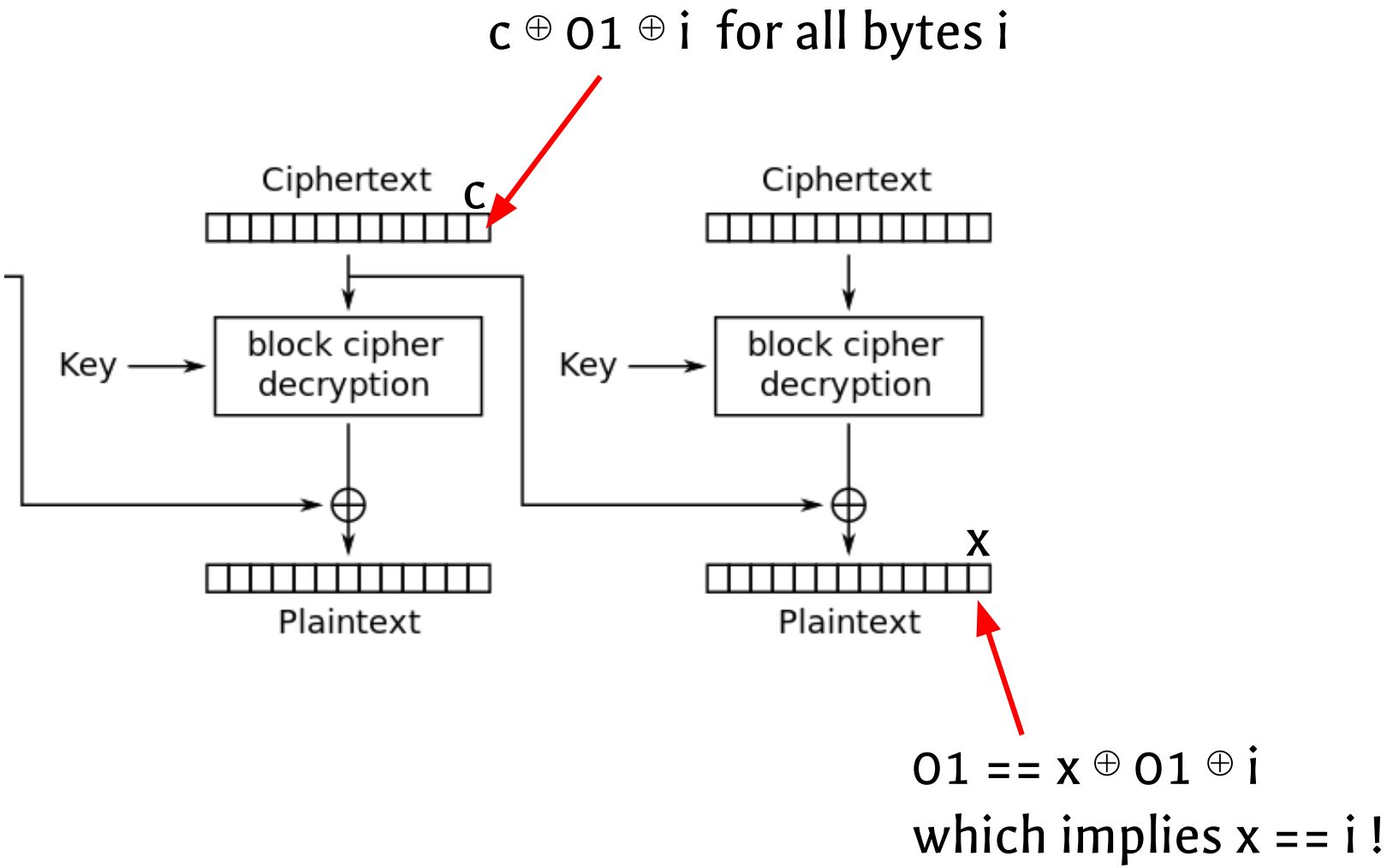
```
01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
...
```

(PKCS5 is for 8 byte block size only)

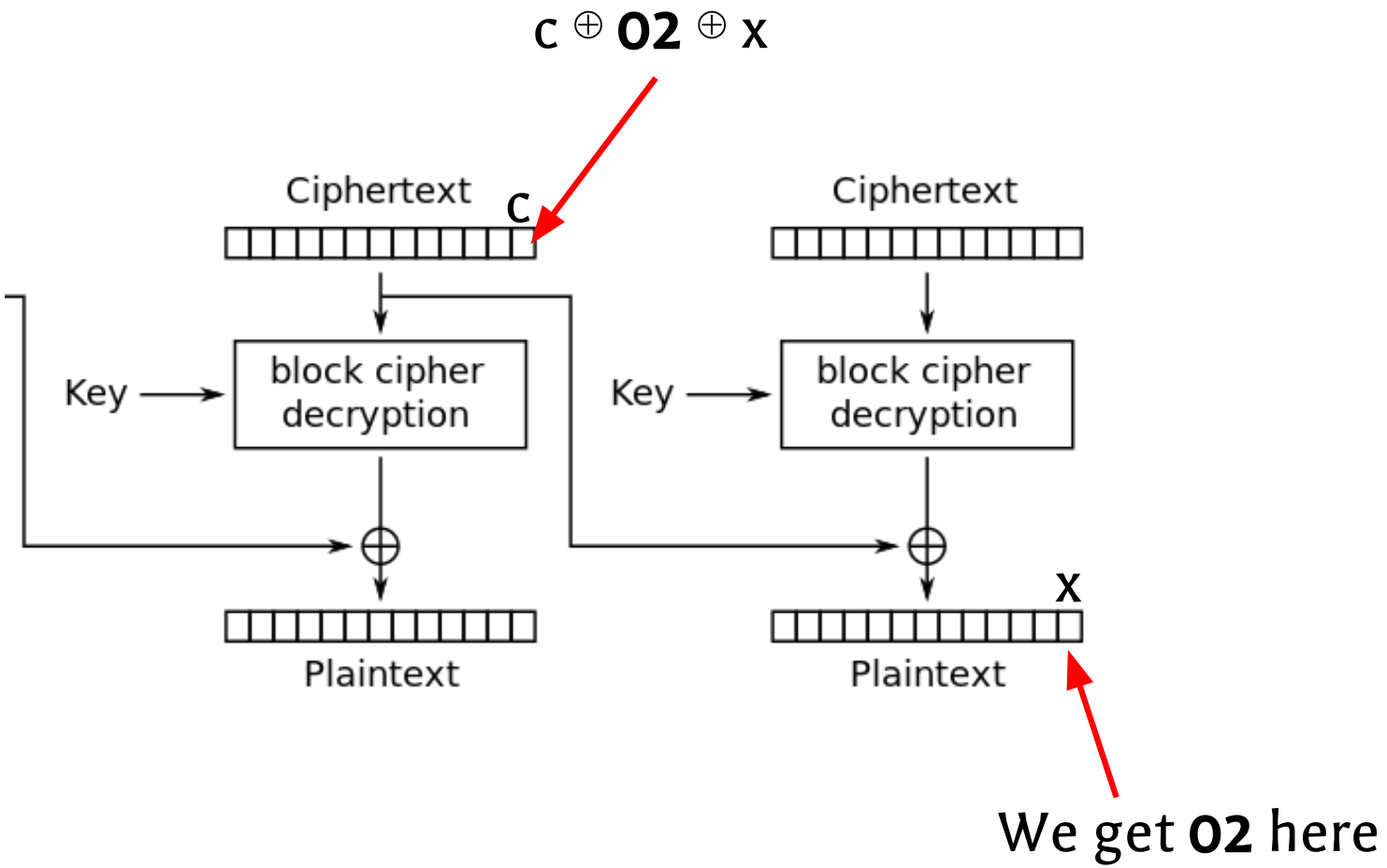
Padding oracle attack on CBC



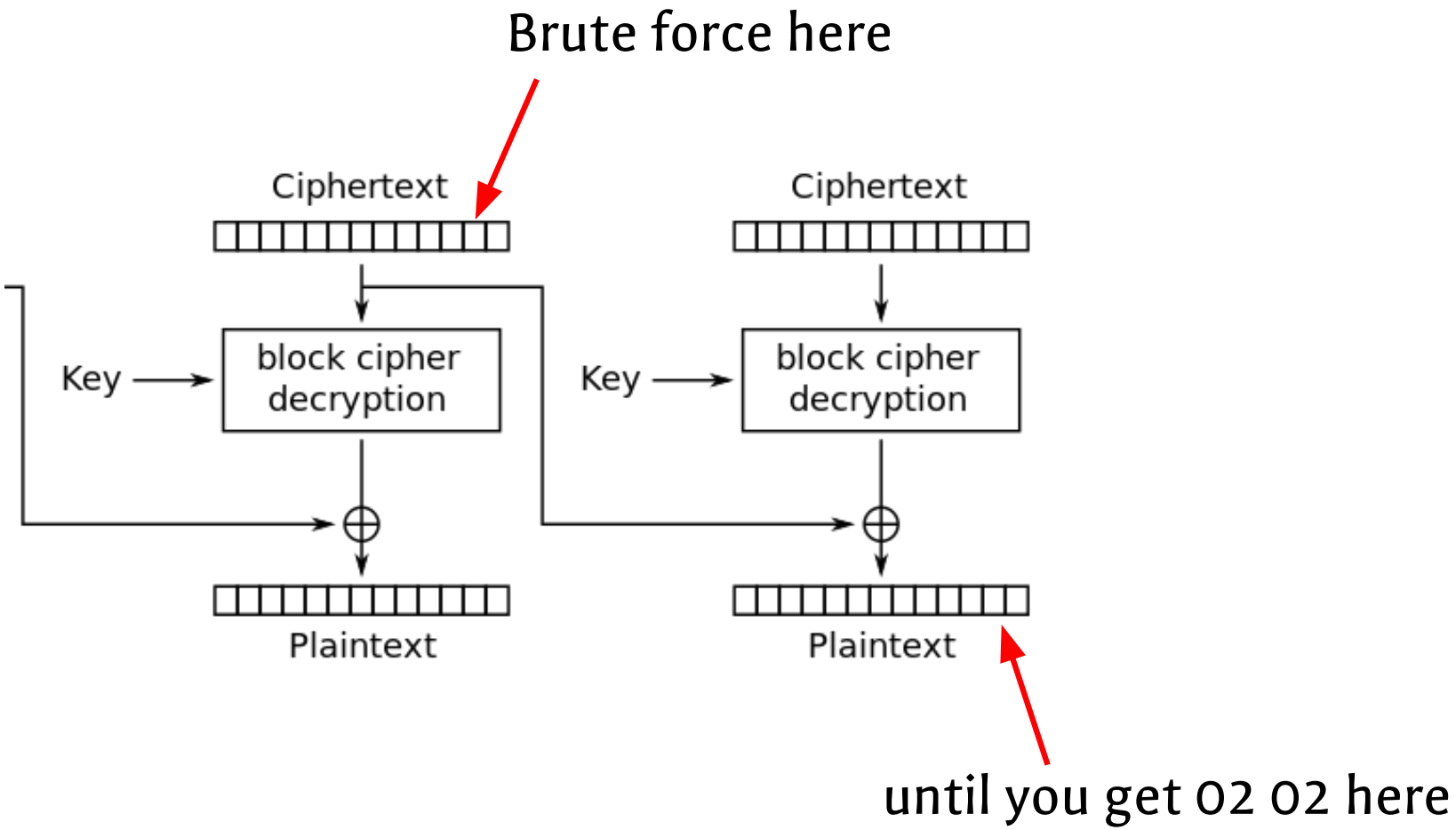
Padding oracle attack on CBC



Iterating for the next byte

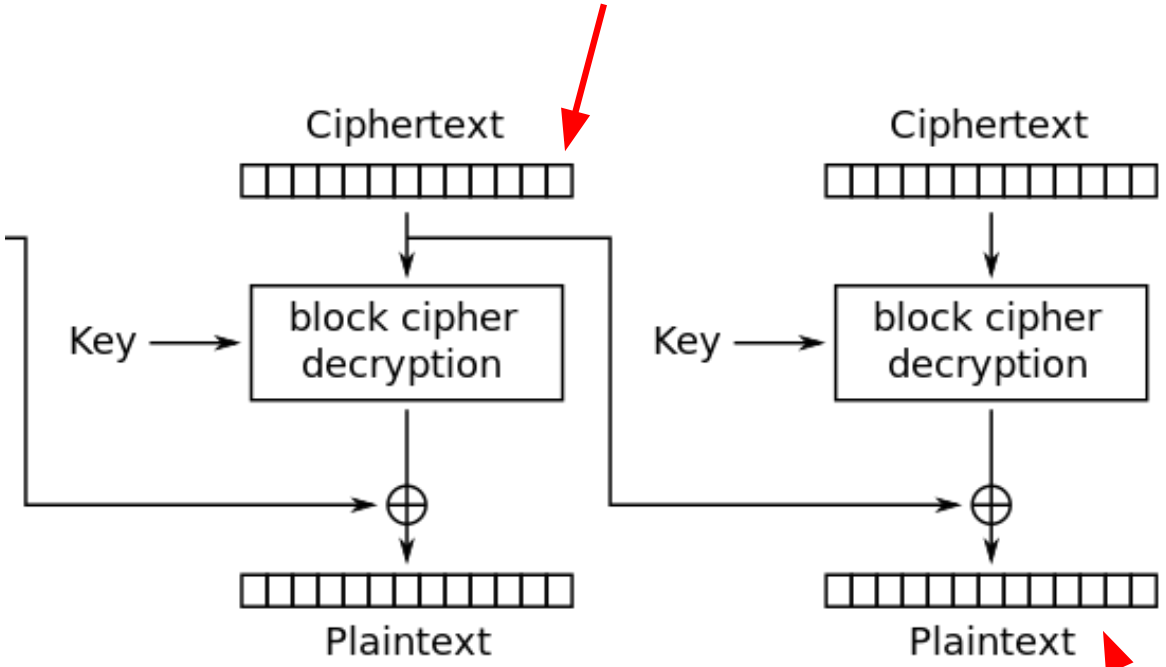


Iterating for the next byte



What if it is correctly padded?

Brute force: we get two “yes” answers

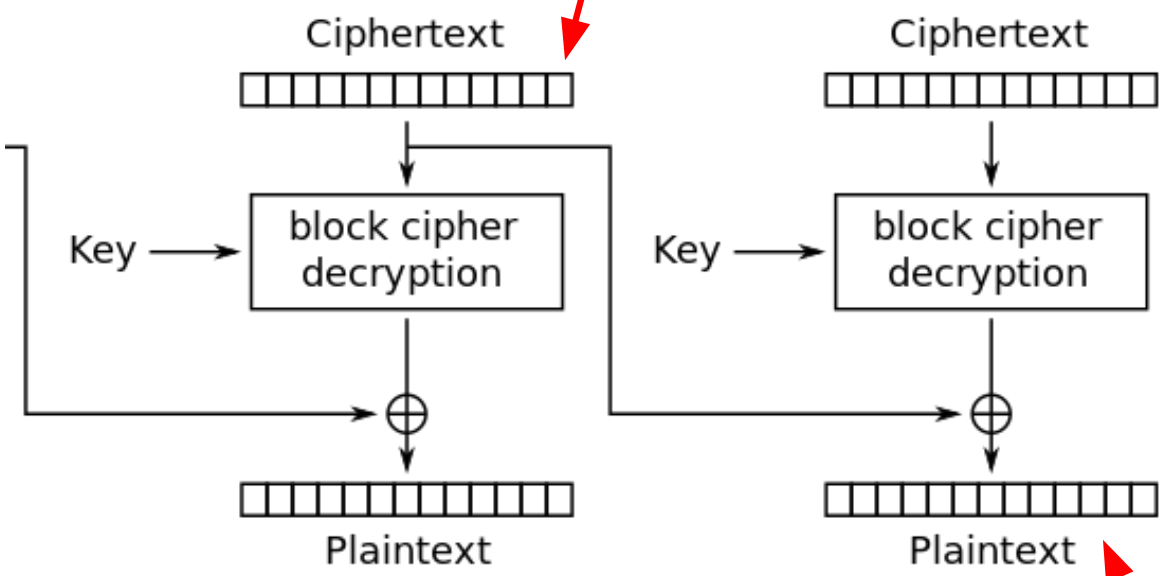


05 05 05 05 05 $\quad c \oplus 01 \oplus 01$

05 05 05 05 01 $\quad c \oplus 01 \oplus i$

What if it is correctly padded?

Brute force: we get two “yes” answers



~~05 05 05 05 05~~ ~~c ⊕ 01 ⊕ 01~~

05 05 05 05 01 c ⊕ 01 ⊕ i

Key Management

RSA SecurID Breach (March 2011)

- Seed values for devices **stored insecurely**, compromised after phishing breach.
- **40M devices replaced**, big companies breached, massive brand damage.

Sophisticated attacks on crypto

May 2012, sophisticated attack on Iranian nuclear programme named **FLAME**

- **A fake certificate** using an MD5 collision was used to install the malware, bypassing software update check
- The MD5 collision method used was **different from the one publicly known**

Conclusion

Cryptography is complex

Cyber criminals use sophisticated attacks against crypto

It is important to understand what security level is provided by the various mechanisms

Have a look at www.cryptosense.com !