

# Introduction to Cryptography

System Security (CM0625, CM0631) 2026-27  
Università Ca' Foscari Venezia

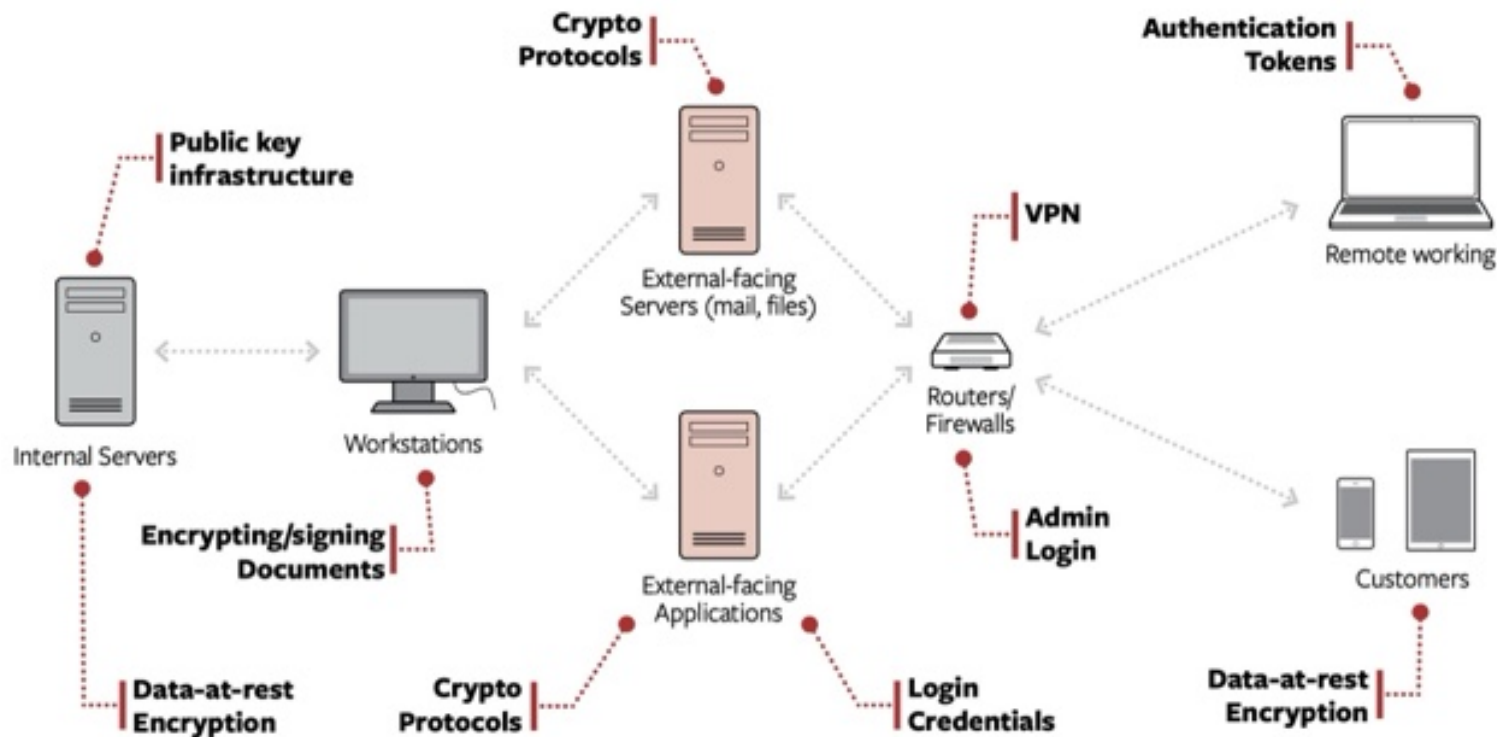
Riccardo Focardi

<https://www.unive.it/people/focardi>  
[seccgroup.dais.unive.it](https://seccgroup.dais.unive.it)

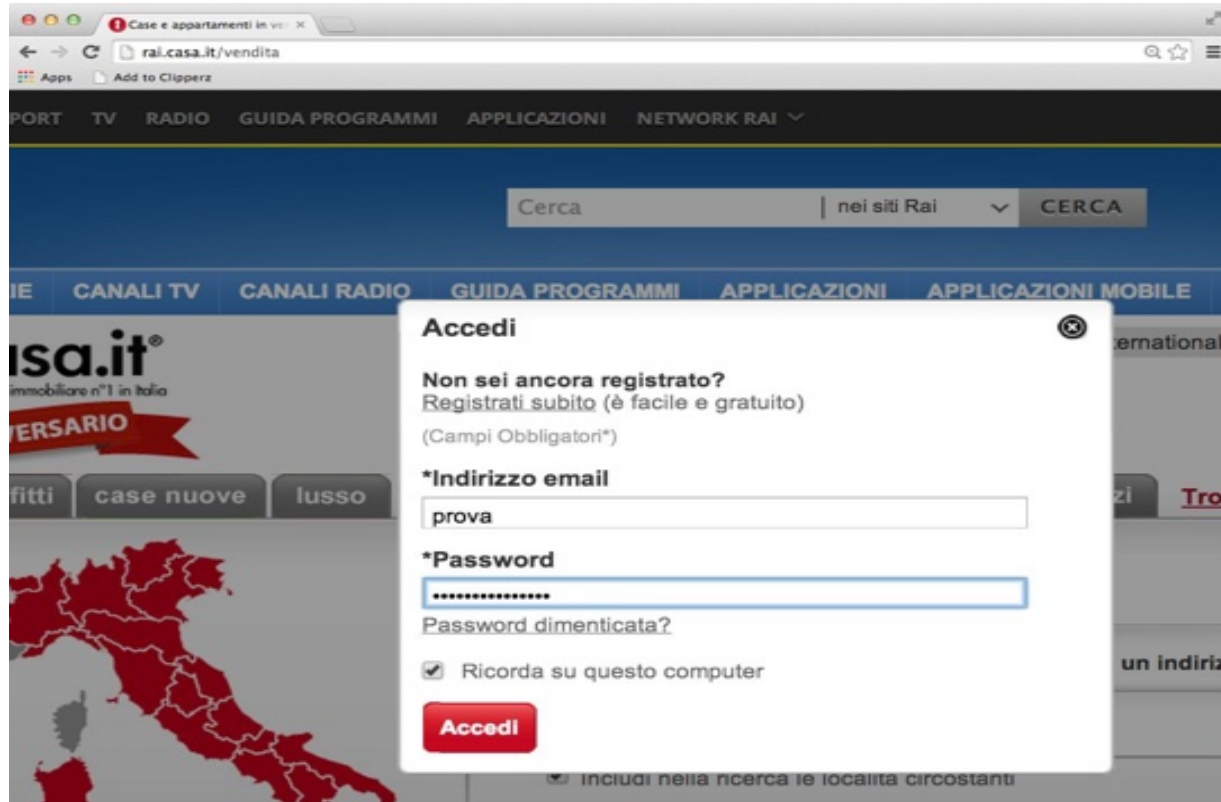


Università  
Ca' Foscari  
Venezia

# Cryptography is everywhere



# Example: cryptography over the web



The image shows a web browser window displaying the Rai Casa website. The address bar shows the URL `rai.casa.it/vendita`. The website has a dark blue header with navigation links: PORT, TV, RADIO, GUIDA PROGRAMMI, APPLICAZIONI, and NETWORK RAI. Below the header is a search bar with the text "Cerca" and a dropdown menu showing "nei siti Rai". The main content area features the Rai Casa logo, a banner for "BORSARIO", and a map of Italy. A login modal is open in the center of the screen.

**Accedi**

Non sei ancora registrato?  
[Registrati subito](#) (è facile e gratuito)  
(Campi Obbligatori\*)

\*Indirizzo email

\*Password

[Password dimenticata?](#)

☒ Ricorda su questo computer

**Accedi**

# http: no protection!

en1 [Wireshark 1.6.3 (SVN Rev 39702 from /trunk-1.6)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Follow TCP Stream

Filter: tcp.stream

| No. | Time     |
|-----|----------|
| 11  | 7.982900 |
| 12  | 8.103004 |
| 13  | 8.103095 |
| 14  | 8.105126 |
| 15  | 8.206281 |
| 16  | 8.810061 |
| 17  | 8.810129 |

Stream Content

```
POST /login_verify.ds HTTP/1.1
Host: rai.casa.it
Connection: keep-alive
Content-Length: 55
Origin: http://rai.casa.it
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/34.0.1847.137 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer: http://rai.casa.it/vendita
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,it;q=0.6
Cookie: NSC_etbqqgbsn=e2451c313660;
lmdstok=awQjZmY4MDgxODE0MmZlOTdmMzAxNDYyMGQyMGN1MDRlZjg6MzU0ODE5NDk2MDMxNTpiYzcyYzU5YzYk
SNjgxNTRlMmYyNDI5NkU0MTNmNzU0YQ; s_nr=1400710641395;
JSESSIONID=0A004D54AE602C9391470186C21E7579;
__utma=135384818.195807481.1400709972.1400709972.1400750729.2;
__utmb=135384818.2.10.1400750729; __utmc=135384818;
__utmz=135384818.1400709972.1.1.utmcsr=sipra-rai|utmccn=institutional|utmcmd=cobrand-
tab|utmctt=logo-link; s_cc=true; _stc=raicobweb; s_sq=%5B%5B%5D%5D
username=prova&password=passwordsegreta&rememberMe=true...5...@c.-L..K.
M..lk...|W.nQ7n...
.p
```

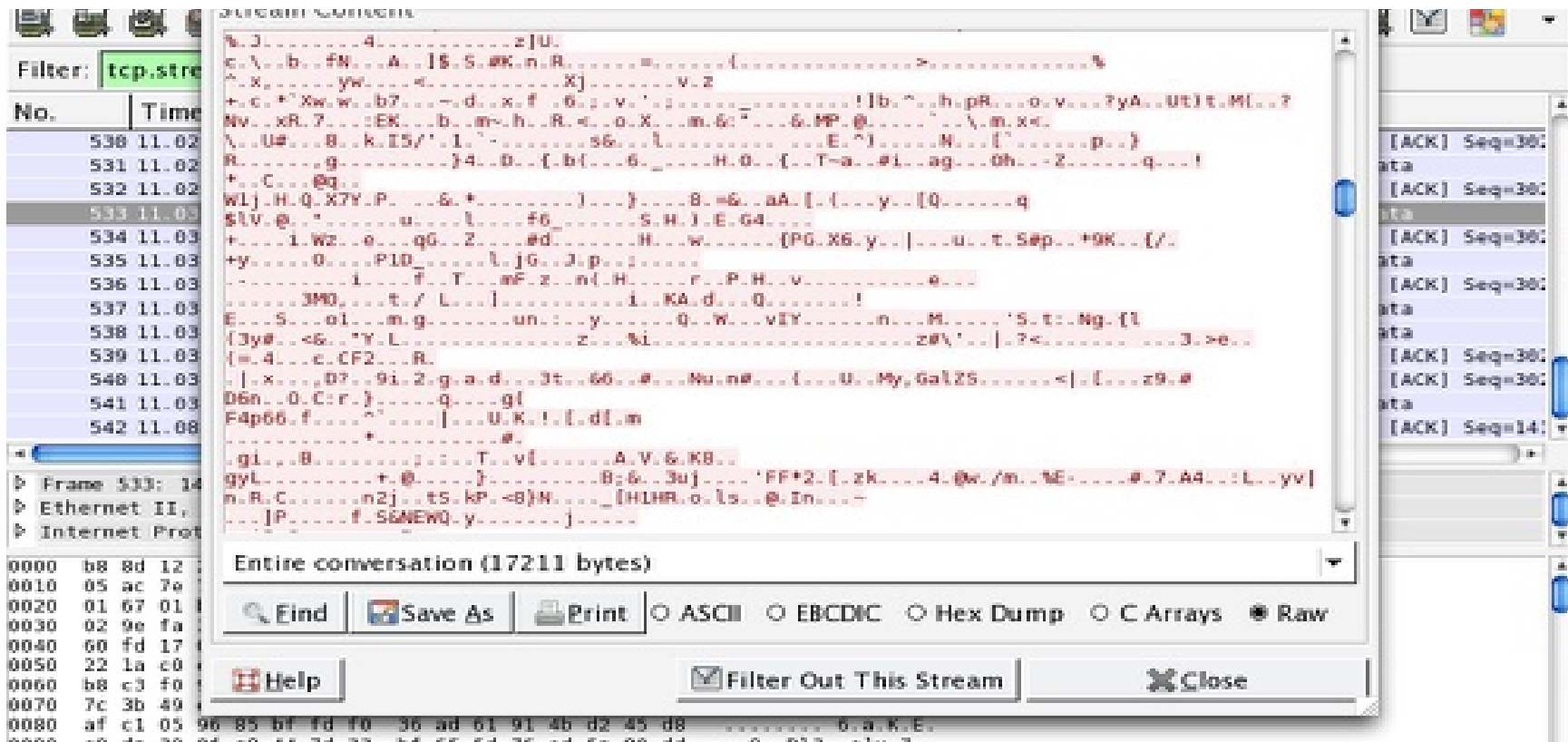
Entire conversation (1297 bytes)

Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw

Help Filter Out This Stream Close

File: \*/var/folders/b\_/6833\_... Packets: 28 Displayed: 7 Marked: 0 Dropped: 0 Profile: Default

https: communication is encrypted



# Cryptography in embedded devices



# Cryptography in banks

Payments, ATMs, money transfers, ...

## Hardware Security Module (HSM)

- Costs about 5k-20k € for a market of 200M € a year
- Now available in the Cloud (CloudHSM)



# What is a Cipher?

A cipher is defined through two functions

- **Encryption:** given a plaintext and a key  $K_1$  returns a ciphertext

$$E_{K_1}(X) = Y$$

- **Decryption:** given a ciphertext and a key  $K_2$  returns a plaintext

$$D_{K_2}(Y) = X$$

# Symmetric and asymmetric ciphers

Keys  $K_1$  and  $K_2$  are related: decrypting the encryption of  $X$  we obtain  $X$ :

$$D_{K_2}(E_{K_1}(X)) = X$$

- When  $K_1 = K_2$  we have a **symmetric** key cipher (example: AES)
- When  $K_1 \neq K_2$  we have an **asymmetric** key cipher (example: RSA)

**Security** (*known plaintext* scenario): it should be **infeasible** to compute  $X$  or  $K_2$  from  $Y$  even knowing other pairs  $(X_1, Y_1), \dots, (X_n, Y_n)$

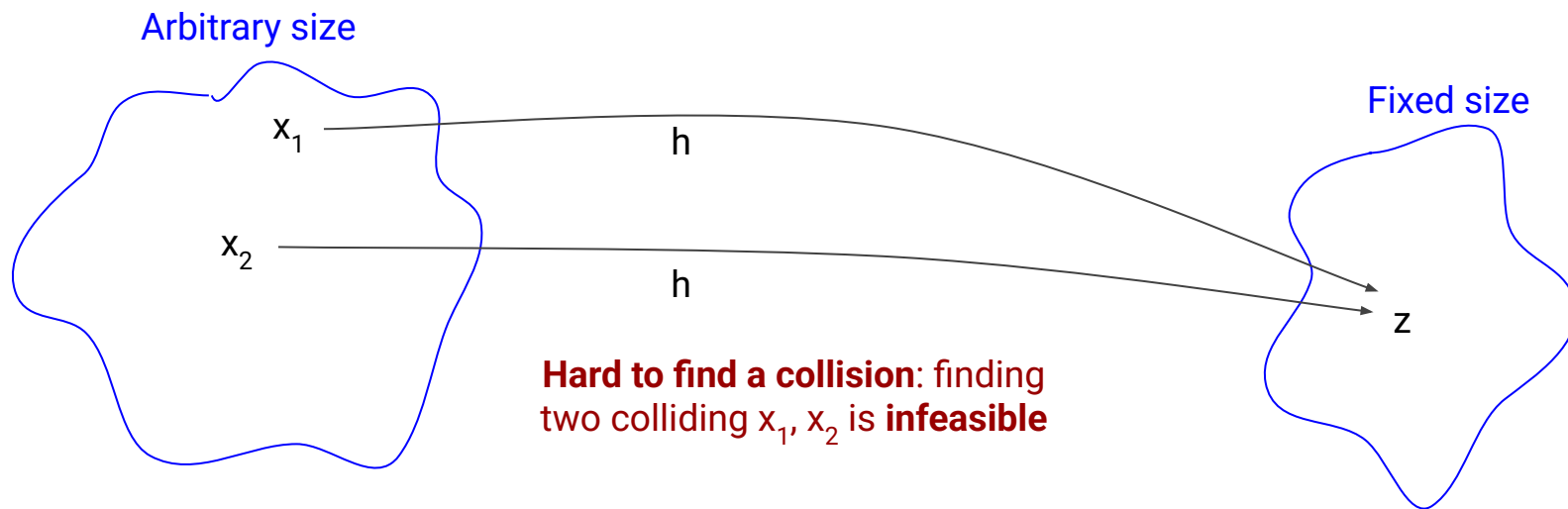
# Cryptographic hash functions

Recall what a hash function is:

**Definition** (*hash function*). A hash function  $h$  computes efficiently a **fixed length** value  $h(x)=z$  called **digest**, from an  $x$  of **arbitrary size**.

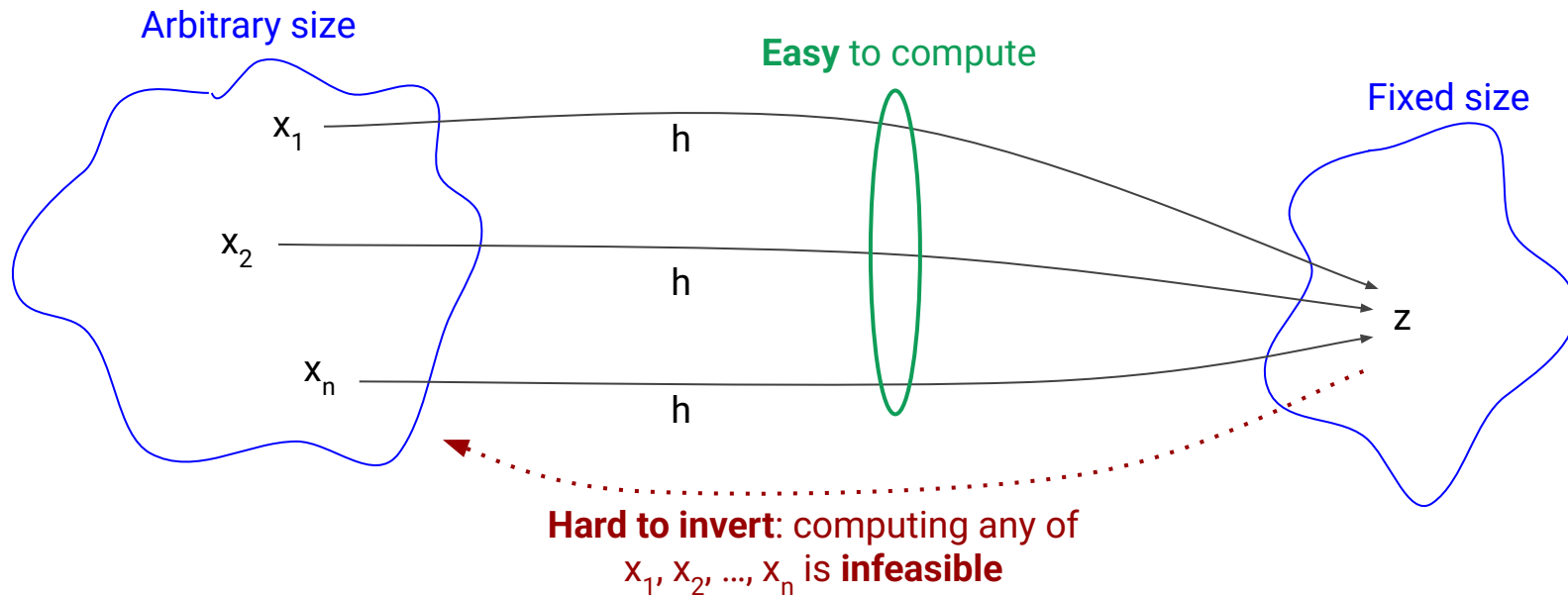
**NOTE: Collisions** are possible:  $h(x_1) = h(x_2)$

# Collision resistant hash function



**Definition** (*collision resistant hash function*). A hash function  $h$  is **collision resistant** if it is infeasible to compute different  $x_1, x_2$  such that  $h(x_1) = h(x_2)$

# One-way hash function



**Definition** (*one-way hash function*). A hash function  $h$  is **one-way** if, given a digest  $z$ , it is **infeasible** to compute a preimage  $x'$  such that  $h(x')=z$

# Nice and elegant ... but things can go wrong

Many attacks on real world cryptography in the last years:

- S Calzavara, R Focardi, M Nemec, A Rabitti, M Squarcina. **Postcards from the post-HTTP world: amplification of HTTPS vulnerabilities in the web ecosystem**. IEEE S&P 2019
- R Focardi, F Palmarini, M Squarcina, G Steel, M Tempesta. **Mind Your Keys? A Security Evaluation of Java Keystores**. NDSS 2018
- 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

Many TLS vulnerabilities are still around!



DROWN



HEARTBLEED



ROBOT

NOT ON TOP SITES, RIGHT?!

# Smartcards and crypto tokens

| Brand   | Device Model     | Supported Functionality |    |      |      |   |    | Attacks found |    |    |    | Tk |
|---------|------------------|-------------------------|----|------|------|---|----|---------------|----|----|----|----|
|         |                  | s                       | as | cobj | chan | w | ws | wd            | rs | ru | su |    |
| 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 |



The code for devices like RSA Security's SecurID 800 constantly changes, but computer scientists have found weaknesses.

## Scientists Make Short Work Of Breaking Security Keys

By SOHINI SENGU/PTA

For years private companies and government agencies have given their employees a card or token that produces a constantly changing set of numbers. Those devices became the preferred method of securing confidential communications online. No one could have access to the data without a secret key generated by the device.

Computer scientists say they have now figured out how to extract that key from a widely used RSA electronic token in as little as 13 minutes.

The scientists, who call them-

encryption tools were anticipated and susceptible to attack.

"It would be nice if manufacturers paid more heed to what they might see only as theoretical attacks and were more cautious," said Chris Peikert, a theoretical cryptographer who teaches computer science at the Georgia Institute of Technology. "In an ideal world this problematic standard would have been transmitted away from years ago."

One of the reasons this standard has persisted, Mr. Peikert said, is that until now, researchers and manufacturers reckoned that it would take a long time to

# 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

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

**(In)security of mechanisms:** Crypto mechanisms **are not** equally secure

**Configuration and management:** The **configuration and management** of cryptographic systems is complex and error prone

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

# Vulnerabilities in applications

# 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



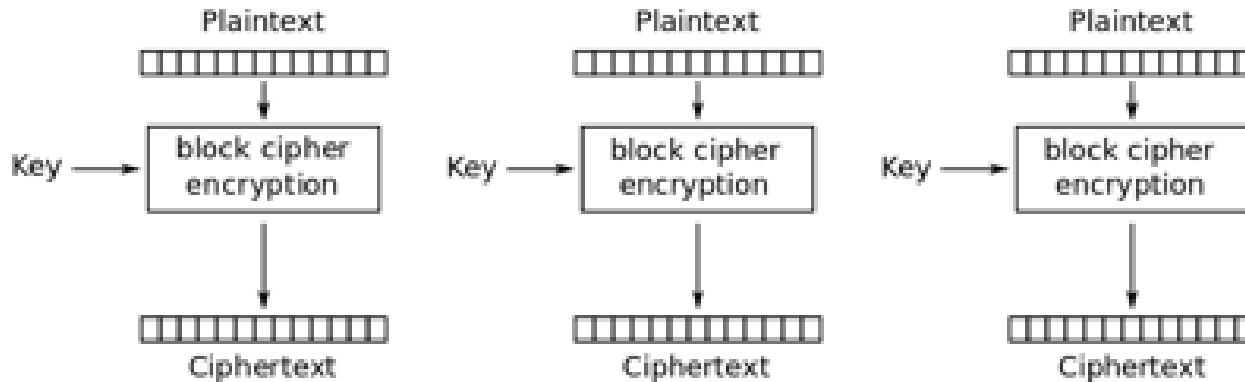
<http://heartbleed.com/>

(In)security of mechanisms

# Modes of operation

Needed, for example, when data is bigger than the block size

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



# ECB weaknesses

In ECB blocks are encrypted **independently** under the same key

- **Problem 1:** Equal blocks are encrypted in the same way
- **Problem 2:** Swapping encrypted blocks also swaps plaintext blocks

⇒ Poor confidentiality and integrity

# Example 1: poor confidentiality



Università  
Ca' Foscari  
Venezia

plaintext



ciphertext (ECB)

# Example 1: simple substitutions of blocks



Università  
Ca'Foscari  
Venezia

plaintext



Università  
Ca'Foscari  
Venezia

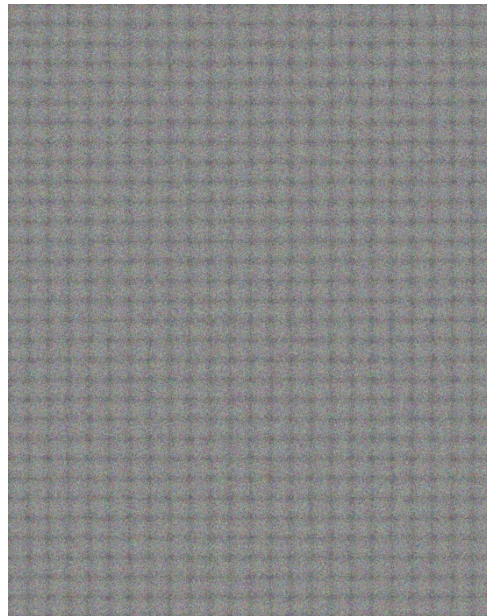
ciphertext, after simple substitutions

# Example 1: using CBC!



Università  
Ca'Foscari  
Venezia

plaintext



ciphertext (CBC mode)

# Example 2: breaking integrity

Consider sentences:

- Security course is great!!
- Today's weather is really bad!

When splitted in 16 bytes (AES) blocks they become:

- Security course is great!!
- Today's weather is really bad!

**Task:** given the two ciphertexts in AES-ECB, forge a new valid ciphertext putting the security course in a bad light 😏

# Chosen plaintext attack in ECB

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

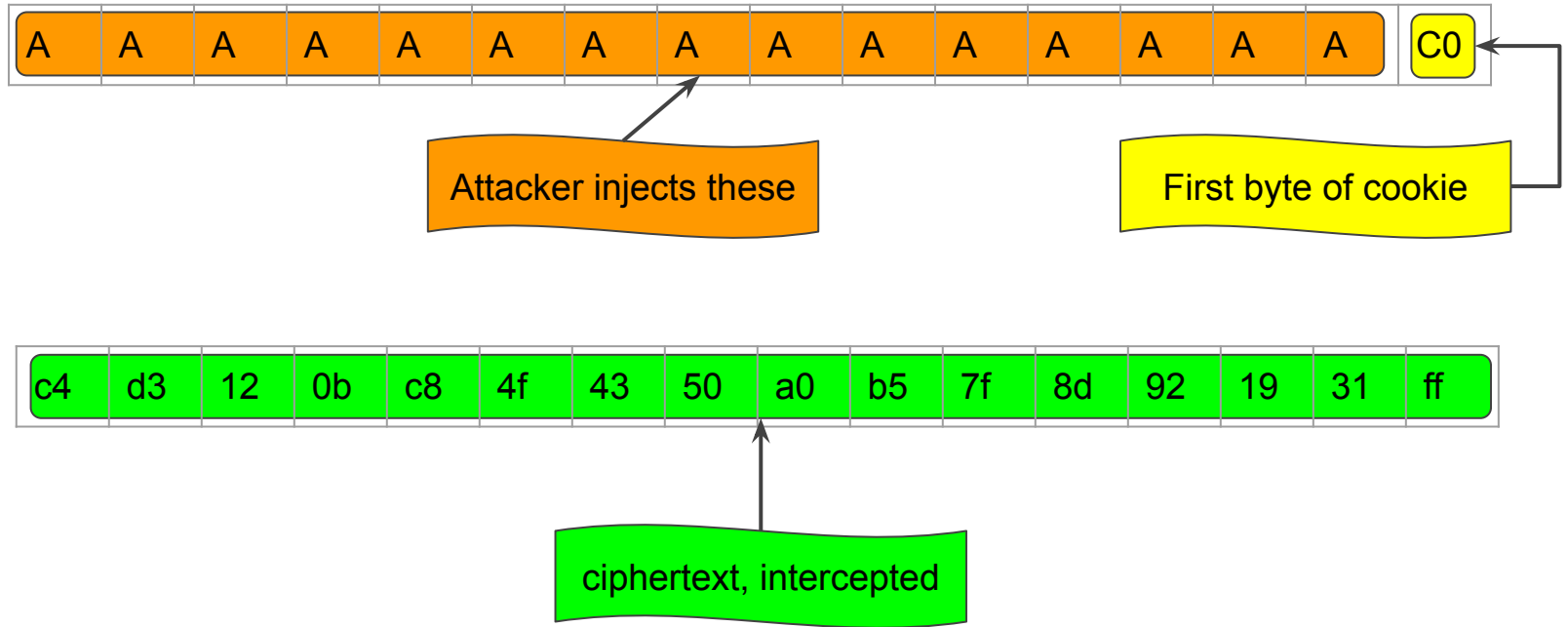
Intuitively:

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

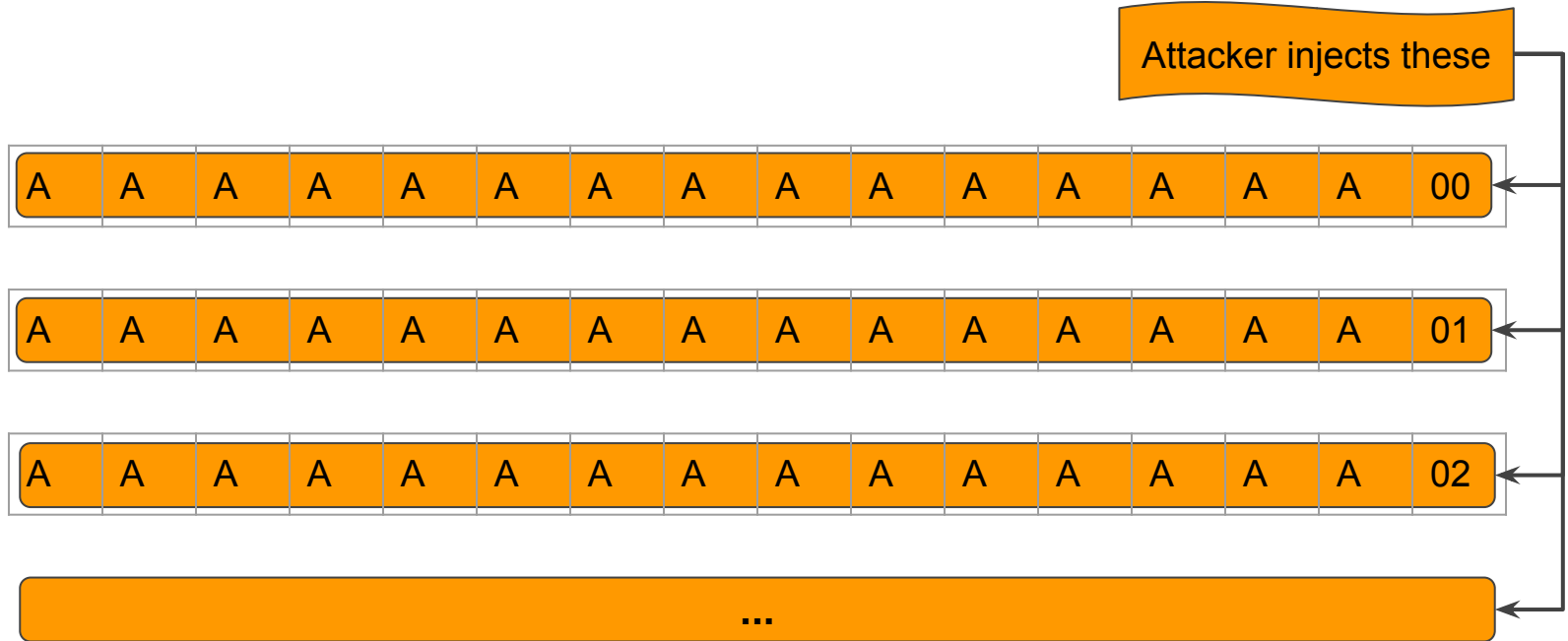
**Realistic scenario** (similar to the one in the BEAST attack):

- Secure session cookies are sent over HTTPS
- Javascript cannot access them
- Malicious javascript can forge cross-domain requests to honest domains (cookie is sent!)
- Attacker can add plaintext before the cookie value!

# The attack in detail (1)

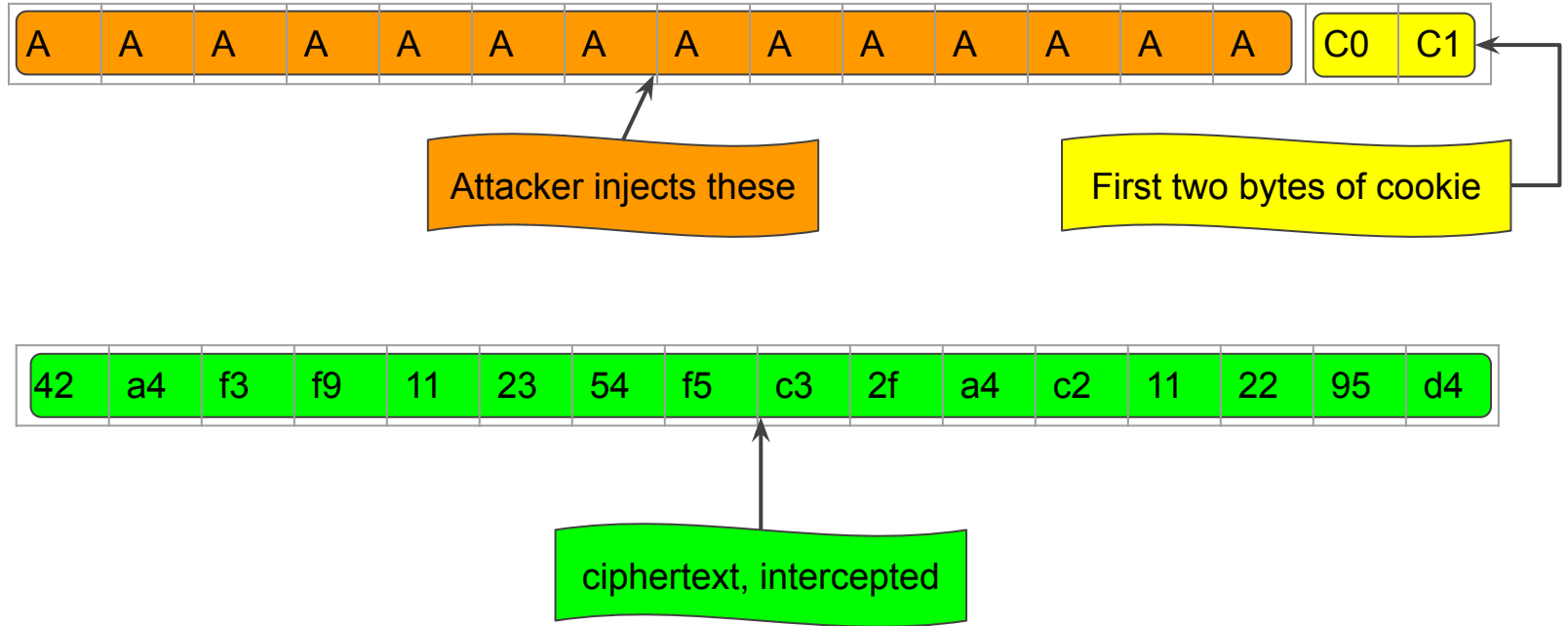


# The attack in detail (2)

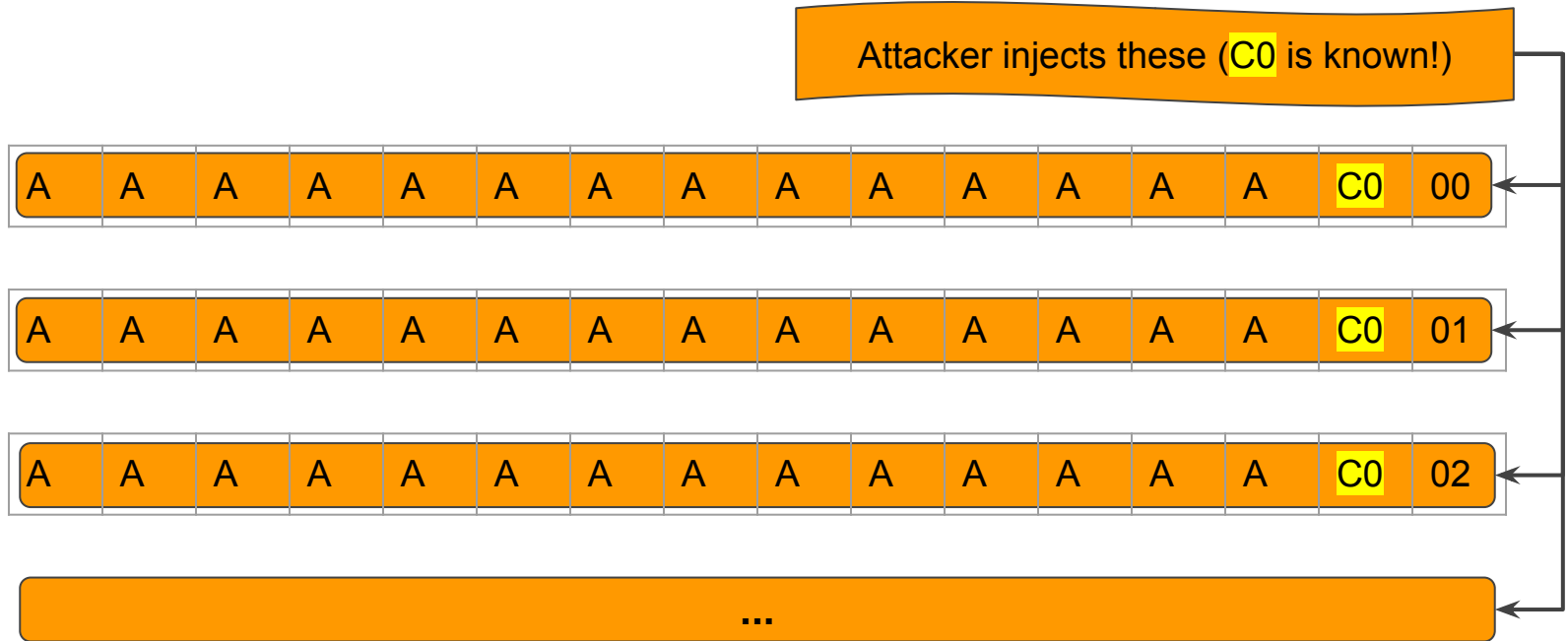


... until the ciphertext matches the previous one  $\Rightarrow$  **C0** is leaked!

# The attack in detail (3)



# The attack in detail (4)



... until the ciphertext matches the previous one  $\Rightarrow$  **C1** is leaked!

# Configuration and Management

# Key Management

## RSA SecurID Breach (March 2011)

- Seed values (i.e., secret keys) for devices **stored insecurely**, compromised after phishing breach
- **40M devices replaced**, big companies breached, massive brand damage



# Cryptanalysis

# Sophisticated attacks on crypto

May 2012, sophisticated attack on Iranian nuclear programme named **FLAME**  
(and related to Stuxnet)

- **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**

⇒ State-level cyber-attack!!

**NOTE:** MD5 is now deprecated and should not be used for cryptographic applications

# Cryptographic vulnerabilities (summary)

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

**Example:** Heartbleed

**(In)security of mechanisms:** Crypto mechanisms **are not** equally secure

**Example:** Weak modes of operation (ECB), padding oracles (PKCS7)

**Management:** The **configuration and management** of cryptographic systems is complex and error prone

**Examples:** Fixed IV, bad key management

**Cryptanalysis:** Improvement in **technology/cryptanalysis** requires better crypto

**Example:** broken cryptographic hashes