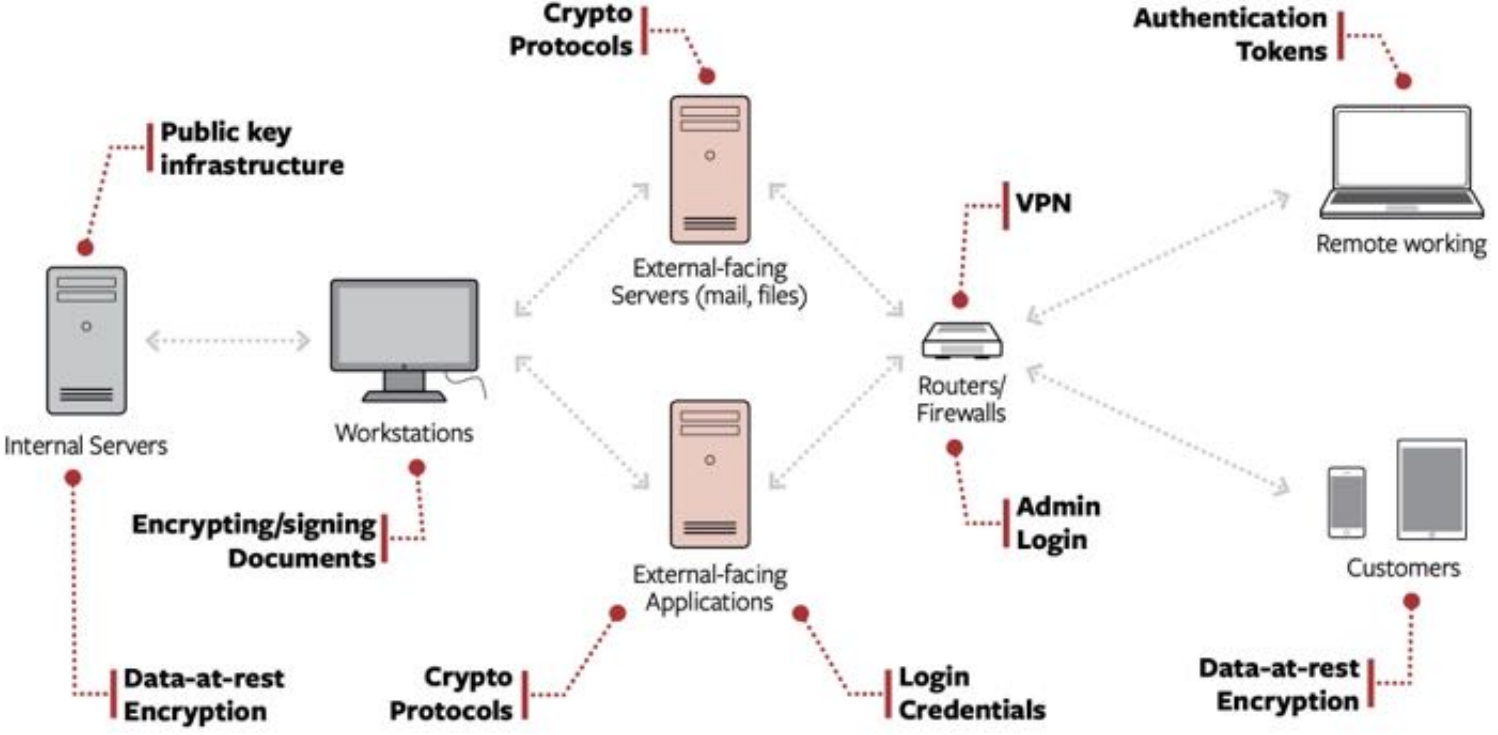
Introduction to Cryptography

System Security (CM0625, CM0631) 2024-25
Università Ca' Foscari Venezia

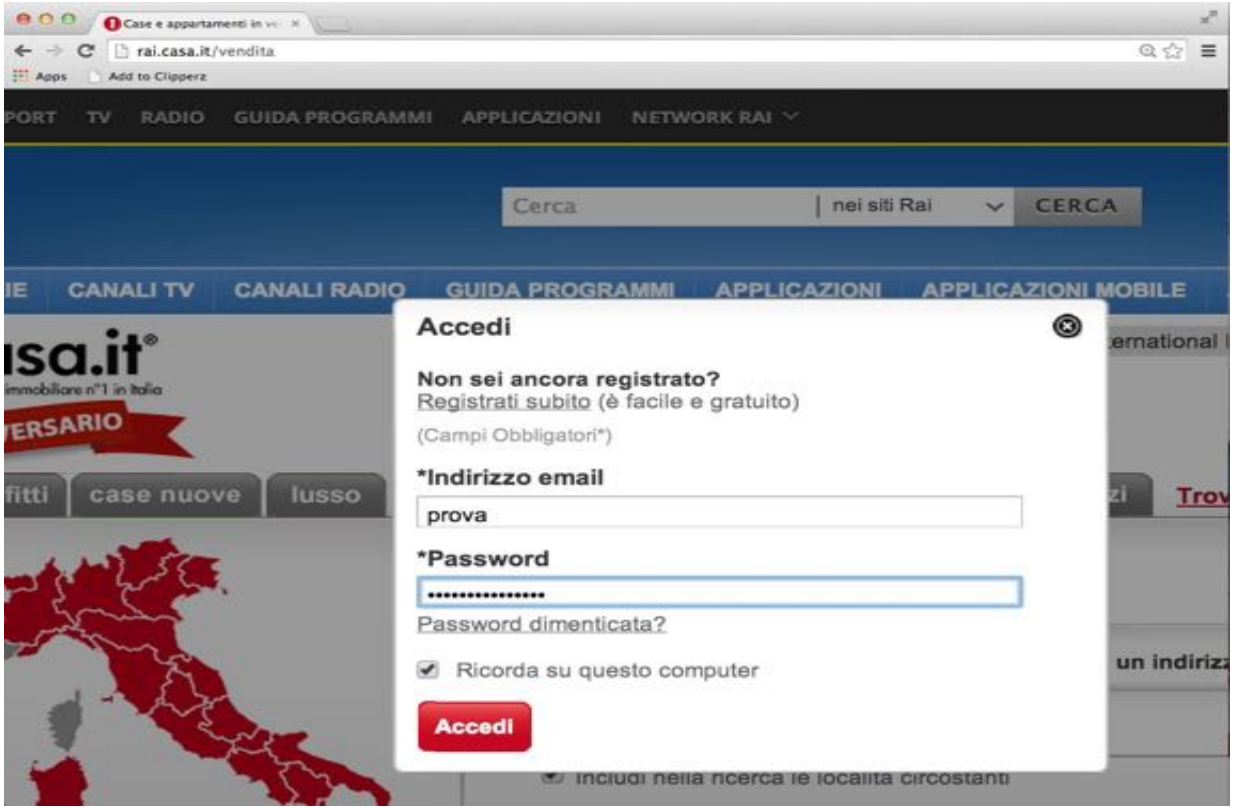
Riccardo Focardi
www.unive.it/data/persone/5590470
secgroup.dais.unive.it

Università Ca'Foscari Venezia

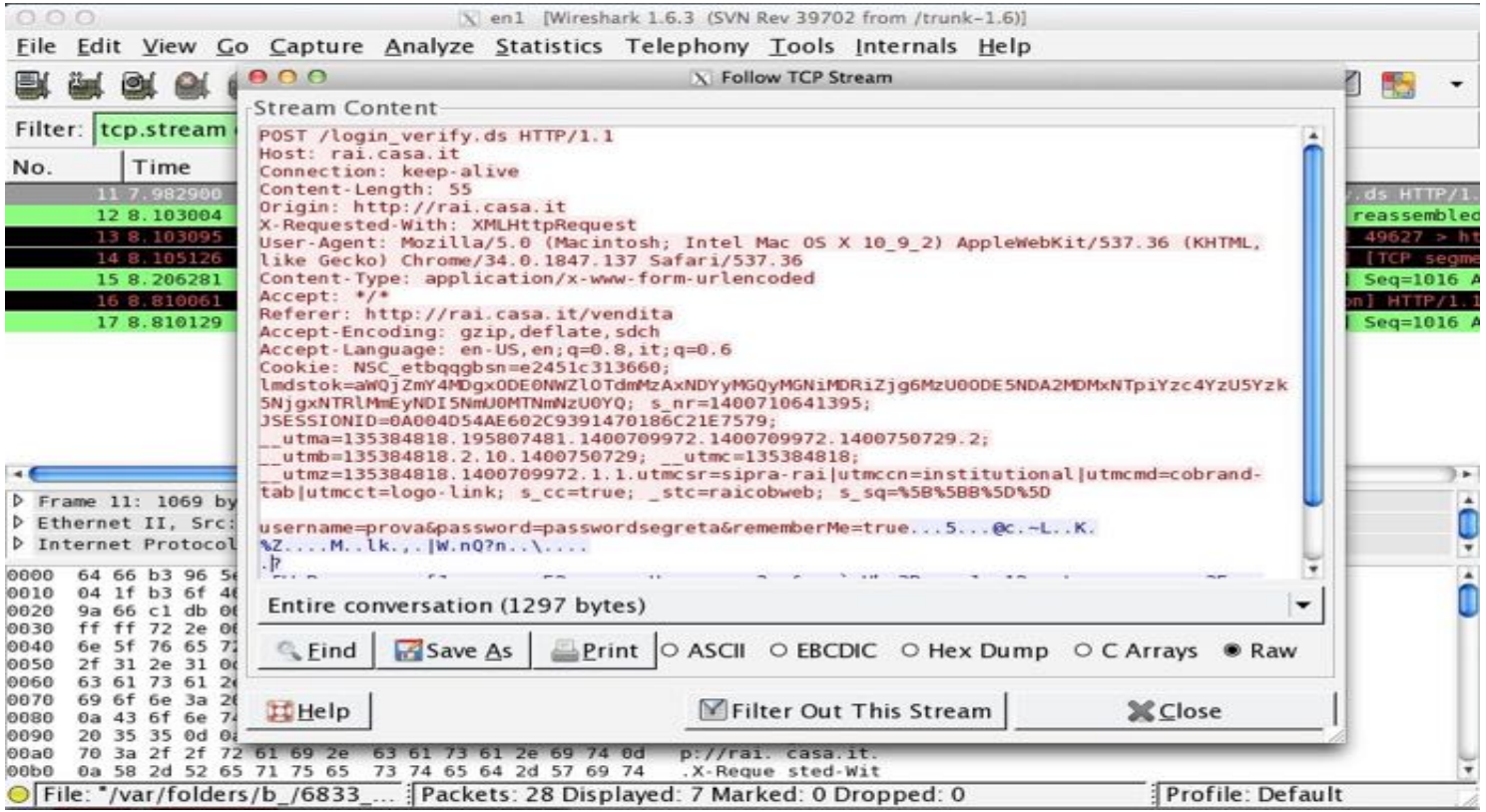
Cryptography is *everywhere*



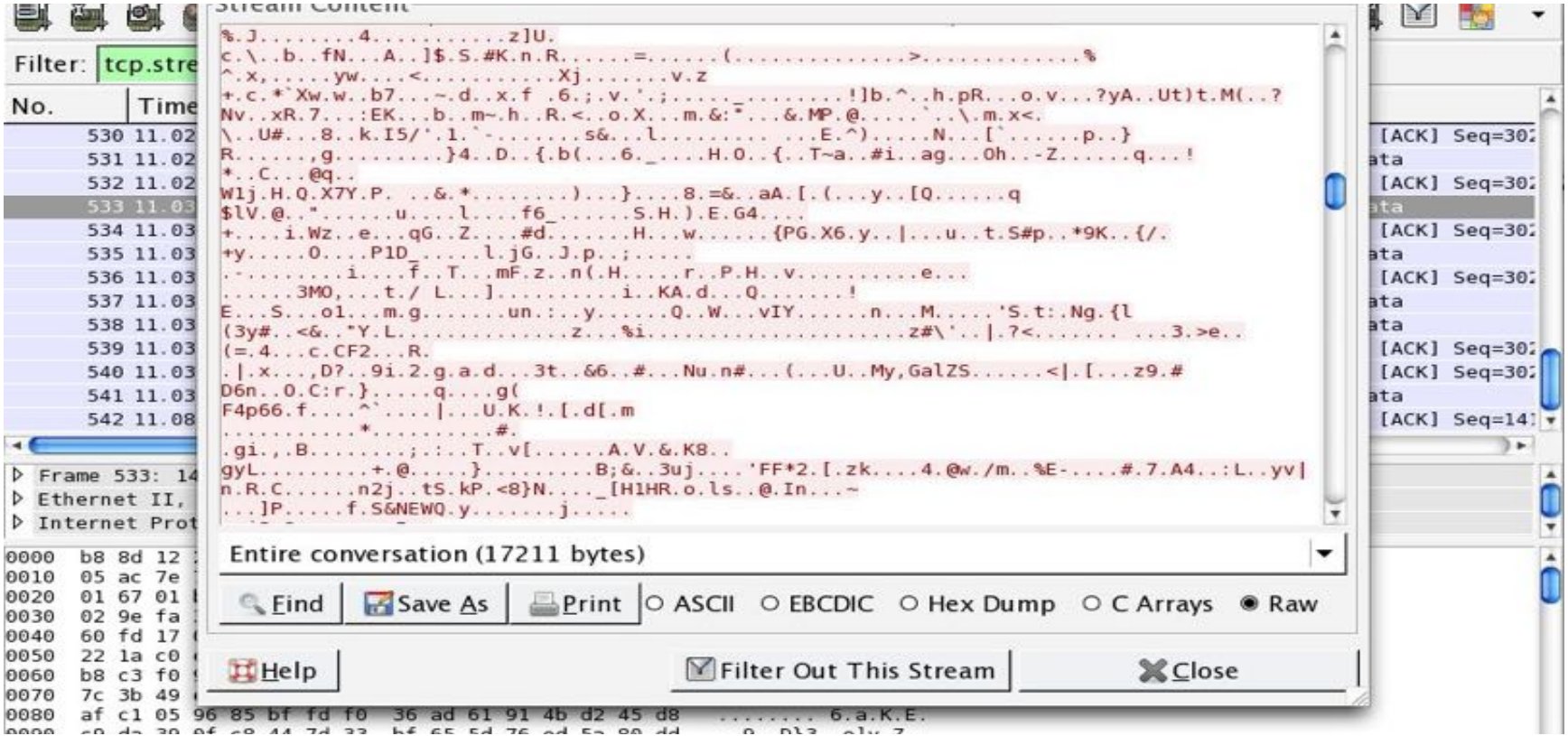
Example: cryptography over the web



http: no protection!



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

What is a Cipher?

A cipher is defined through two functions

- **Encryption**: given a plaintext and a key K1 returns a ciphertext

$$\mathbf{E}_{K1}(X) = Y$$

- **Decryption**: given a ciphertext and a key K2 returns a plaintext

$$\mathbf{D}_{K2}(Y) = X$$

Symmetric and asymmetric ciphers

Keys K1 and K2 are related: decrypting the encryption of X we obtain X:

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

- When K1=K2 we have a **symmetric** key cipher (example: AES)
- When K1≠K2 we have an **asymmetric** key cipher (example: RSA)

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

Cryptographic hash functions

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

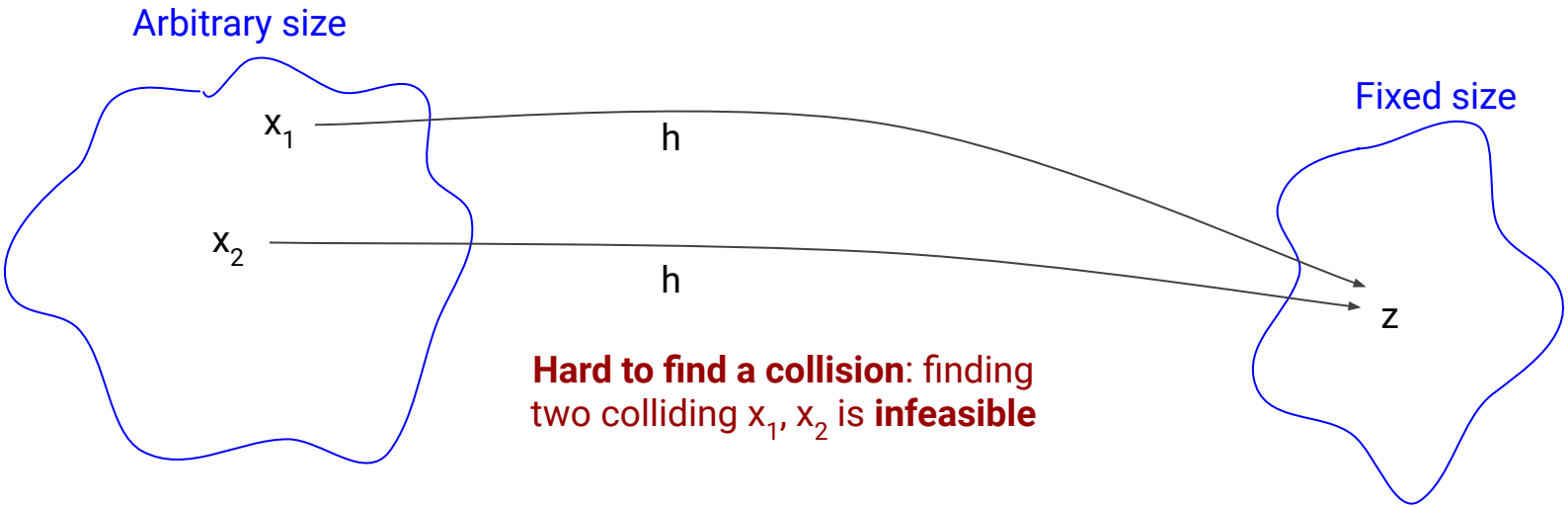
NOTE: **Collisions** are possible: h(x1) = h(x2)

Definition (***collision resistant hash function***). A hash function h is **collision resistant** if it is infeasible to compute different x1, x2 such that h(x1) = h(x2)

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

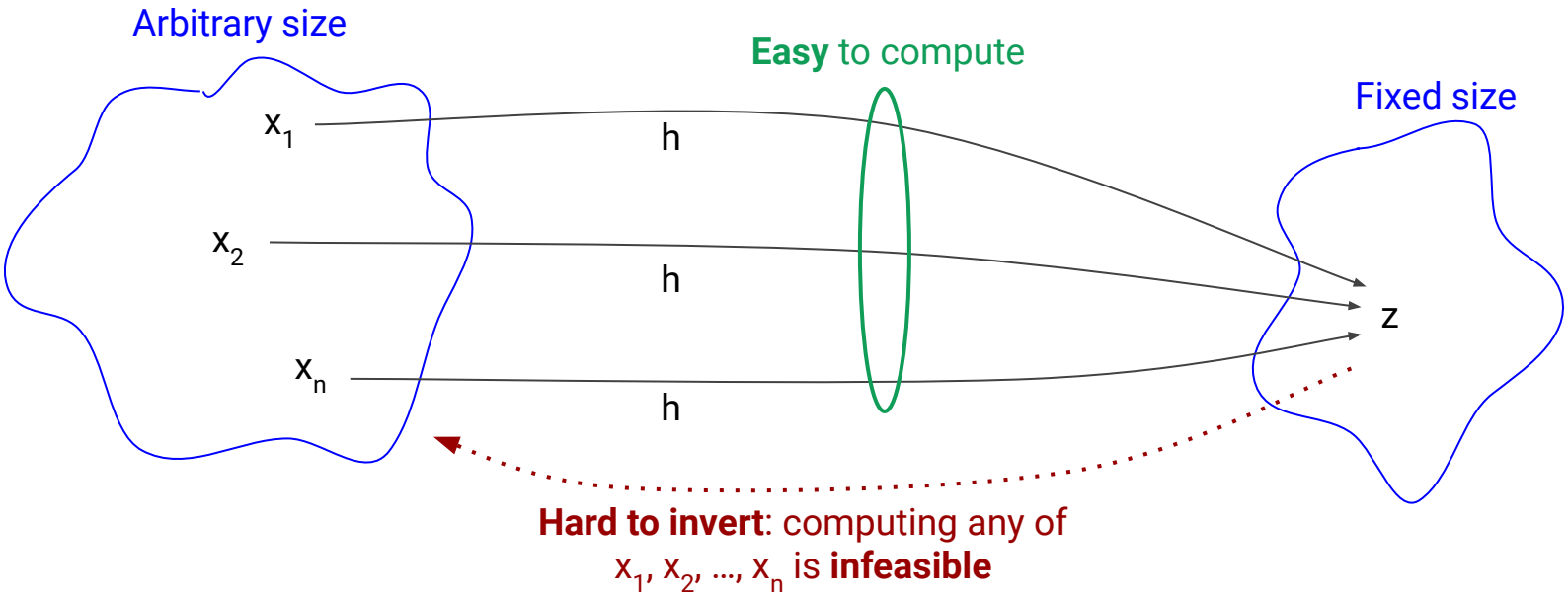
⇒ **Finding** a pre-image is computationally infeasible

Collision resistant hash function



Hard to find a collision: finding two colliding x_1, x_2 is **infeasible**

One-way hash function



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

Amplification of HTTPS vulnerabilities in the web ecosystem. IEEE S&P 2019

Smartcards and crypto tokens

Device		Supported Functionality						Attacks found				
Brand	Model	s	as	cobj	chan	w	ws	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

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 SOMINI SENGUPTA

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 themencryption tools were antiquated 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 transitioned 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.

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 when:

- Data is bigger than the block size
- It is necessary to encrypt a *stream* with a *block* cipher

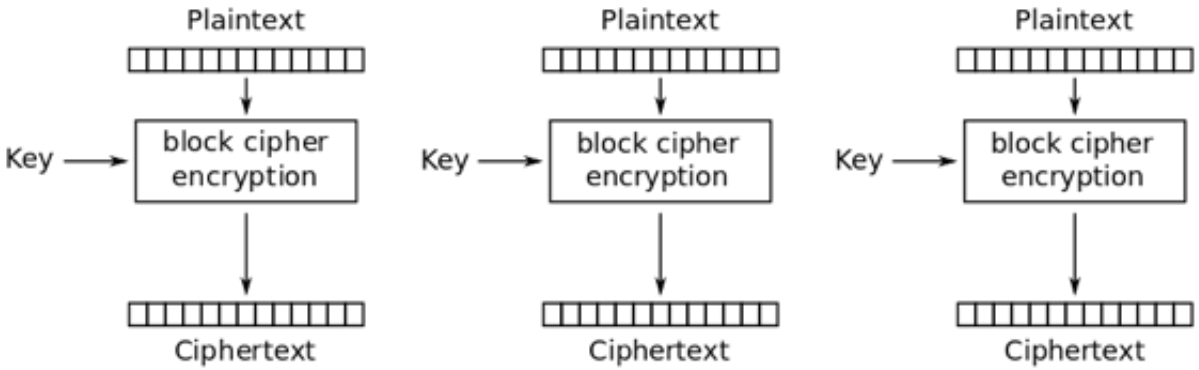
Example: AES-ECB is a mode of operation that splits long messages into blocks of 16 bytes (the size of AES block)

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

⇒ not so different from simple substitution ciphers!

ECB mode of operation



Electronic Codebook (ECB) mode encryption

⇒ Poor confidentiality and integrity

Example 1: poor confidentiality

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

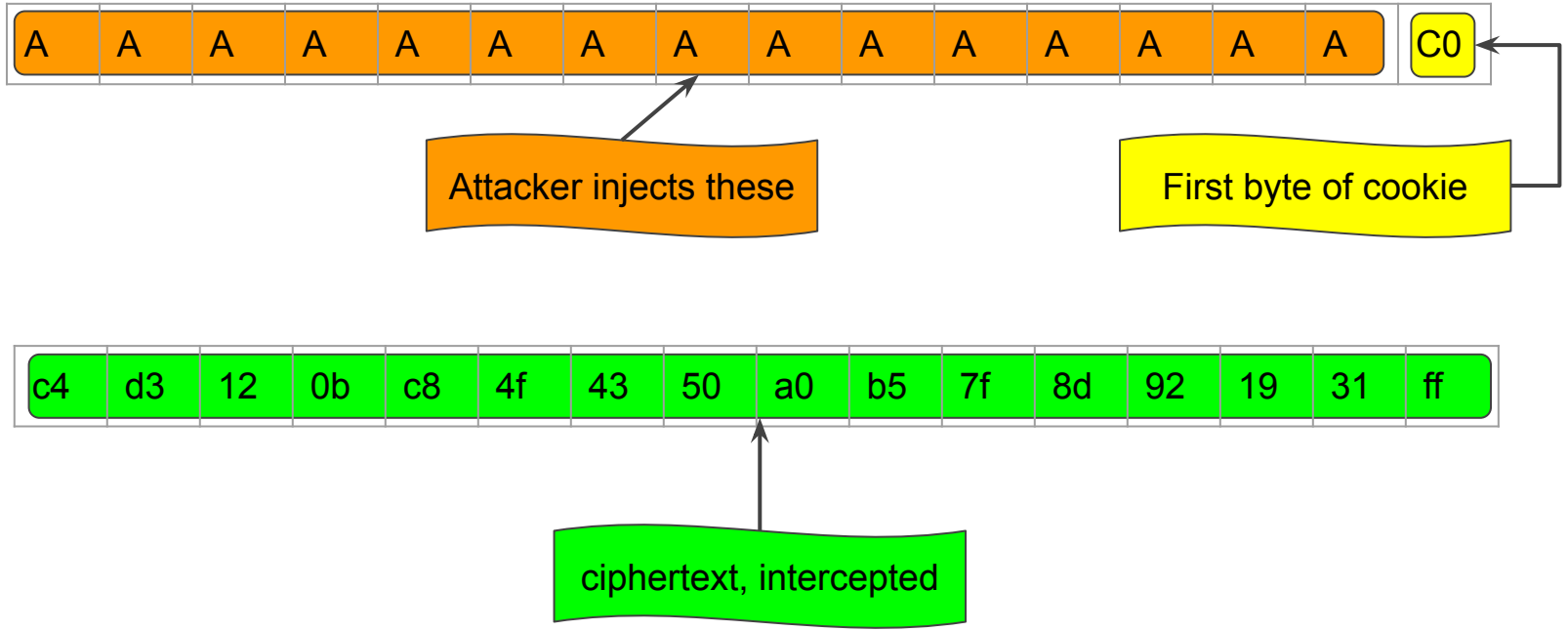
Intuitively:

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

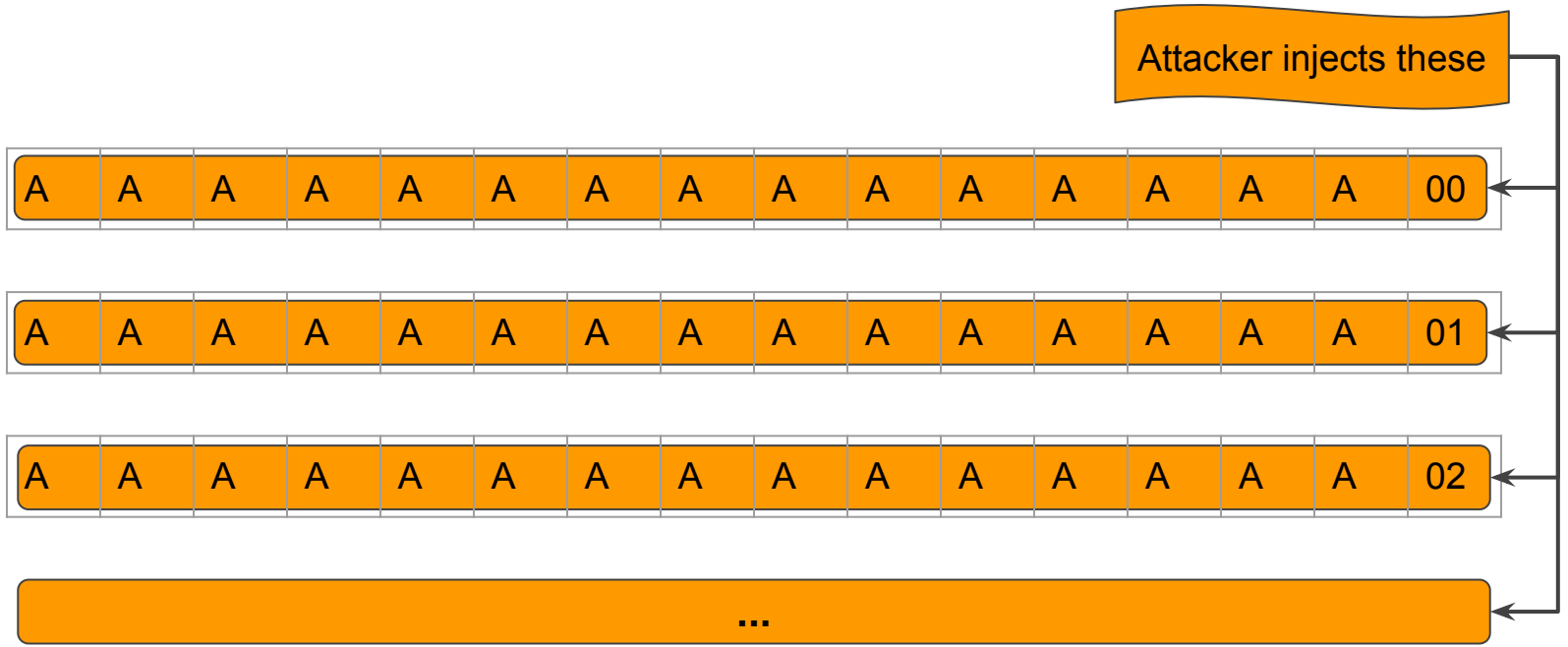
Realistic scenario (similar to the one in the BEAST attack):

- 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)

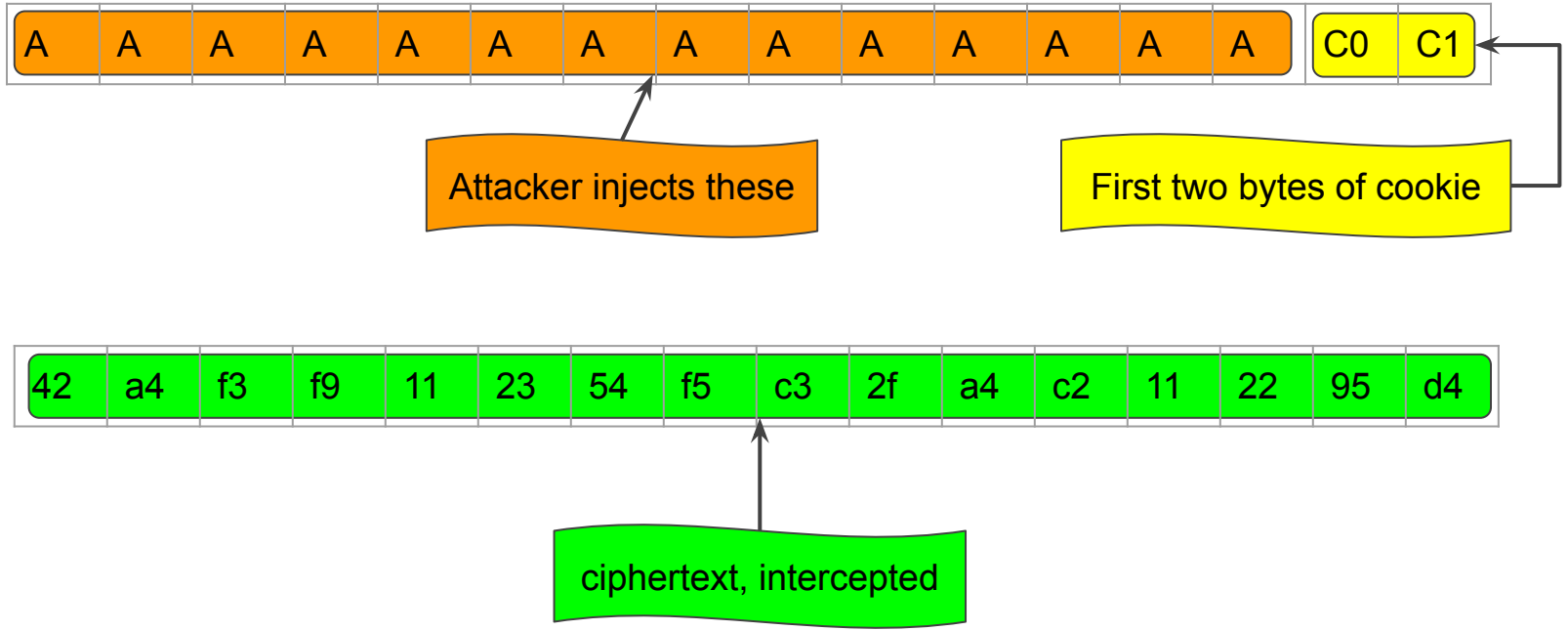


The attack in detail (2)

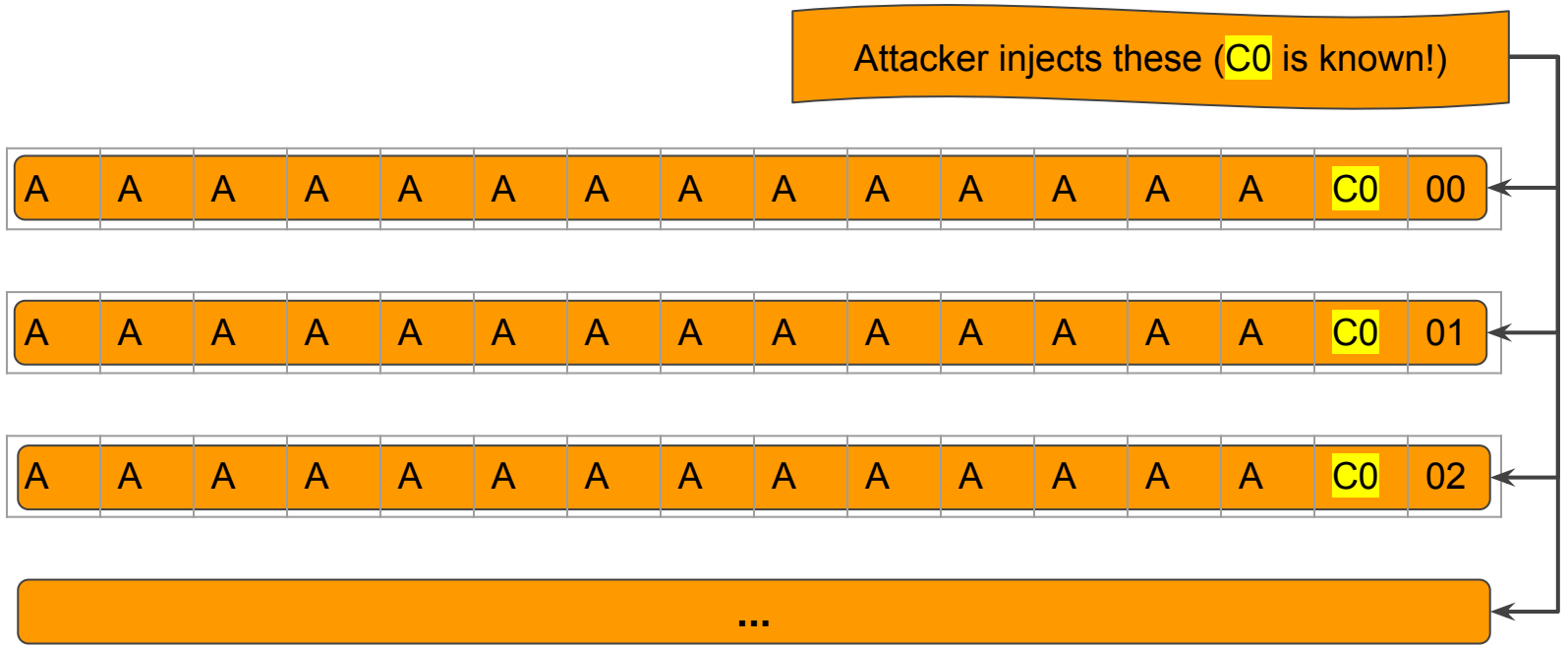


… until the ciphertext matches the previous one ⇒ C0 is leaked!

The attack in detail (3)



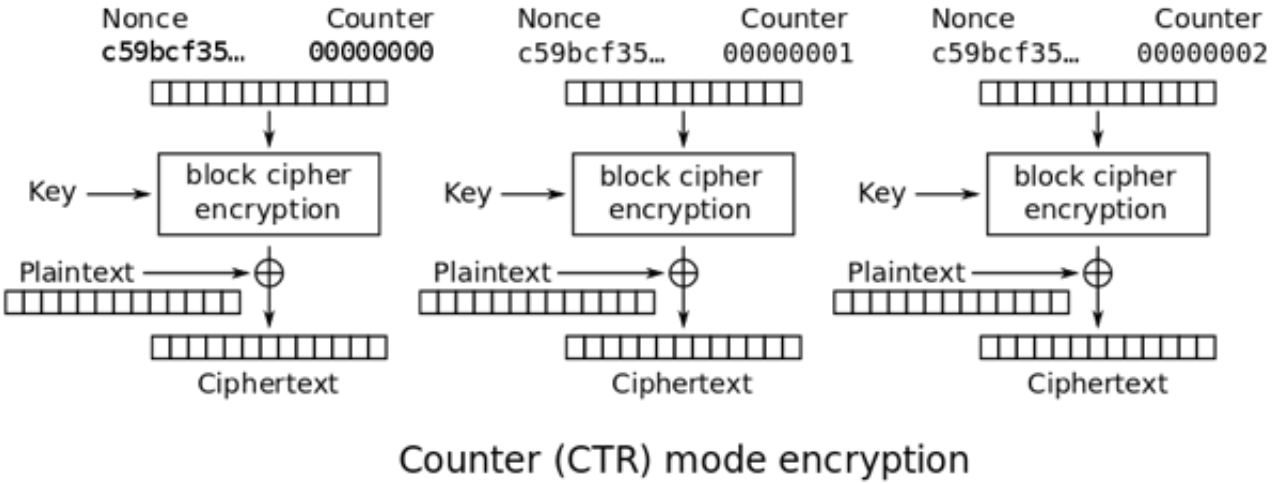
The attack in detail (4)



… until the ciphertext matches the previous one ⇒ C1 is leaked!

Configuration and Management

CTR: stream cipher



Counter (CTR) mode encryption

⇒ Random nonce is fundamental for security!

Fixed IV is a typical configuration mistake!

ciphertext 1:

```
8f079a817d1dfa5bb2b1e069b0f4027abc65db6d130e6f3c154611d165d66b0a2342473479
0df0769cc3c4f4f289e784ac0cc5cab7e47c5c1a
```

ciphertext 2:

```
9f0a92807d33fb1ab7a9ad36e5cd4064a320da7a56122e21004c42c46d93214b28595b7776
12e46c9dc3c4eefedde88ee31c97c1b1e834135c
```

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

$$P1 \oplus P2 = C1 \oplus C2$$
$$P2 = P1 \oplus C1 \oplus C2$$

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

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

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: Weaks 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