



# **IT-Security (ITS) B1**

**DIKU, E2023**



# Today's agenda

- Crypto building blocks
- (Later: Real-world crypto protocols)

# Lecture plan

Week	Date	Time	Lecturer	Topic
36	04-sep	10-12	TL	Security concepts and principles
	08-sep	10-12	TL	Cryptographic building blocks
37	11-sep	10-12	TL	Key establishment and certificate management
	15-sep	10-12	CJ	User authentication, IAM
38	18-sep	10-12	CJ	Operating systems security, web, browser and mail security
	22-sep	10-12	CJ	IT security management and risk assessment
39	25-sep	10-12	TL	Software security - exploits and privilege escalation
	29-sep	10-12	TL	Malicious software
40	02-okt	10-12	CJ	Firewalls and tunnels, security architecture
	06-okt	10-12	CJ	Cloud and IoT security
41	09-okt	10-12	TL	Intrusion detection and network attacks
	13-okt	10-12	TL	Forensics
42				Fall Vacation - No lectures
43	23-okt	10-12	CJ	Privacy and GDPR
	27-okt	10-12	CJ	Privacy engineering
44	30-okt	10-12	CJ,TL	Final guest lecture and Exam Q/A



# Main messages

- Cryptography is key to security
- But cryptography is, alas, not a panacea
- Cryptography is hard to get right



# Topics

- Cryptography defined
- Cryptography from a historical perspective
- Encryption and decryption
- Digital signatures
- Cryptographic hash functions
- Message authentication
- Some other related topics



# Cryptography defined



# Cryptography defined

- Cryptology (*crypto*, from Ancient Greek, meaning “secret” or “hidden”) is the study of codes, both creating and breaking them
- Cryptography is the art of creating codes
- Cryptanalysis is the art of breaking codes

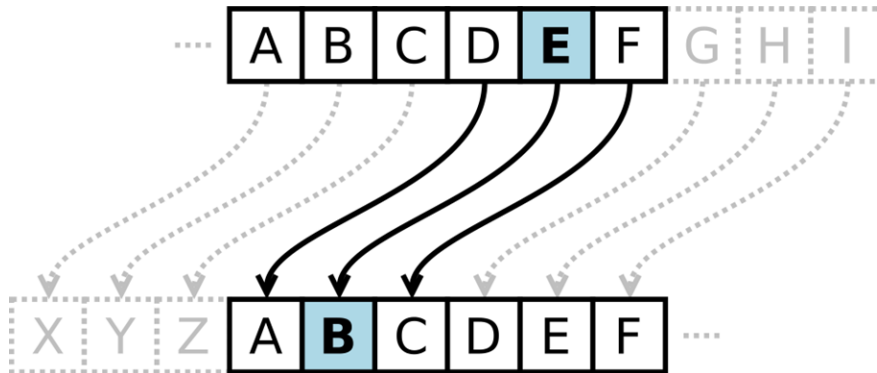


# Cryptography from a historical perspective



# Caesar cipher

- One of the simplest and most widely known encryption techniques – each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet

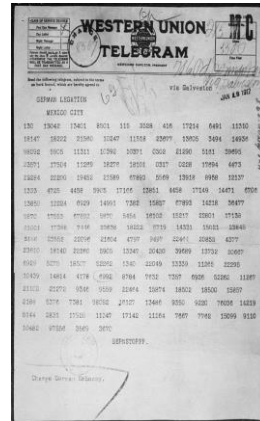


$$E_n(x) = (x + n) \bmod 26$$

$$D_n(x) = (x - n) \bmod 26$$

# World War I

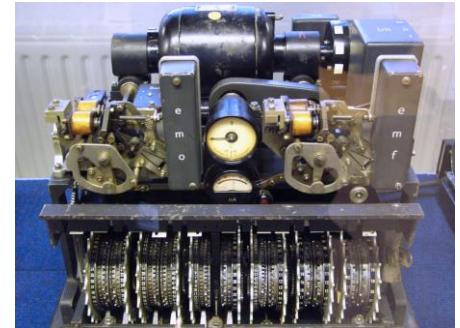
- The Zimmermann Telegram was a secret diplomatic communication from Germany to Mexico that proposed a military alliance between Germany and Mexico if the United States entered World War I against Germany



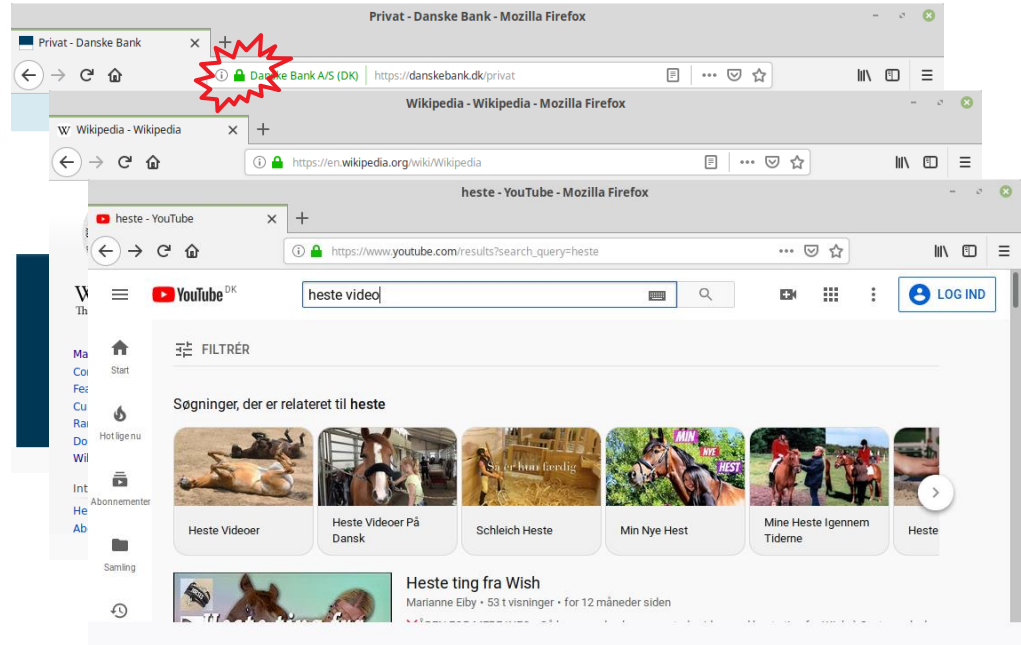
---

# World War II

- Bletchley Park is an English country house that became the principal centre of Allied code-breaking during the Second World War, most importantly the German Enigma and Lorenz ciphers



# Our goal: Secure online communication





# Warm-up question



# FileCrypt

“FileCrypt is a dynamic non-factor based quantum AI encryption hardware solution.

Developed by our cryptographic experts and hardwired into a tamper-resistant USB token.

Plug the token into your PC, start the program and encrypt the files you need to protect”

*What problems do you see with this solution?*



# Multiple concerns

#1: “Developed by our cryptographic experts”

Should we trust proprietary crypto over public peer-reviewed time-tested crypto?

#2: “Dynamic non-factor based quantum AI”

What does that mean? Are there any academic papers that discuss this concept?

#3: “Plug the token into your PC”

Can anyone do this? What if token is lost? Violates **Kerckhoffs’ Principle**

# Kerckhoffs' (2<sup>nd</sup>) Principle

The security of a cryptographic algorithm must rest solely in the secrecy of its **key**, not in the secrecy of the algorithm itself

Collaries:

- Assume attacker knows the algorithm
- Make it available for public analysis
- Protect the key!



Auguste Kerckhoffs  
(1835 – 1903)





# Symmetric cryptosystems



# Symmetric cryptosystems

## Stream ciphers

# One time pad

If  $k$  random,  $|k| \geq |p|$ , never reused, and kept secret, then it is impossible to decrypt or break without knowing the key (Shannon, 1949)

Key	0	1	0	1	1	1	0	0	1	0
Plaintext	1	1	0	0	0	1	1	0	0	0
Ciphertext	1	0	0	1	1	0	1	0	1	0

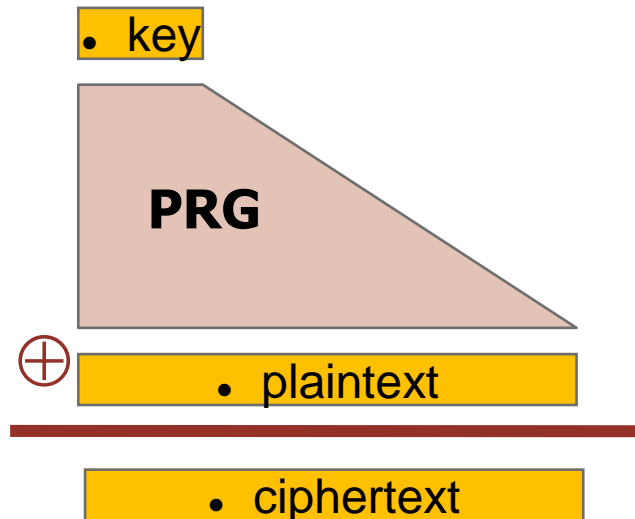
# Towards modern stream ciphers

Problem

OTP key as long as plaintext

Solution

Generate pseudo random keystream



st

# 1 rule of stream ciphers

Never reuse key

$$C_1 \leftarrow P_1 \oplus \text{PRG}(k)$$

$$C_2 \leftarrow P_2 \oplus \text{PRG}(k)$$

$$C_1 \oplus C_2 \rightarrow P_1 \oplus P_2$$

$$P_1 \oplus P_2 \rightarrow P_1, P_2$$



## Solution: Initialisation Vector (IV)

For each message

- Generate IV

- Mix  $k$  with IV

- Generate keystream  $\text{PRG}(k + \text{IV})$  and encrypt

- Send  $c$  and IV (in plaintext)

Change  $k$  before IVs run out

---

# Stream ciphers in the wild



**https://**



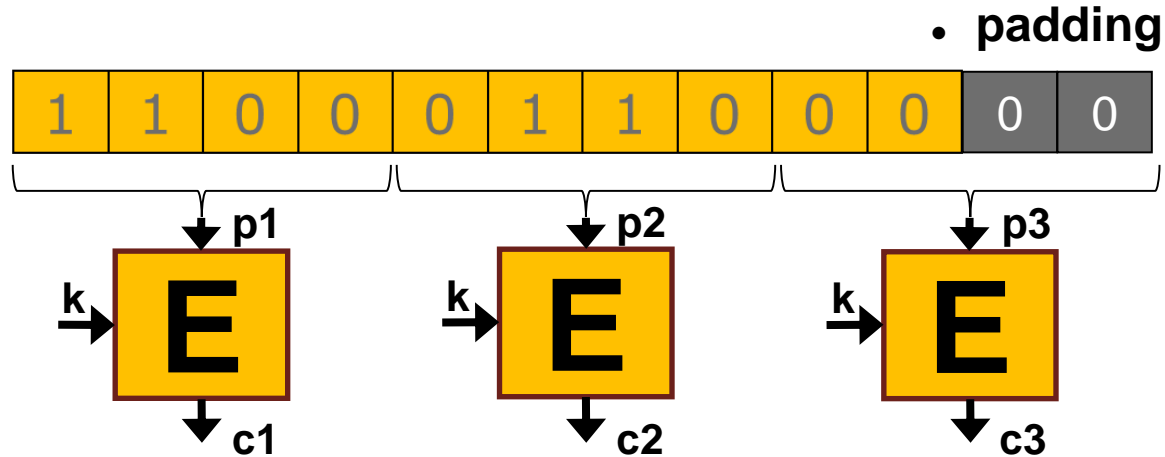


# Block ciphers



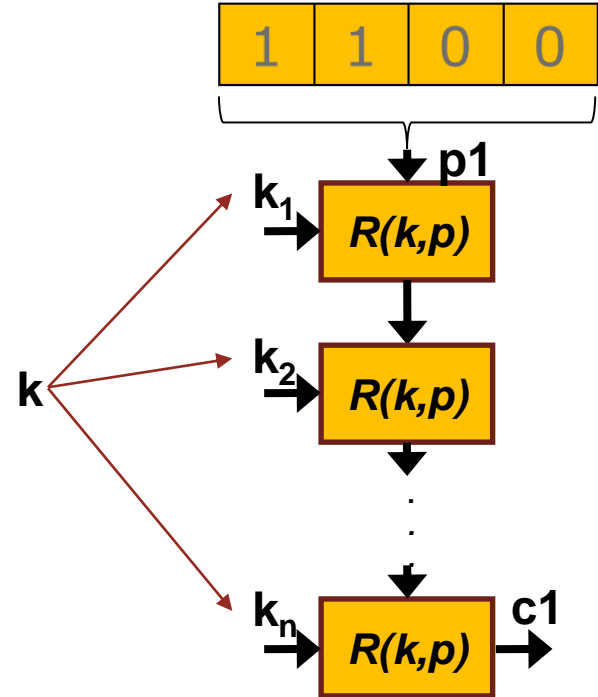
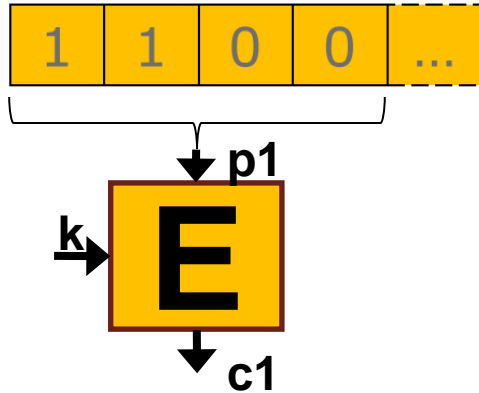
# Block ciphers

One block at a time – as opposed to one bit at a time



# One block at a time

Blocks, rounds function, key schedule, iterations



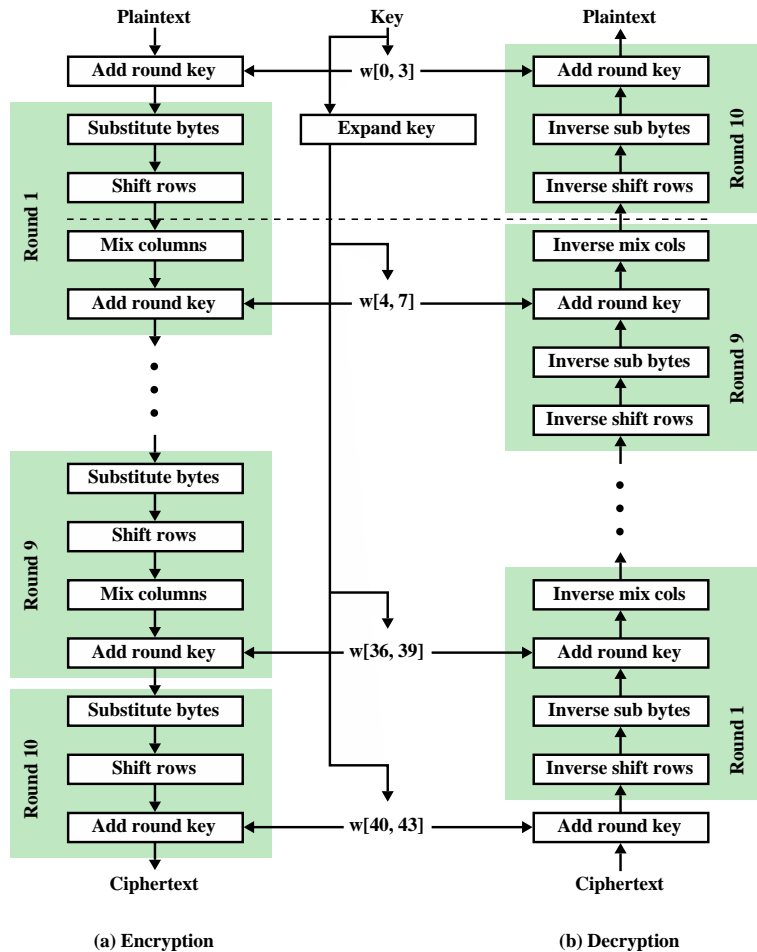
# AES

AES

Keys 128/192/256

Block 128

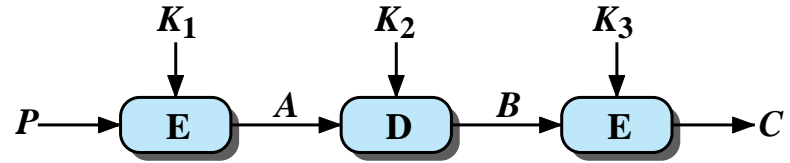
Rounds 10/12/14



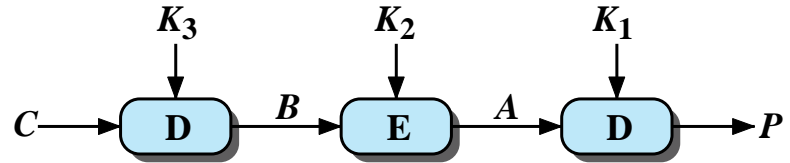
# DES

DES

Key 64, block 64, rounds 16



(a) Encryption

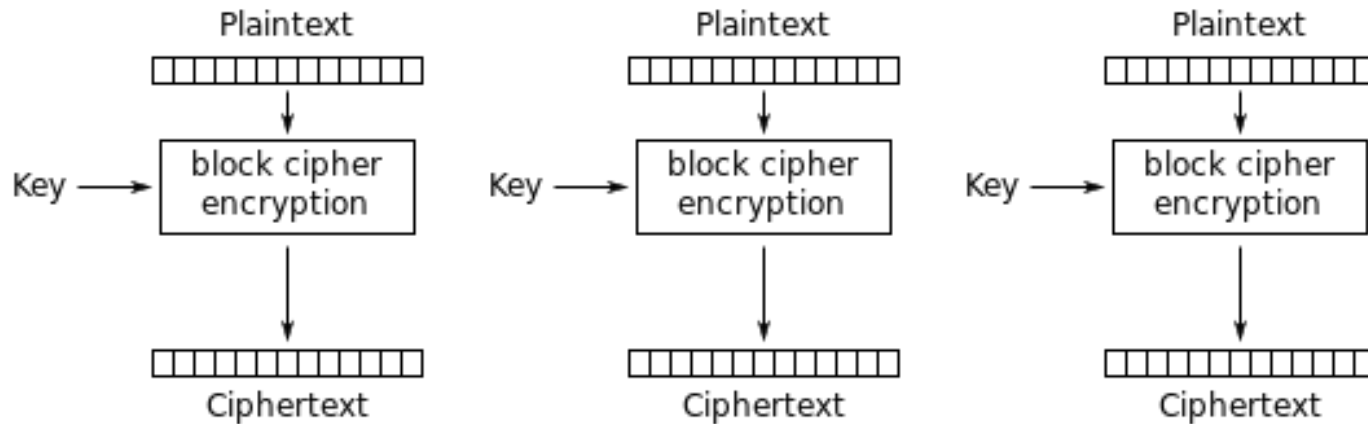


(b) Decryption



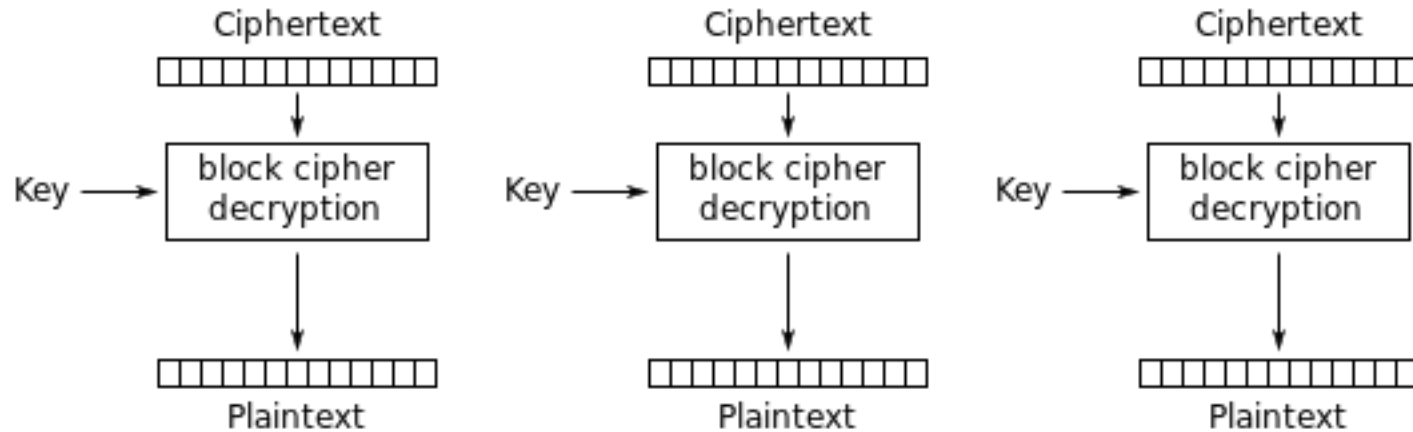
# Modes of operation

# Electronic Codebook (ECB)



Electronic Codebook (ECB) mode encryption

# ECB decryption



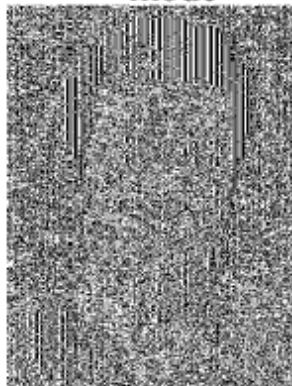
Electronic Codebook (ECB) mode decryption

**If  $p1 = p2$ , then  $c1 = c2$**

An example plaintext

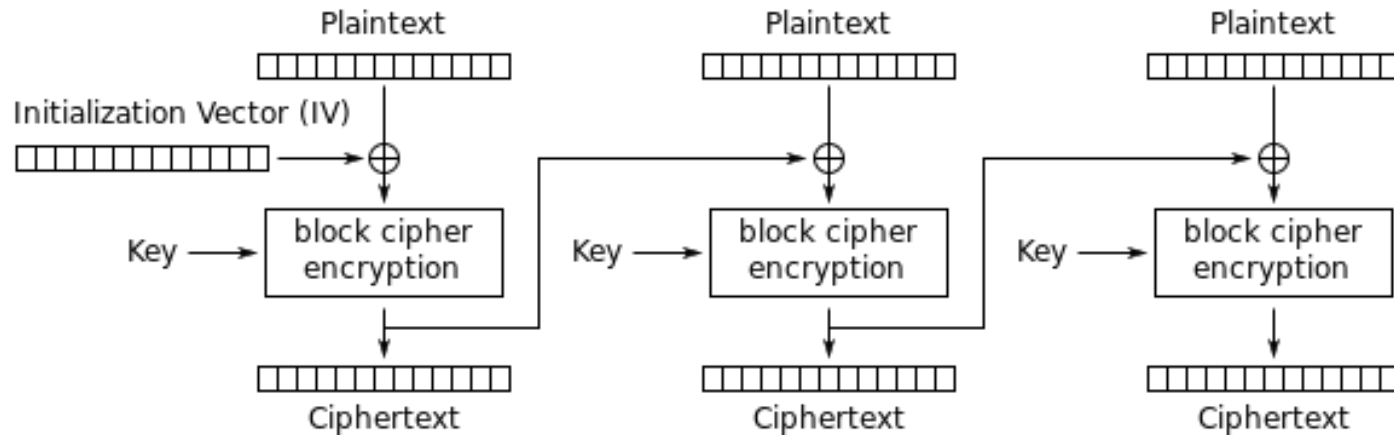


Encrypted with AES in ECB mode



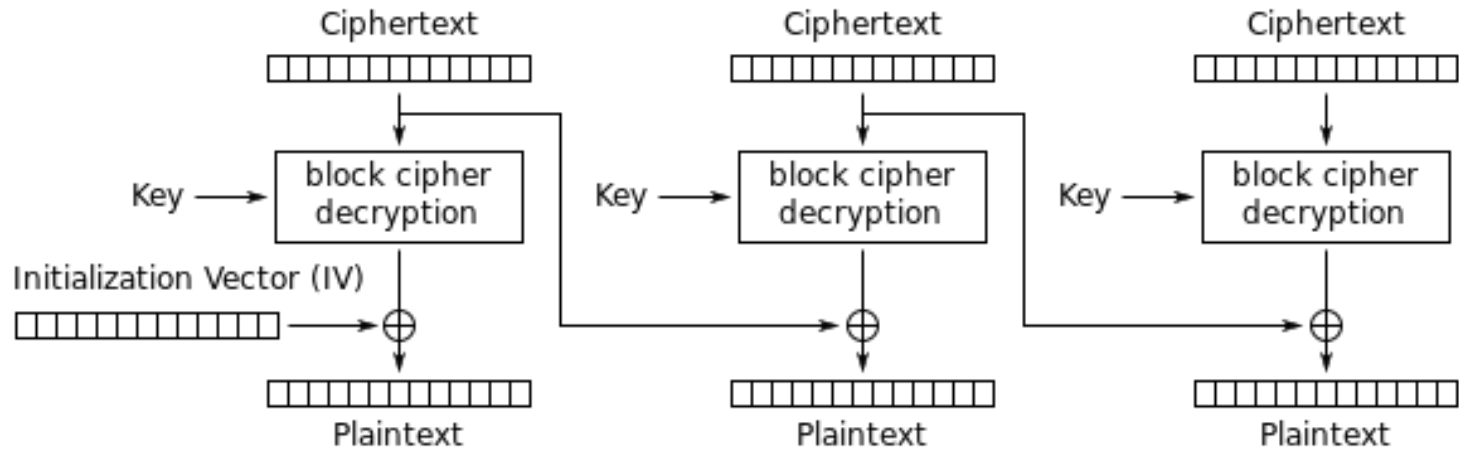


# Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption

# CBC decryption



Cipher Block Chaining (CBC) mode decryption

**Better**

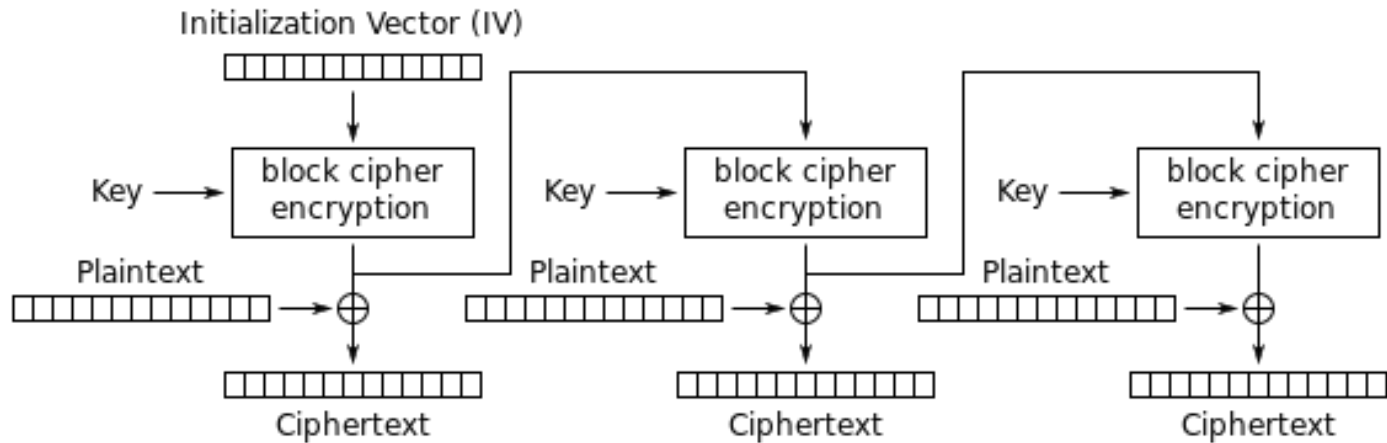
An example plaintext



Encrypted with AES in CBC  
mode



# Output Feedback



Output Feedback (OFB) mode encryption



# Security goals revisited

“Susceptibility to malicious insertions and modifications. Because each symbol is separately enciphered, an active interceptor who has broken the code can splice together pieces of previous messages and transmit a spurious new message that may look authentic.” - Phleeger & Phleeger in Security in Computing, Pearson, 2003

*Is this a disadvantage of stream cipher? Why, why not?*

**Security goal of encryption: Confidentiality**



# Status

*Confidentiality: Check!*

*Integrity: Missing*

# Hands-on

```
Terminal
File Edit View Search Terminal Help

[modes]$ cat myfile
1234567890abcde
1234567890abcde
1234567890abcde
1234567890abcde
[modes]$ openssl enc -aes-128-ecb -e -K 0 -in myfile -nopad | xxd
hex string is too short, padding with zero bytes to length
00000000: 4049 2e80 ddda bc83 0fa2 2096 1d47 c439  @I..... ..G.9
00000010: 4049 2e80 ddda bc83 0fa2 2096 1d47 c439  @I..... ..G.9
00000020: 4049 2e80 ddda bc83 0fa2 2096 1d47 c439  @I..... ..G.9
00000030: 4049 2e80 ddda bc83 0fa2 2096 1d47 c439  @I..... ..G.9
[modes]$ openssl enc -aes-128-cbc -e -K 0 -in myfile -nopad -iv 0 | xxd
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
00000000: 4049 2e80 ddda bc83 0fa2 2096 1d47 c439  @I..... ..G.9
00000010: b42e 0395 8128 e946 ea26 b84c 6c61 7f13  ....(.F.&.Lla..
00000020: 0b52 c084 f04e d7ba f39e 86e3 af54 cf64  .R...N.....T.d
00000030: 1bbc 9a11 8163 8b06 ba0a cdb9 1245 0b0a  ....C.....E..
[modes]$
```



# Message authentication code (MAC)





# Message authentication code

Goal: Provide integrity

Process

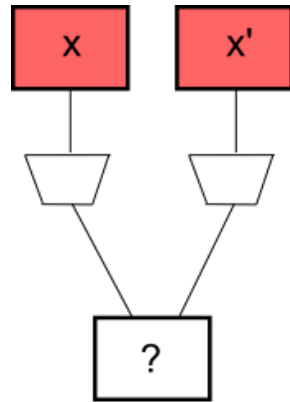
Choose a cryptographic hash function  $h : \{0,1\}^x \rightarrow \{0,1\}^n$

Sender: Send  $h(m), m$

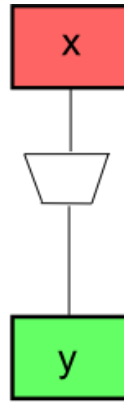
Receiver: Calculate  $h(m)$  and verify it matches  $h(m)$

Examples MD5 ( $n = 128$ ), SHA-256 ( $n = 256$ )

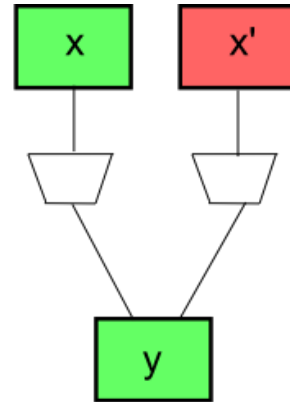
# Cryptographic hash functions



Finding  
Collision



Finding  
Inversion



Finding  
2nd Pre-image



# Hash-based MAC (HMAC)

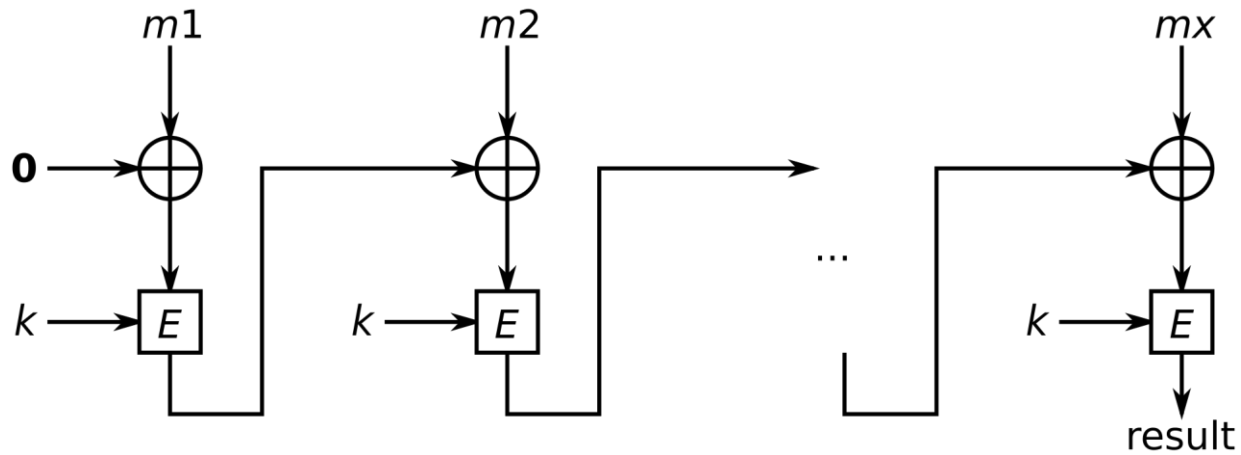
RFC2104: Hash-based MAC

$\text{HMAC}(h,k,m) =$

$$h((k \oplus \text{opad}) \parallel h((k \oplus \text{ipad}) \parallel m))$$

HMAC provides integrity and authenticity

# CBC-MAC





# Car keys

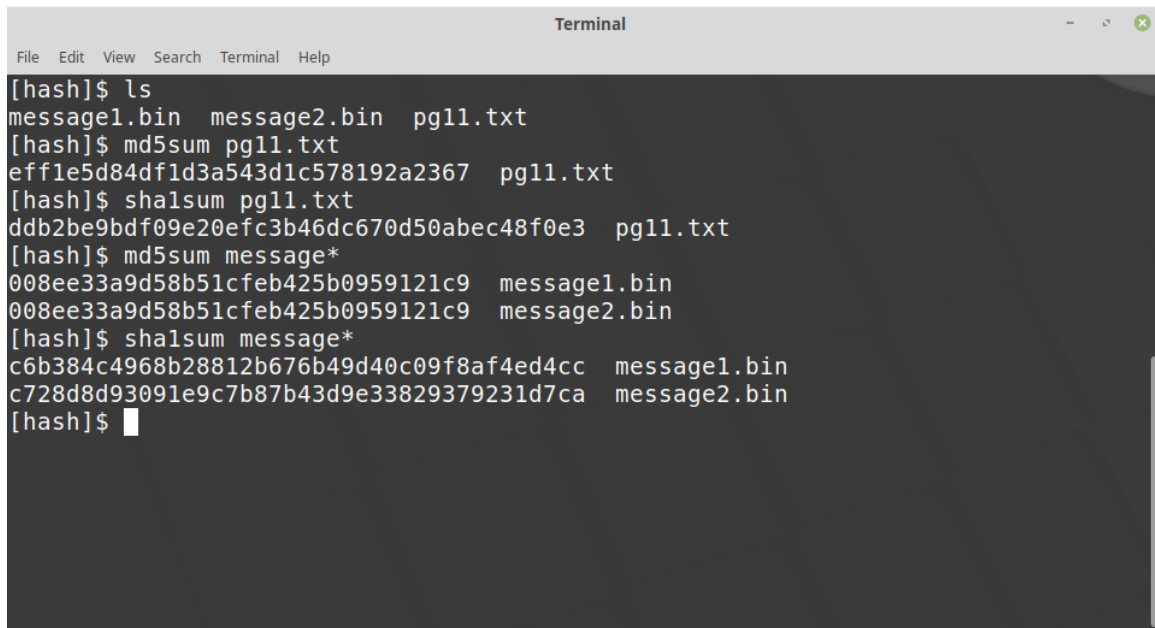
Your car key sends the code for "open the door", together with a MAC, to the car whenever you press the button.

*What could go wrong?*

Replay attack: attacker records message and replays it later

We need some freshness: a timestamp or nonce

# Hands-on



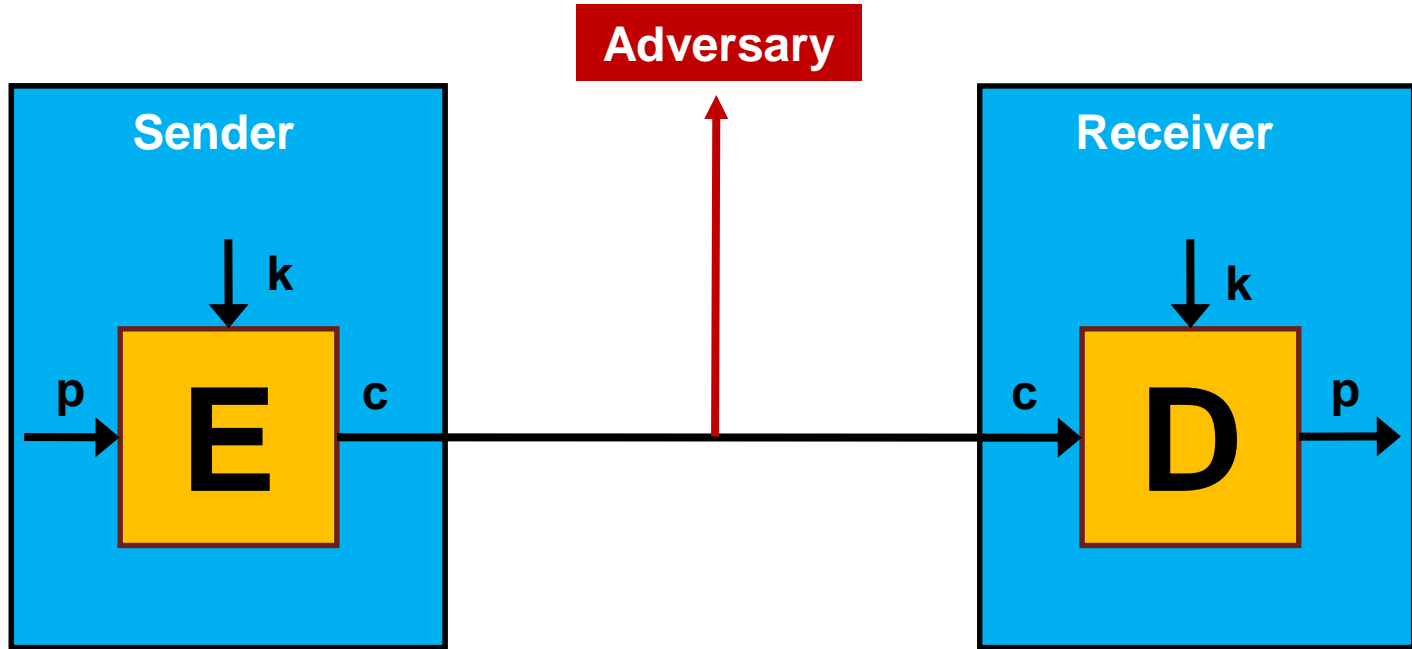
```
Terminal
File Edit View Search Terminal Help

[hash]$ ls
message1.bin message2.bin pg11.txt
[hash]$ md5sum pg11.txt
eff1e5d84df1d3a543d1c578192a2367 pg11.txt
[hash]$ sha1sum pg11.txt
ddb2be9bdf09e20efc3b46dc670d50abec48f0e3 pg11.txt
[hash]$ md5sum message*
008ee33a9d58b51cfcb425b0959121c9 message1.bin
008ee33a9d58b51cfcb425b0959121c9 message2.bin
[hash]$ sha1sum message*
c6b384c4968b28812b676b49d40c09f8af4ed4cc message1.bin
c728d8d93091e9c7b87b43d9e33829379231d7ca message2.bin
[hash]$
```



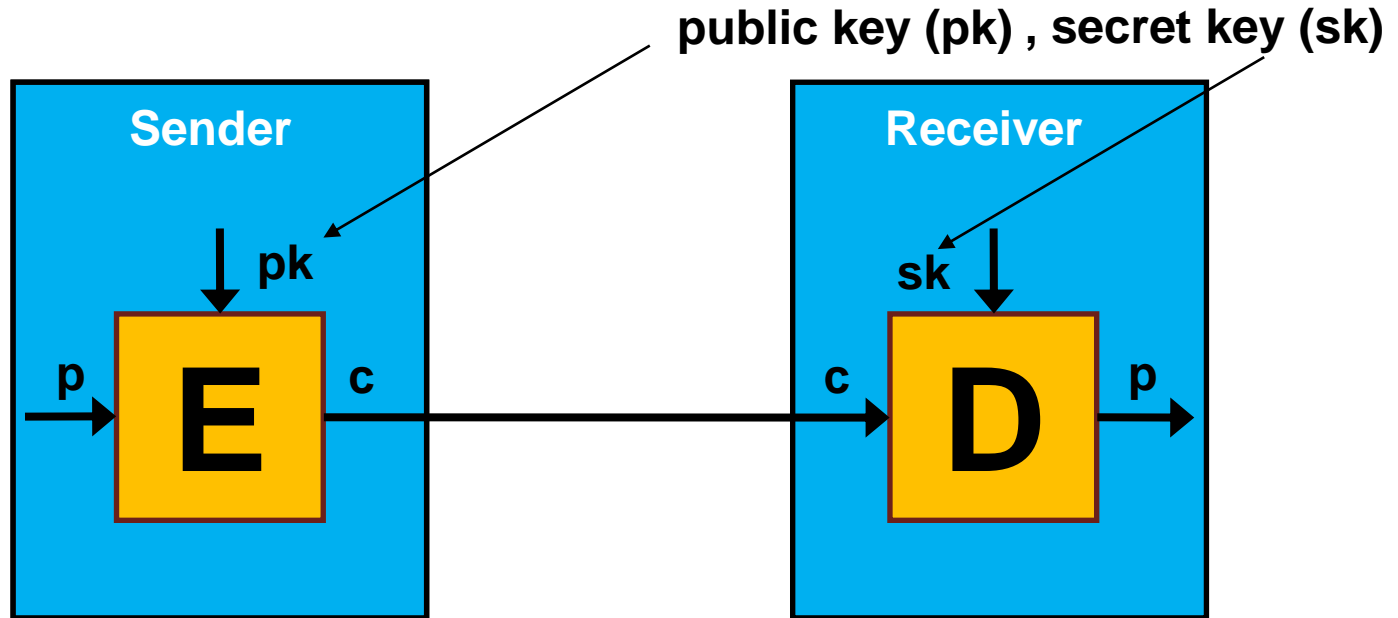
# Non-repudiation

# Cryptosystems





## Enter: Asymmetric encryption



---

# Analogy: Combination locks

Bob sends out locks with combination he only knows

Alice picks one of Bob's locks, places her message in a box and locks it with Bob's lock

Bob is the only one who can open the box now





# No pre-shared key!

Bob

Publish public key, protect private key

Alice

Encrypt message with Bob's public key

Bob

Decrypts with his private key



# Rivest Shamir Adleman (RSA), 1978

First asymmetric cryptosystem



# RSA encryption and decryption

Public key (N,e), private key (d)

$$C = M^e \pmod{N}$$

$$M = C^d \pmod{N}$$

Asymmetric encryption: Yes! But what about non-repudiation?



# Reverse

Public key  $(N, e)$ , private key  $(d)$

Signature  $\text{sig}(M) = M^d \pmod{N}$

Verify  $\text{ver}(M, \text{sig}(M)) = \text{true}$  iff  $M = (M^d)^e \pmod{N}$

# Hands-on

```
Terminal
File Edit View Search Terminal Help

[rsa]$ bc
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software
Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
p=13
q=17
n=p*q
n
221

phi=(p-1)*(q-1)
phi
192

e=5
d=77
e*d % phi
1

m=123

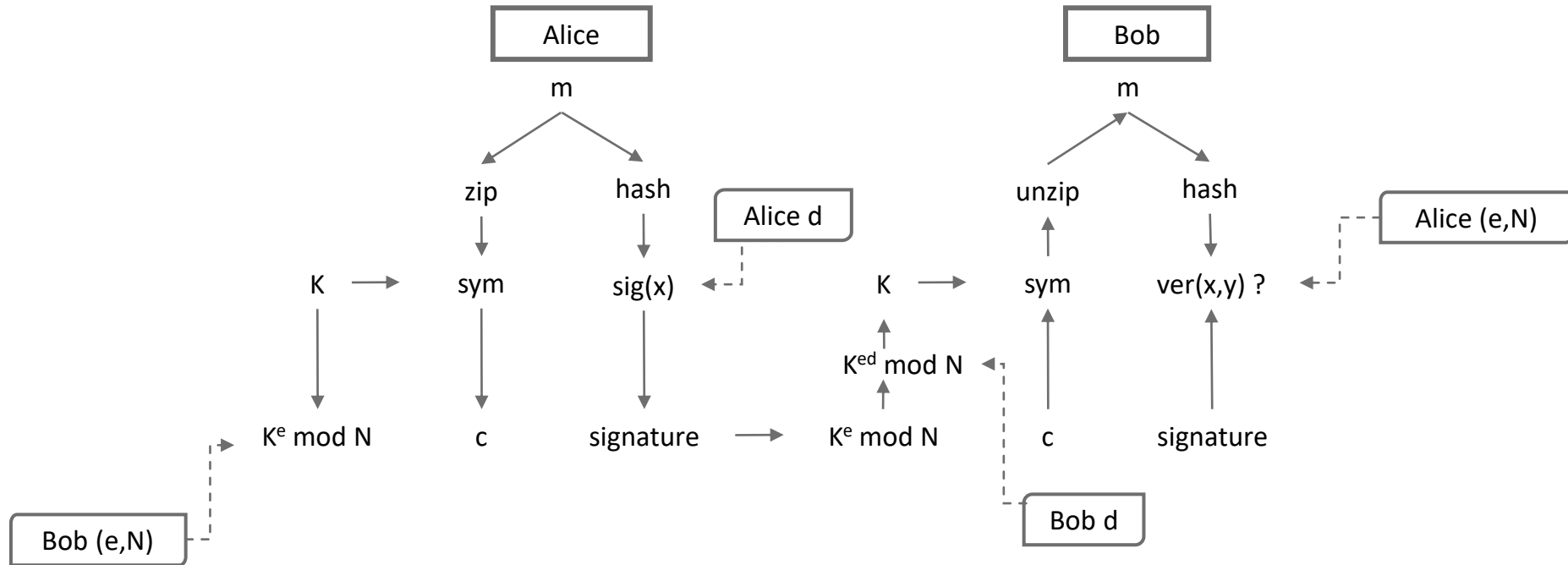
c=m^e % n
c
106

c^d % n
123

m^(e*d)%n
123

quit
[rsa]$ openssl genrsa -out mykeys 2048
```

# Putting it all together







**Later, real-world crypto protocols**



# Wrap-up



# Security goals achieved

Confidentiality

Integrity

Authenticity

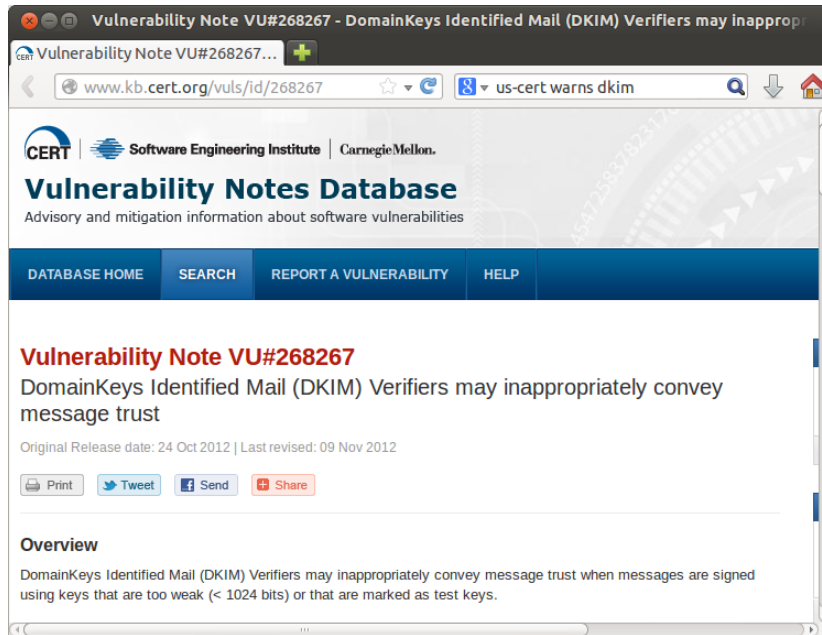
Non-repudiation

CHECK!



**But crypto can still fail**

# Small keys fail



The screenshot shows a web browser window with the title "Vulnerability Note VU#268267 - DomainKeys Identified Mail (DKIM) Verifiers may Inappropriately Convey Message Trust". The address bar shows the URL "www.kb.cert.org/vuls/id/268267". The page header includes the CERT logo and the text "Software Engineering Institute | Carnegie Mellon". The main heading is "Vulnerability Notes Database" with the subtitle "Advisory and mitigation information about software vulnerabilities". A navigation bar contains links for "DATABASE HOME", "SEARCH", "REPORT A VULNERABILITY", and "HELP". The main content area features the title "Vulnerability Note VU#268267" in red, followed by the subtitle "DomainKeys Identified Mail (DKIM) Verifiers may inappropriately convey message trust". Below this, it states "Original Release date: 24 Oct 2012 | Last revised: 09 Nov 2012". There are buttons for "Print", "Tweet", "Send", and "Share". The "Overview" section begins with the text: "DomainKeys Identified Mail (DKIM) Verifiers may inappropriately convey message trust when messages are signed using keys that are too weak (< 1024 bits) or that are marked as test keys."

Vulnerability Note VU#268267 - DomainKeys Identified Mail (DKIM) Verifiers may Inappropriately Convey Message Trust

Vulnerability Note VU#268267...

www.kb.cert.org/vuls/id/268267

us-cert warns dkim

CERT | Software Engineering Institute | Carnegie Mellon.

**Vulnerability Notes Database**

Advisory and mitigation information about software vulnerabilities

DATABASE HOME SEARCH REPORT A VULNERABILITY HELP

**Vulnerability Note VU#268267**

DomainKeys Identified Mail (DKIM) Verifiers may inappropriately convey message trust

Original Release date: 24 Oct 2012 | Last revised: 09 Nov 2012

Print Tweet Send Share

**Overview**

DomainKeys Identified Mail (DKIM) Verifiers may inappropriately convey message trust when messages are signed using keys that are too weak (< 1024 bits) or that are marked as test keys.

# Collision fail



The image is a screenshot of the Ars Technica website. At the top left is the 'ars technica' logo, with 'ars' in a red circle. To its right is an orange banner with the text 'See what Accuweather built for Windows'. Below the logo is a dark navigation bar containing a home icon, 'MAIN MENU', 'MY STORIES: 25', 'FORUMS', 'SUBSCRIBE', and 'VIDEO'. The main heading is 'RISK ASSESSMENT / SECURITY & HACKTIVISM'. The article title is 'Crypto breakthrough shows Flame was designed by world-class scientists', followed by a sub-headline 'The spy malware achieved an attack unlike any cryptographers have seen before.' The byline is 'by Dan Goodin - June 7 2012, 8:20pm -200'. At the bottom right are tags for 'BLACK HAT', 'NATIONAL SECURITY', and a comment icon with the number '161'.

ars technica See what Accuweather built for Windows

MAIN MENU MY STORIES: 25 FORUMS SUBSCRIBE VIDEO

RISK ASSESSMENT / SECURITY & HACKTIVISM

**Crypto breakthrough shows Flame was designed by world-class scientists**

The spy malware achieved an attack unlike any cryptographers have seen before.

by Dan Goodin - June 7 2012, 8:20pm -200

BLACK HAT NATIONAL SECURITY 161

# Impressive fail

## New attack steals e-mail decryption keys by capturing computer sounds

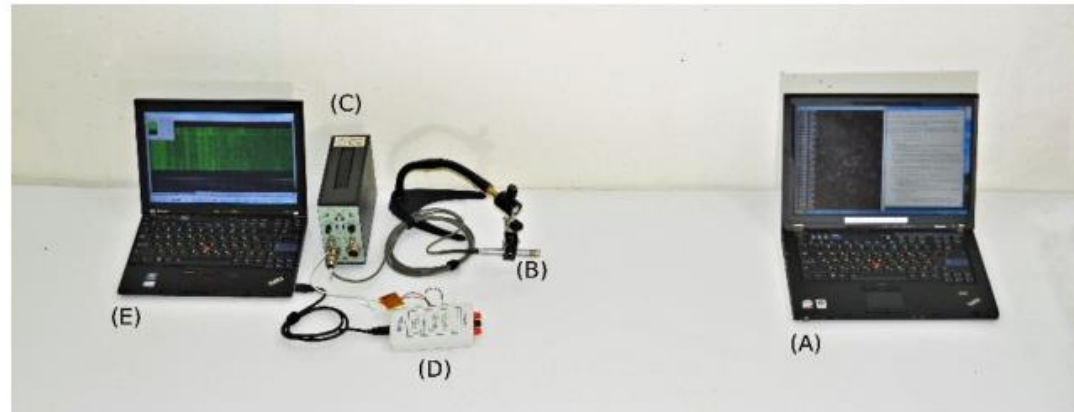
Scientists use smartphone to extract secret key of nearby PC running PGP app.

by Dan Goodin - Dec 18, 2013 11:25 pm UTC

Share

Tweet

108





# Bad choice fail

## IRS Encourages Poor Cryptography

Buried in one of the [documents](#) are the [rules for encryption](#):

While performing AES encryption, there are several settings and options depending on the tool used to perform encryption. IRS recommended settings should be used to maintain compatibility:

- Cipher Mode: ECB (Electronic Code Book).
- Salt: No salt value
- Initialization Vector: No Initialization Vector (IV). If an IV is present, set to all zeros to avoid affecting the encryption.
- Key Size: 256 bits / 32 bytes Key size should be verified and moving the key across operating systems can affect the key size.
- Encoding: There can be no special encoding. The file will contain only the raw encrypted bytes.
- Padding: PKCS#7 or PKCS#5.

ECB? Are they [serious](#)?



# DIY fail



**Smart grid security WORSE than we thought**

OSGP's DIY MAC is a JOKE



# Backdoor fail

Topic: [Security](#)

Follow via:  

## NIST finally dumps NSA-tainted random number algorithm

**Summary:** *Many years since a backdoor was discovered, probably planted by the NSA, public pressure finally forces NIST to formally remove Dual\_EC\_DRBG from their recommendations.*



By [Larry Seltzer](#) for [Zero Day](#) | April 23, 2014 -- 14:04 GMT (07:04 PDT)

Follow [@lseltzer](#)

Comments

2



Vote

1



[more +](#)

# Supply chain fail

## Schneier on Security

[Blog](#)[Newsletter](#)[Books](#)[Essays](#)[News](#)[Talks](#)[Academic](#)[About Me](#)[Home](#) > [Blog](#)

### Crypto AG Was Owned by the CIA

The Swiss cryptography firm Crypto AG sold equipment to governments and militaries around the world for decades after World War II. They were owned by the CIA:

But what none of its customers ever knew was that Crypto AG was secretly owned by the CIA in a highly classified partnership with West German intelligence. These spy agencies rigged the company's devices so they could easily break the codes that countries used to send encrypted messages.

This isn't really news. We have long known that Crypto AG was backdooring crypto equipment for the Americans. What is new is the formerly classified documents describing the details:

The decades-long arrangement, among the most closely guarded secrets of the Cold War, is laid bare in a classified, comprehensive CIA history of the operation obtained by The Washington Post and ZDF, a German public broadcaster, in a joint reporting project.

The account identifies the CIA officers who ran the program and the

#### Search

Powered by [DuckDuckGo](#)

☐[Blog](#)☐[Essays](#)☒[Whole site](#)

#### Subscribe



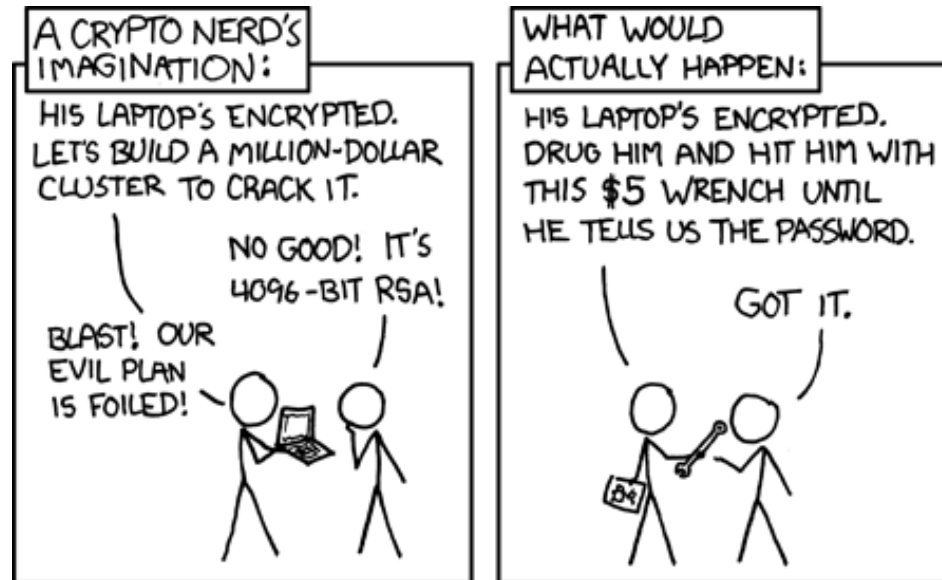
#### About Bruce Schneier



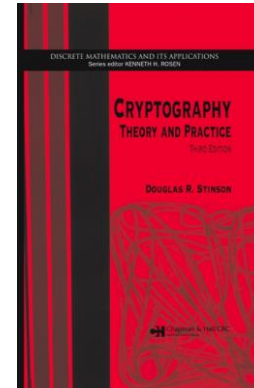
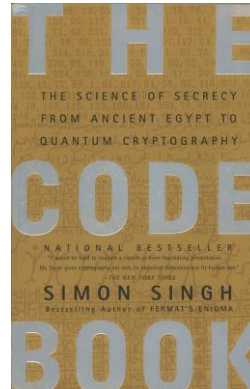
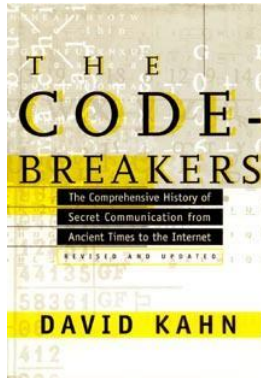
# Malware fail



## Real-world fail



# Suggested reading





# Lecture plan

Week	Date	Time	Lecturer	Topic
36	04-sep	10-12	TL	Security concepts and principles
	08-sep	10-12	TL	Cryptographic building blocks
37	11-sep	10-12	TL	Key establishment and certificate management
	15-sep	10-12	CJ	User authentication, IAM
38	18-sep	10-12	CJ	Operating systems security, web, browser and mail security
	22-sep	10-12	CJ	IT security management and risk assessment
39	25-sep	10-12	TL	Software security - exploits and privilege escalation
	29-sep	10-12	TL	Malicious software
40	02-okt	10-12	CJ	Firewalls and tunnels, security architecture
	06-okt	10-12	CJ	Cloud and IoT security
41	09-okt	10-12	TL	Intrusion detection and network attacks
	13-okt	10-12	TL	Forensics
42				Fall Vacation - No lectures
43	23-okt	10-12	CJ	Privacy and GDPR
	27-okt	10-12	CJ	Privacy engineering
44	30-okt	10-12	CJ,TL	Final guest lecture and Exam Q/A