# New developments in cryptology

Prof. Bart Preneel

COSIC

Bart.Preneel(at)esatDOTkuleuven.be

http://homes.esat.kuleuven.be/~preneel

February 2011

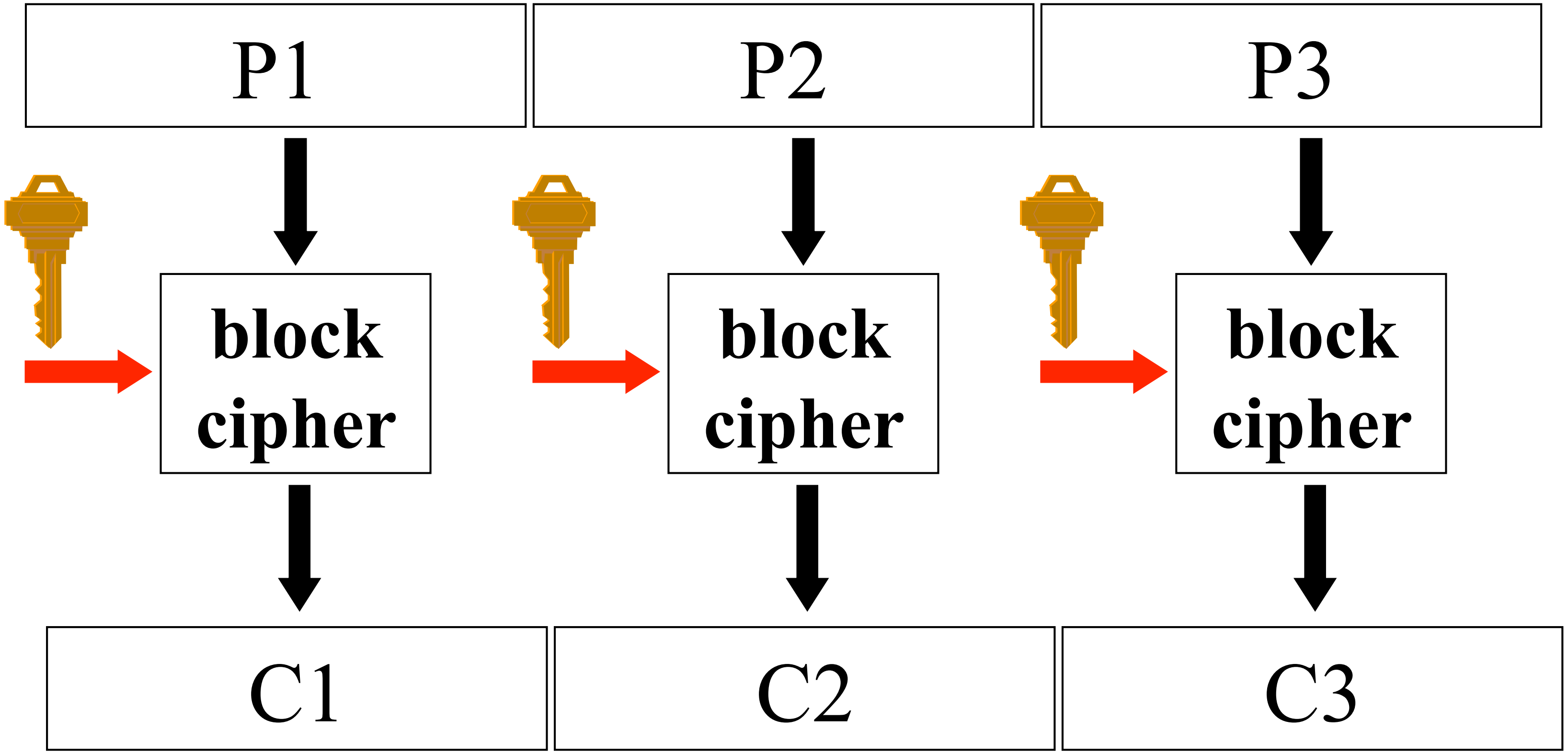
# Outline

- 1. Cryptology: concepts and algorithms
- 2. Cryptology: protocols
- 3. Public-Key Infrastructure principles
- 4. Networking protocols
- **5. New developments in cryptology**
- 6. Cryptography best practices
- 7. Hash functions

# Outline

- Block ciphers/stream ciphers/MAC algorithms
- Modes of operation and authenticated encryption
- How to encrypt using RSA
- Algorithm: secure design and implementation
- Obfuscation
- SPAM fighting

# Block ciphers



- larger data units: 64…128 bits
- memoryless
- repeat simple operation (round) many times

# Block ciphers: Keeloq

- Microchip Inc algorithm, designed in the 1980s
- Allegedly used in large % of the cars for car locks, car alarms
- Block cipher with 32-bit blocks, 64-bit keys and 528 simple rounds
- Leaked on the internet early 2007

[redacted]

# Block ciphers: Keeloq (2)

[Bogdanov07] Car key = Master key + Car ID

[Biham-Dunkelman-Indesteeghe-Keller-Preneel07]:

- 1 hour access to token + 2 days of calculation

[Eisenbarth-Kasper-Moradi-Paar-Salmasizadeh-Manzuri ShalmaniPaar 08]

- **Side channel attack allows to recover master key in hopping mode**

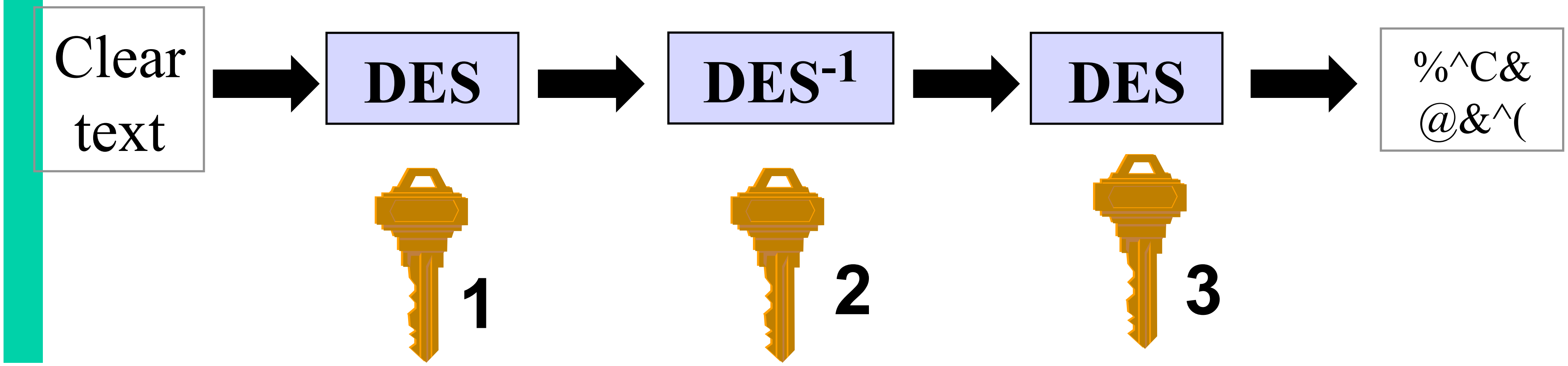
in 2011 cryptographers will drive expensive cars

# 3-DES: NIST Spec. Pub. 800-67

(May 2004)

- Single DES abandoned
- two-key triple DES: until 2009 (80 bit security)
- three-key triple DES: until 2030 (100 bit security)

Highly vulnerable to a related key attack



Clear text

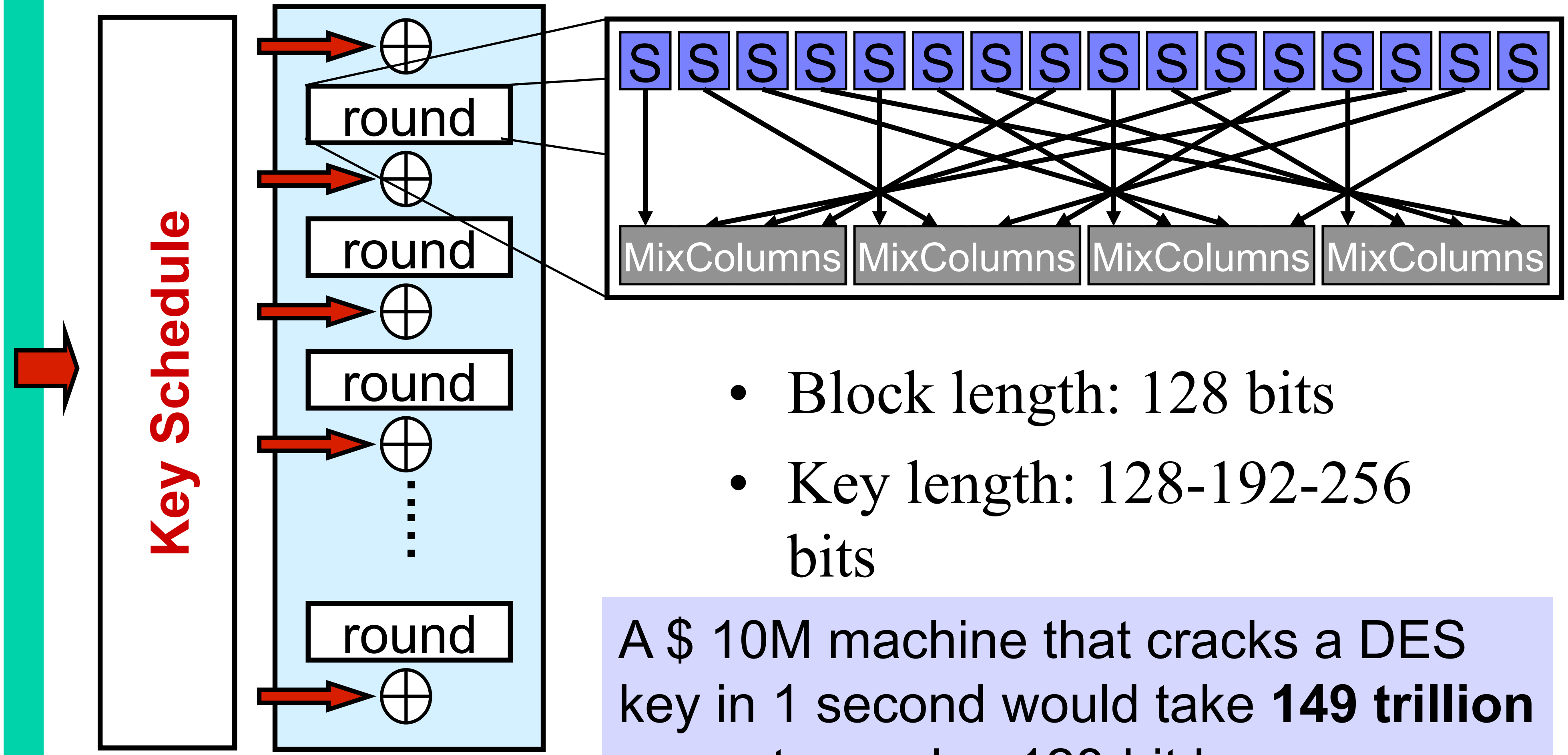
DES

$\mathbf{DES^{-1}}$

DES

%^C& @&^(

# AES (2001)

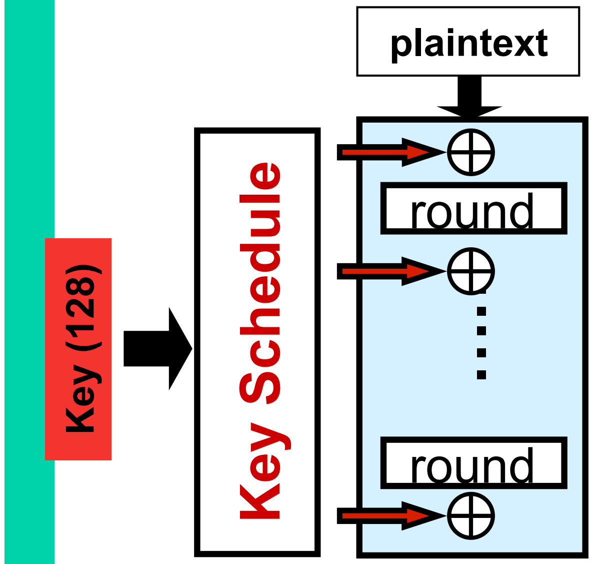


- Block length: 128 bits
- Key length: 128-192-256 bits

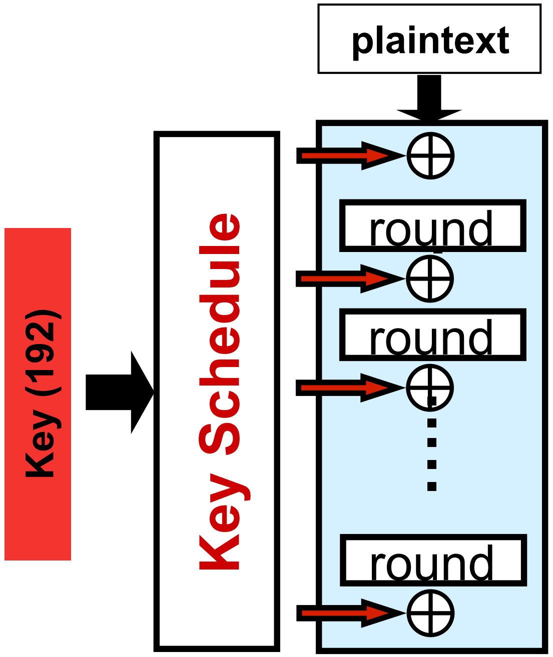
A $ 10M machine that cracks a DES key in 1 second would take **149 trillion years** to crack a 128-bit key

# AES variants (2001)

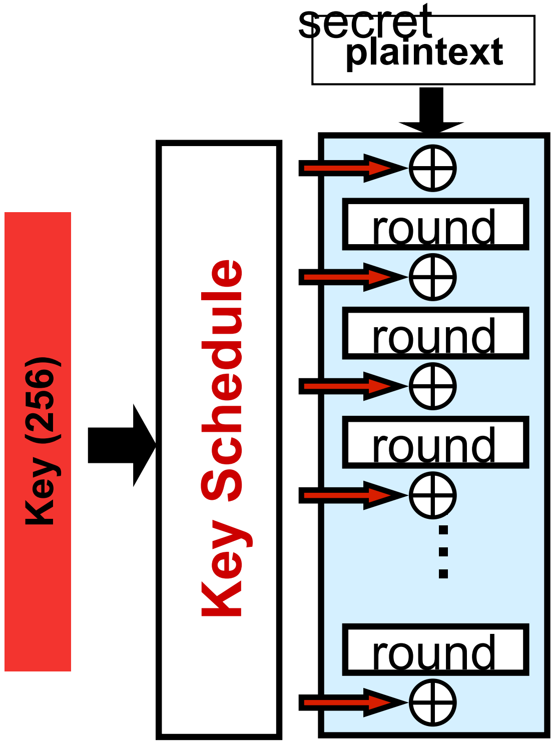
- AES-128
- 10 rounds
- sensitive



- AES-192
- 12 rounds
- classified



- AES-256
- 14 rounds
- secret and top secret



Light weight key schedule, in particular for the 256-bit version

# AES implementations: efficient/compact

- NIST validation list: 1187 implementations (2008: 879)
  http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html

- HW: 43 Gbit/s in 130 nm CMOS [‘05]
- Intel: new AES instruction: 0.75 cycles/byte [’09-’10]
- SW: 7.6 cycles/byte on Core 2 or 110 Mbyte/s bitsliced [Käsper-Schwabe’09]

- HW: most compact: 3600 gates
  - KATAN: 1054, PRESENT: 1570

# AES: security

- cryptanalysis: no attack has been found that can exploit this structure (in spite of the algebraic “attack” [Courtois’02])

- implementation level attack
  - cache attack precluded by bitsliced implementations or by special hardware support
  - fault attack requires special countermeasures

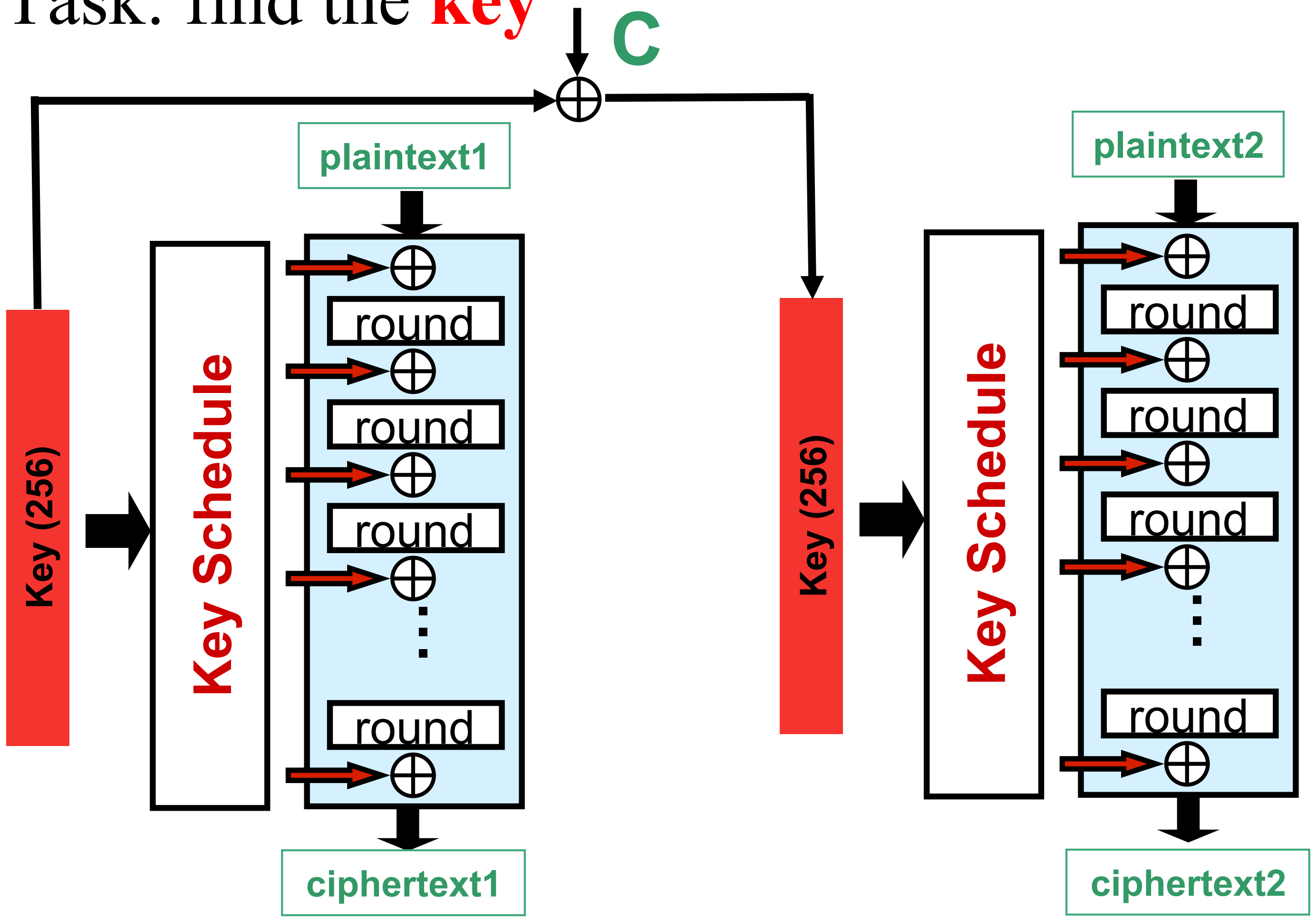
# AES-256 security

- Exhaustive key search on AES-256 takes $2^{256}$ encryptions
  - $2^{64}$: 10 minutes with $ 5M
  - $2^{80}$: 2 year with $ 5M
  - $2^{120}$ : 1 billion years with $ 5B
- [Biryukov-Khovratovich'09] **related key attack on AES-256**
  - requires $\mathbf{2^{119}}$ encryptions with 4 related keys,
  - data & time complexity $\mathbf{2^{119} << 2^{256}}$
- Why does it work? Very lightweight key schedule

- Is AES-256 broken? No, only an academic "weakness" that is easy to fix
- No implications on security of AES-128 for encryption
- Do not use AES-256 in a hash function construction

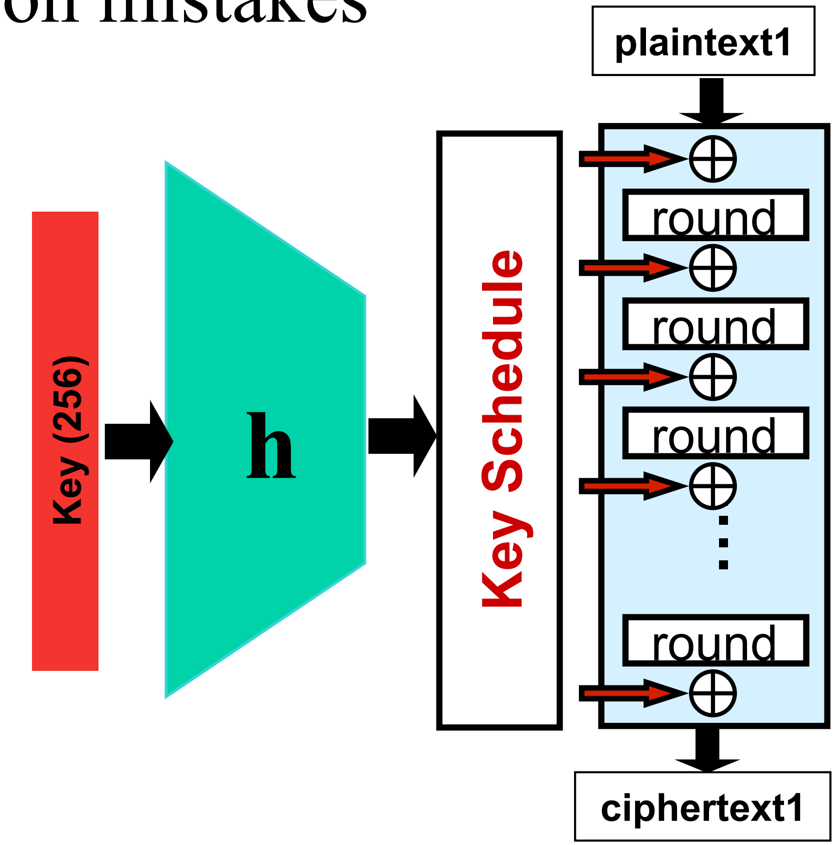
# What is a related key attack?

- Attacker chooses **plaintexts** and **key difference** C
- Attacker gets **ciphertexts**
- Task: find the **key**

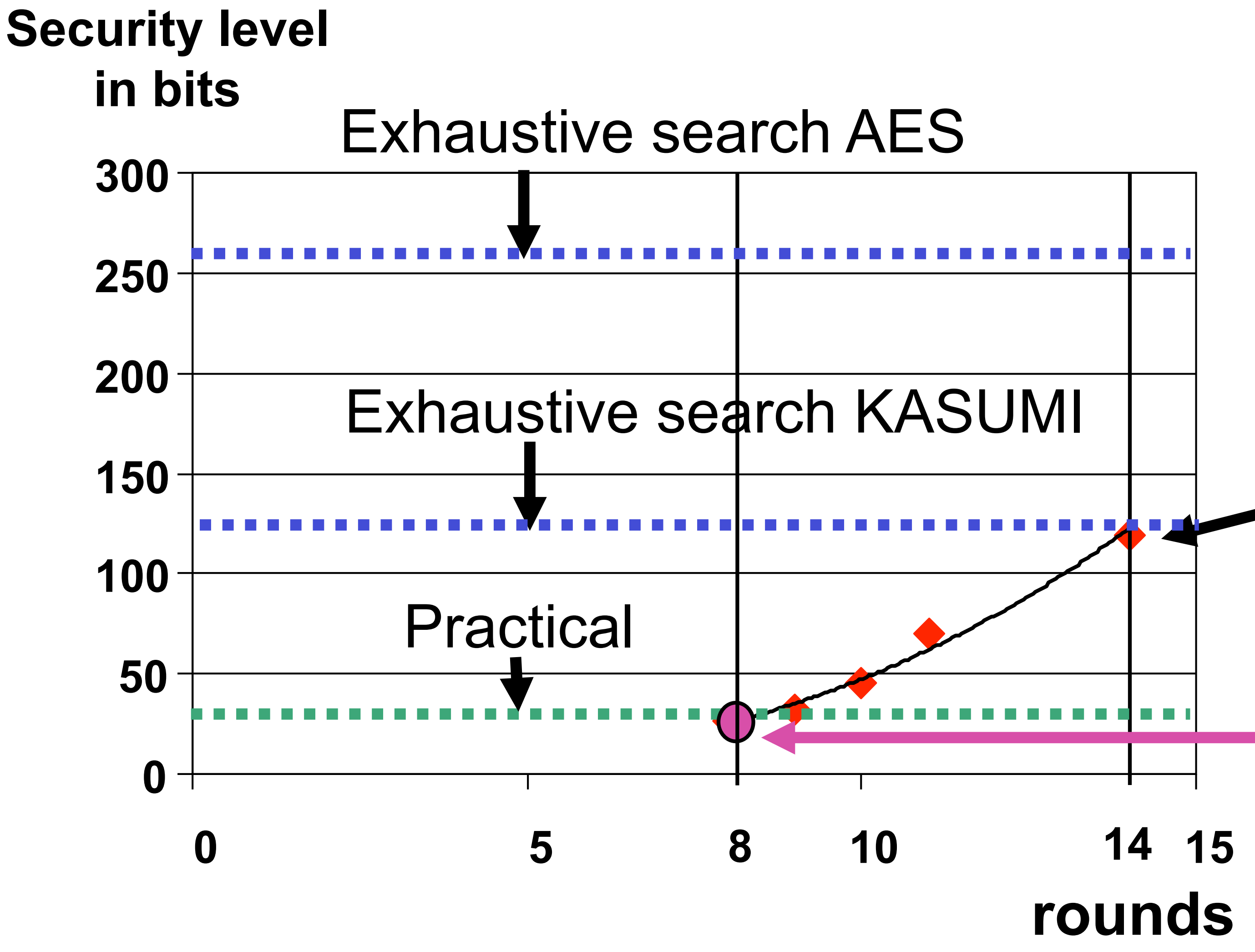


# Should I worry about a related key attack?

- Very hard in practice (except some old US banking schemes)
- If you are vulnerable to a related key attack, you are making very bad implementation mistakes
- This is a very powerful attack model: if an opponent can zeroize 96 key bits of his choice (rather than adding a value), he can find the key in a few seconds
- If you are worried, hashing the key is an easy fix



# Related key attacks on AES-256 and KASUMI



[Biryukov-Khovratovich'09]
4 related keys
data & time complexity
$2^{119} << 2^{256}$

[Dunkelman-Keller-Shamir'10]
Related key attack: 4 keys, $2^{26}$ data & $2^{32}$ time << $2^{128}$

# KASUMI (2002)

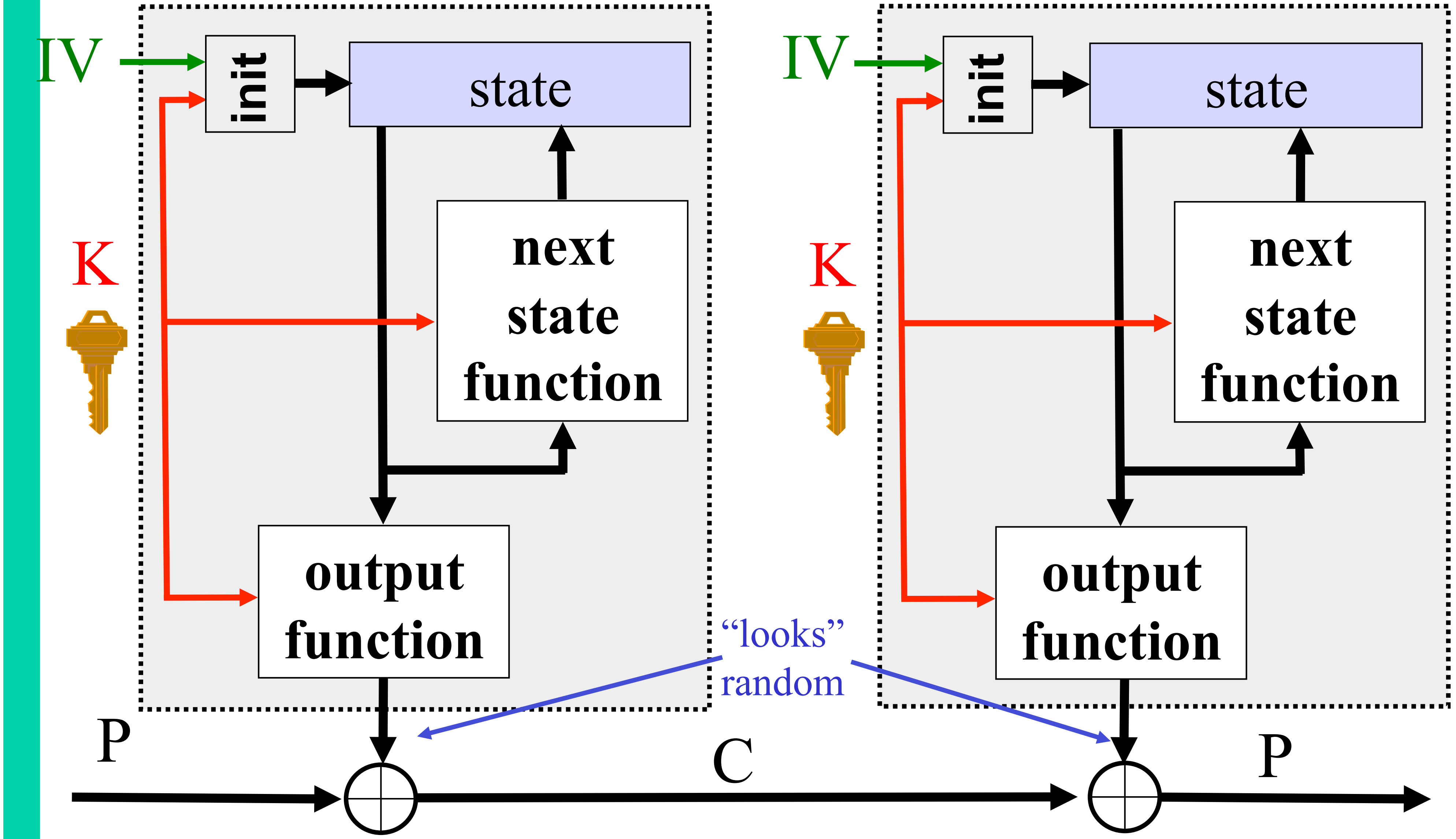
- Widely used in all 3G phones
- Present in 40% of GSM phones but not yet used

- Good news: related key attacks do not apply in  in the GSM or 3G context

# KASUMI
## [Dunkelman-Keller-Shamir'09]

- Practical related key attack announced in December 2009 on the block cipher KASUMI used in 3GPP
  - 4 related keys, $2^{26}$ data, $2^{30}$ bytes of memory, and $2^{32}$ time
- It is not possible to carry out this attack in 3G (as related keys are not available)

# Synchronous Stream Cipher (SSC)



# Stream ciphers

- historically very important (compact)
  - LFSR-based: A5/1, A5/2,  E0 – practical attacks known
  - software-oriented: RC4 – serious weaknesses
  - block cipher in CTR or OFB (slower)
- today:
  - many broken schemes
  - exception: SNOW2.0, MUGI
  - lack of standards and open solutions

# Open competition for stream ciphers

http://www.ecrypt.eu.org

- run by ECRYPT
  - high performance in software (32/64-bit): 128-bit key
  - low-gate count hardware (< 1000 gates): 80-bit key
  - variants: authenticated encryption
- April 2005: 33 submissions
- many broken in first year
- April 2008: end of competition

# The eSTREAM Portfolio

## Apr. 2008 (Rev1 Sept. 2008)

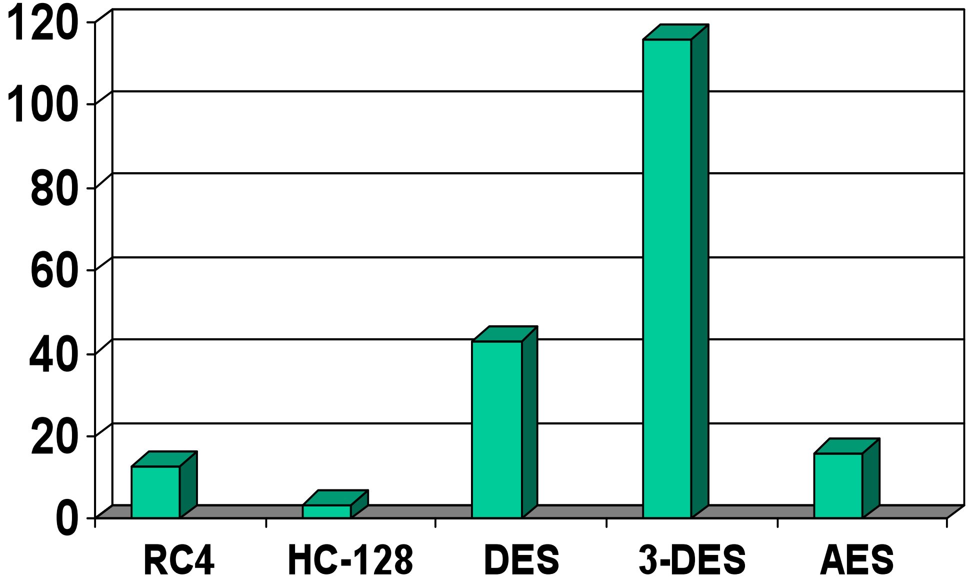
(in alphabetical order)

| Software | Hardware |
|---|---|
| HC-128 | ~~F-FCSR-H~~ |
| Rabbit | Grain v1 |
| Salsa20/12 | MICKEY v2 |
| Sosemanuk | Trivium |
| 3-10 cycles per byte | 1500..3000 gates |

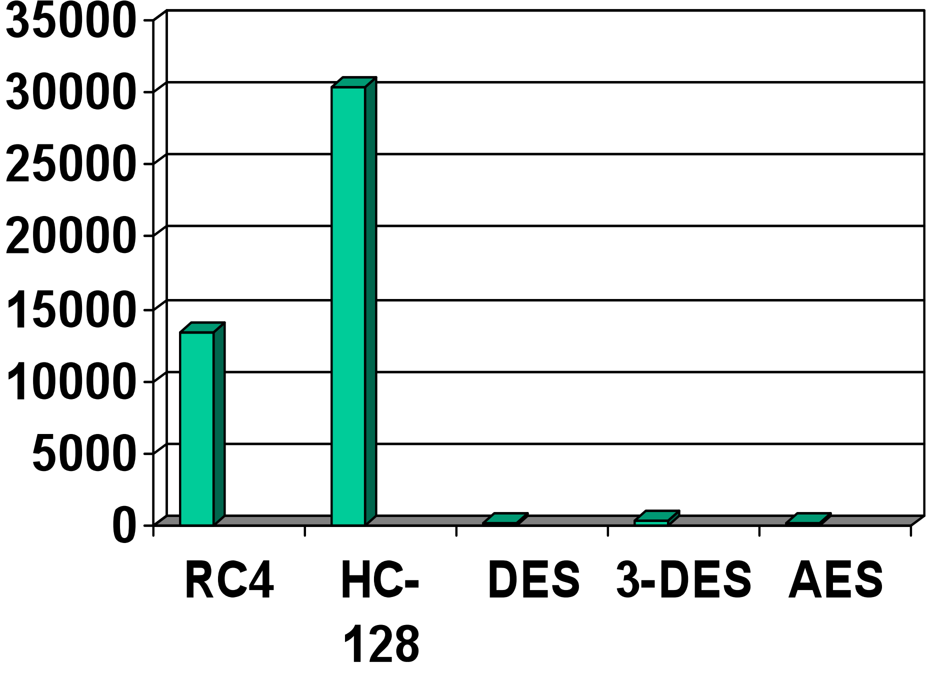
# Performance reference data

(Pentium M 1.70GHz Model 6/9/5)

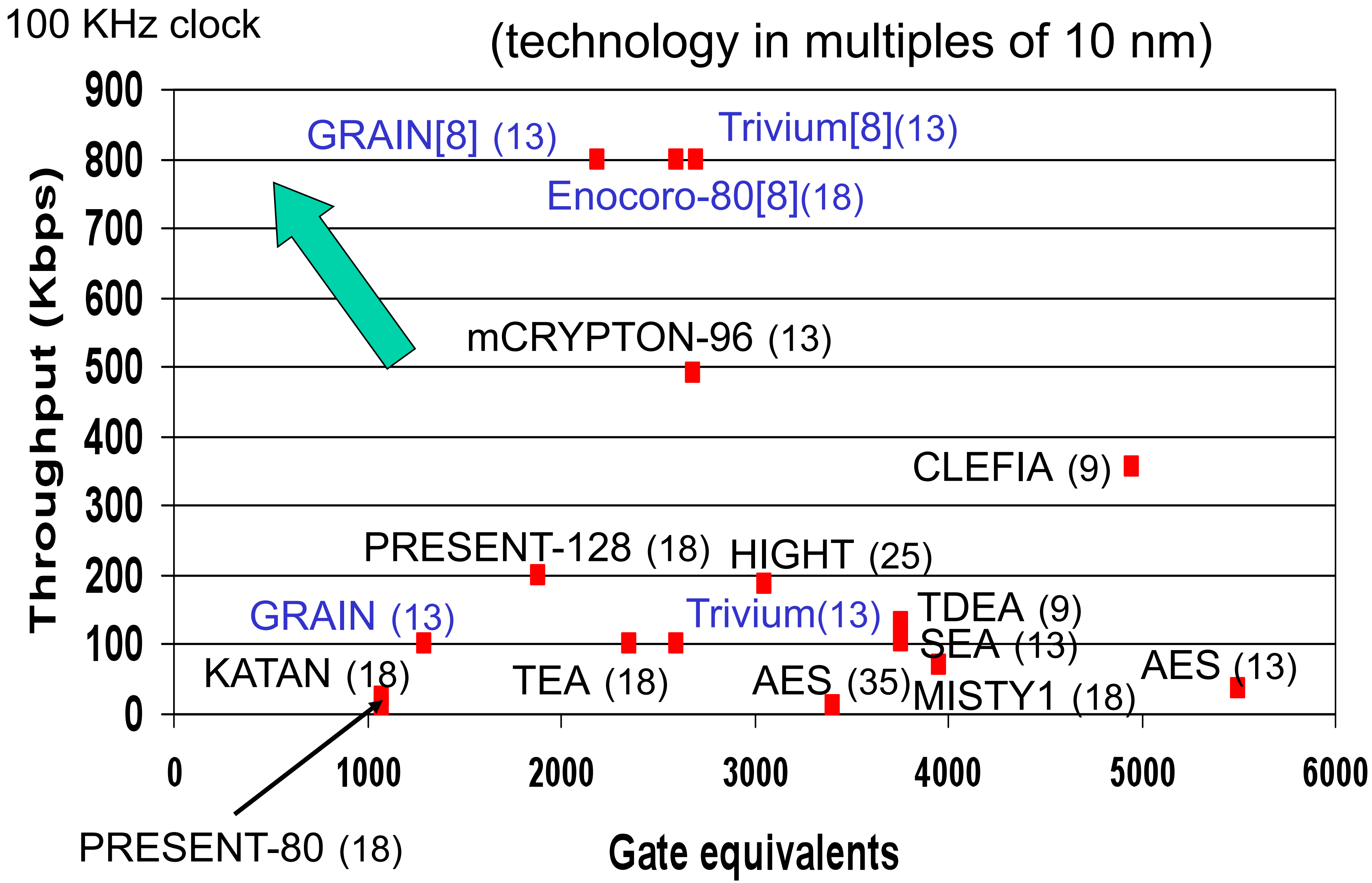
encryption speed (cycles/byte)



key setup (cycles)



# Low cost hw: throughput versus area



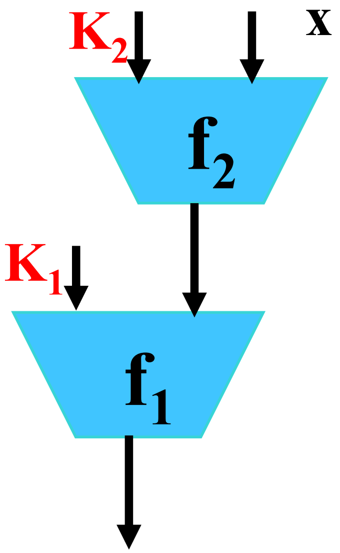
# MAC algorithms

- EMAC based on AES
- HMAC based on MD5/SHA-1
- GMAC
- UMAC

- NIST: 2 standards for authenticated encryption
  - CCM: CTR  + CBC-MAC  [NIST SP 800-38C]
  - GCM: CTR + GMAC [NIST SP 800-38C]

# HMAC based on MDx, SHA

- Widely used in SSL/TLS/IPsec
  - Attacks not yet dramatic
- NMAC weaker than HMAC



| | Rounds in f1 | Rounds in f2 | Data complexity |
|---|---|---|---|
| MD4 | 48 | 48 | $2^{88}$ CP & $2^{95}$ time |
| MD5 | 64 | 33 of 64 | $2^{126}$ CP |
| MD5 | 64 | 64 | $2^{51}$ CP & $2^{100}$ time (RK) |
| SHA(-0) | 80 | 80 | $2^{109}$ CP |
| SHA-1 | 80 | 43 of 80 | $2^{154.9}$ CP |

# GMAC: polynomial MAC (NIST SP 800-38D ‘07 + 3GSM)

- keys $K_1$, $K_2 \in GF(2^{128})$
- input $x$: $x_1, x_2, \ldots, x_t$, with $x_i \in GF(2^{128})$

$$\bullet\ g(x) = K_1 + \Sigma_{i=1}^{t}\ x_i \bullet (K_2)^i$$

- in practice: compute $K_1 = \mathrm{AES}_K(n)$ (CTR mode)

- properties:
  - fast in software and hardware (support from Intel)
  - not very robust w.r.t. nonce reuse, truncation, MAC verifications, due to reuse of $K_2$ *(not in 3GSM!)*
  - versions over GF(p) (e.g. Poly1305-AES) seem more robust

# UMAC RFC 4418 (2006)

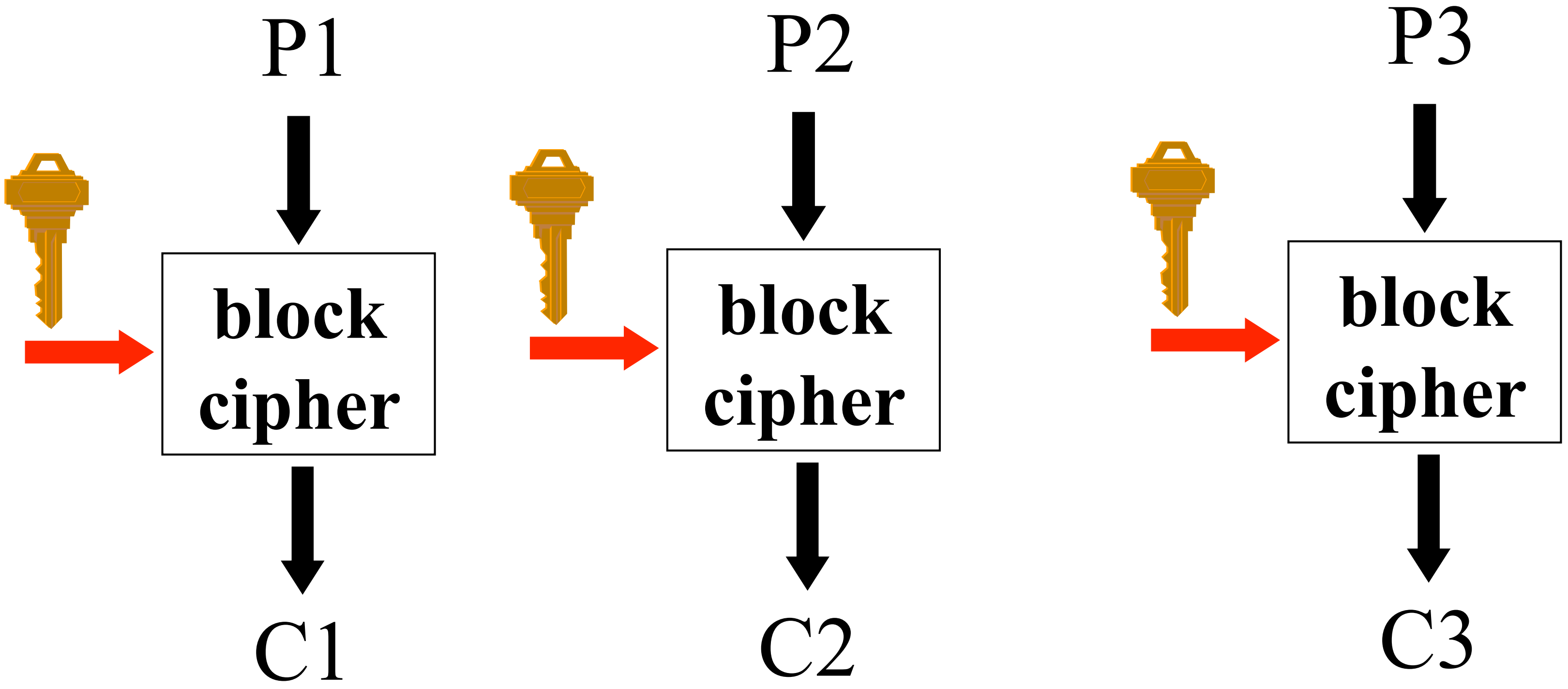
- key $K, k_1, k_2 .., k_{256} \in GF(2^{32})$ *(1024 bytes)*
- input $x$: $x_1, x_2, \ldots, x_{256}$, with $x_i \in GF(2^{32})$
- $g(x) = prf_K(h(x))$
- $h(x) = (\sum_{i=1}^{512} (x_{2i-1} + k_{2i-1}) \bmod 2^{32} \cdot (x_{2i} + k_{2i}) \bmod 2^{32}) \bmod 2^{64}$
- properties
  - software performance: 1-2 cycles/byte
  - forgery probability: $1/2^{30}$ (provable lower bound)
  - [Handschuh-Preneel08] full key recovery with $2^{40}$ verification queries

# How to use cryptographic algorithms

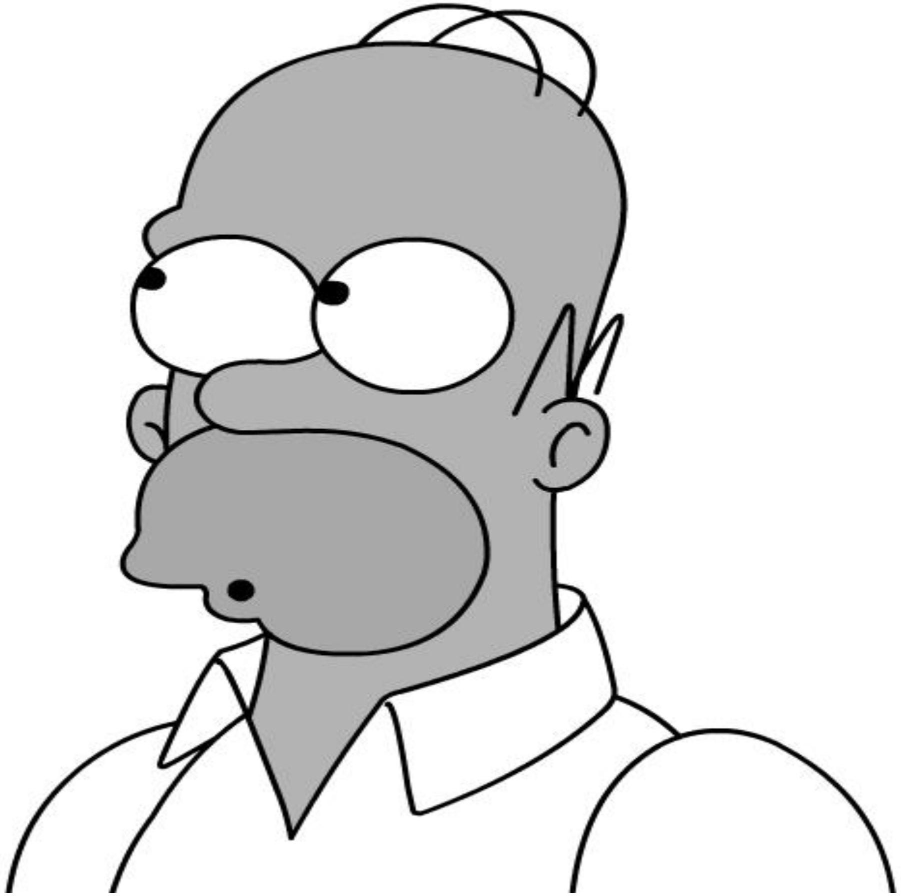
- Modes of operation
- Padding and error messages
- Authenticated encryption

- How to encrypt with RSA

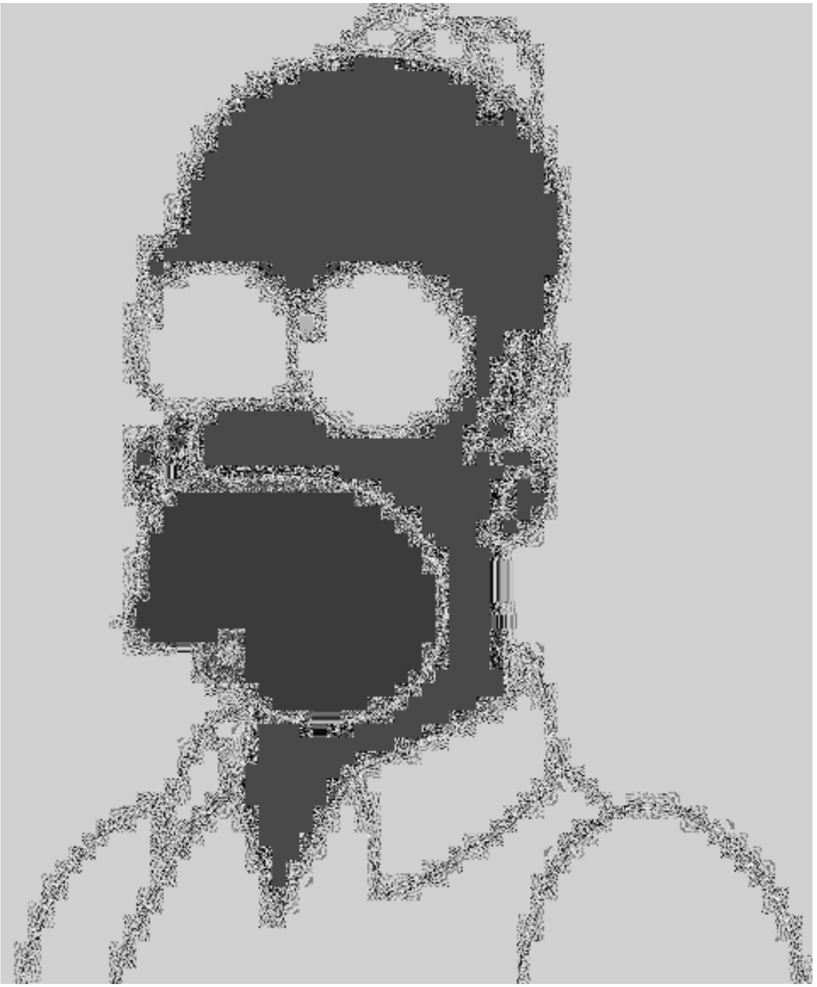
# How NOT to use a block cipher: ECB mode

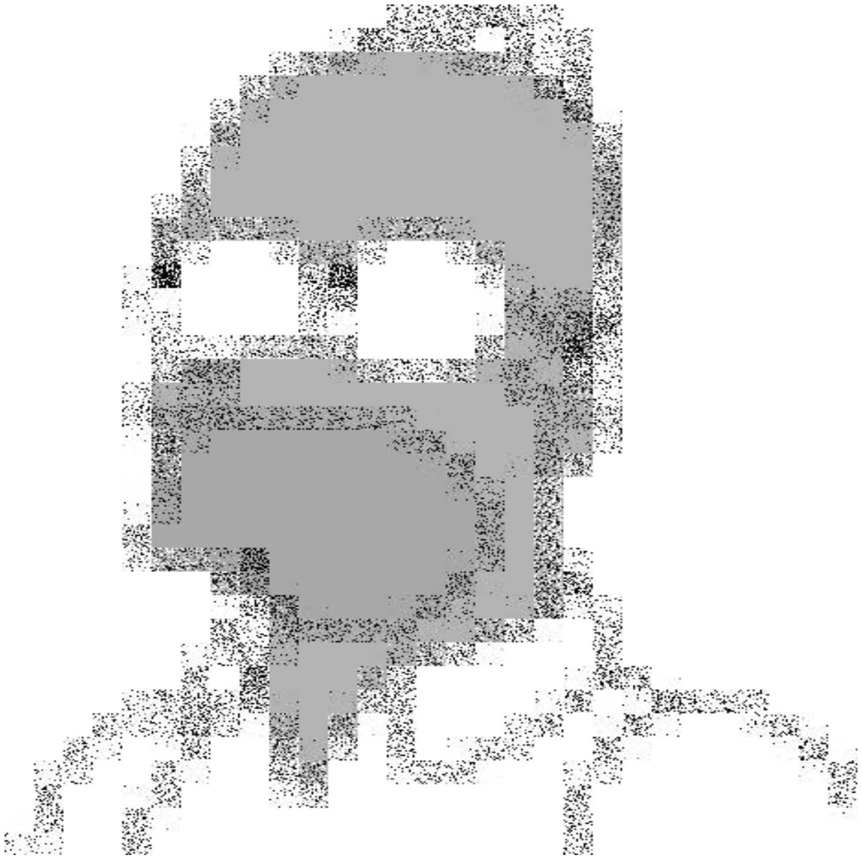


# An example plaintext

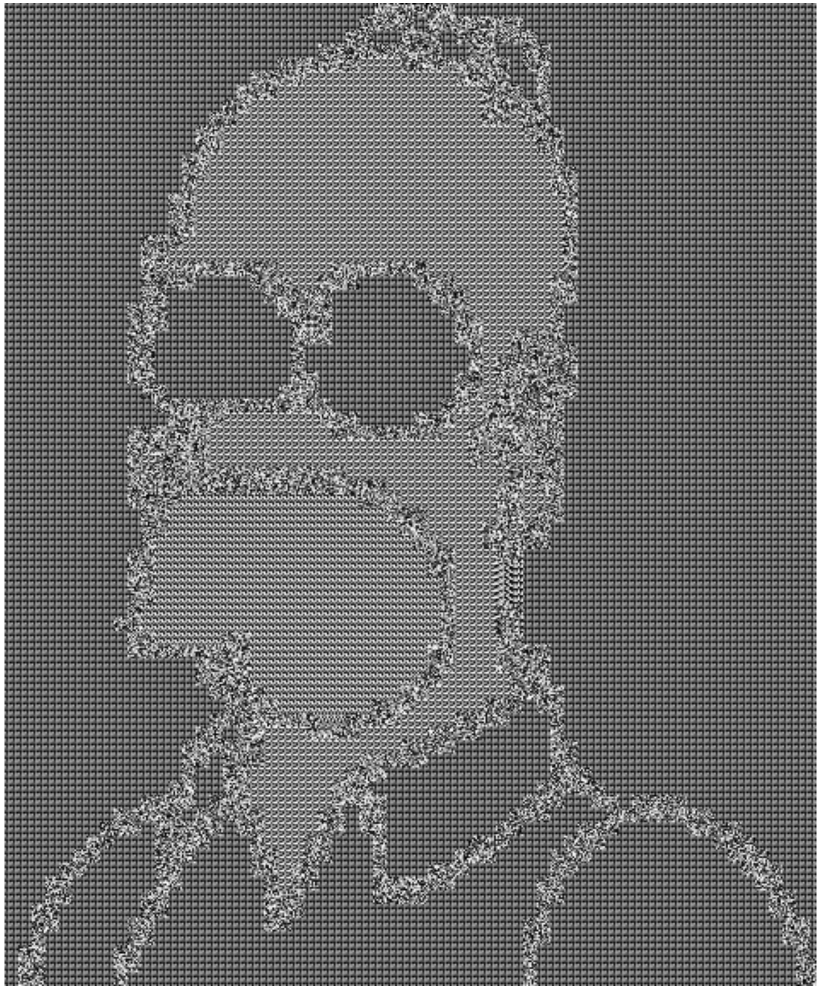


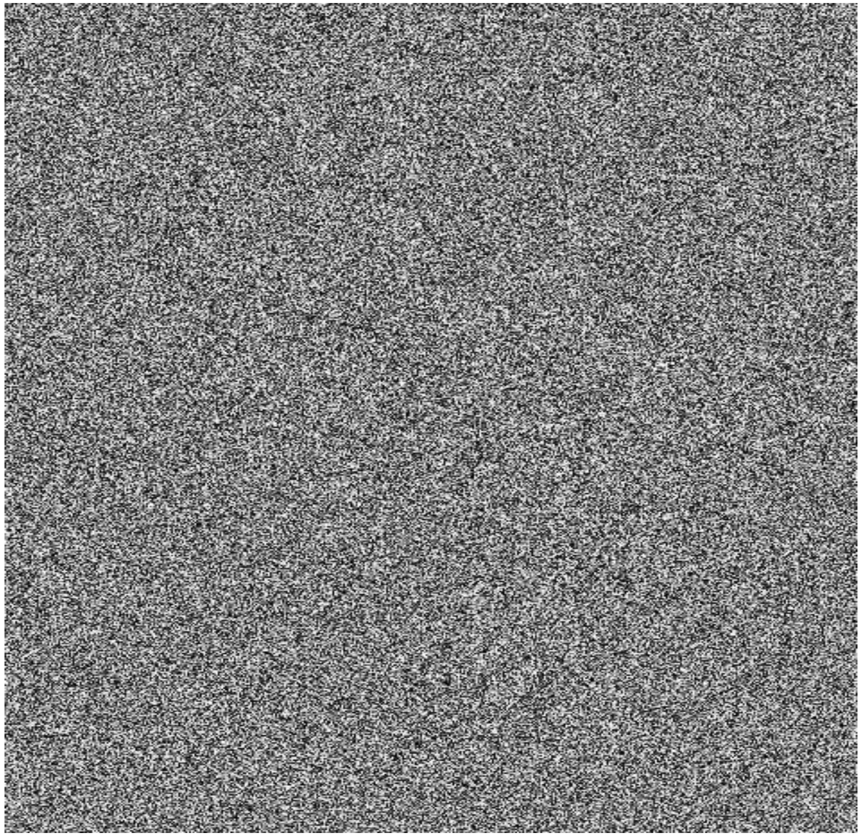
# Encrypted with substitution and transposition cipher



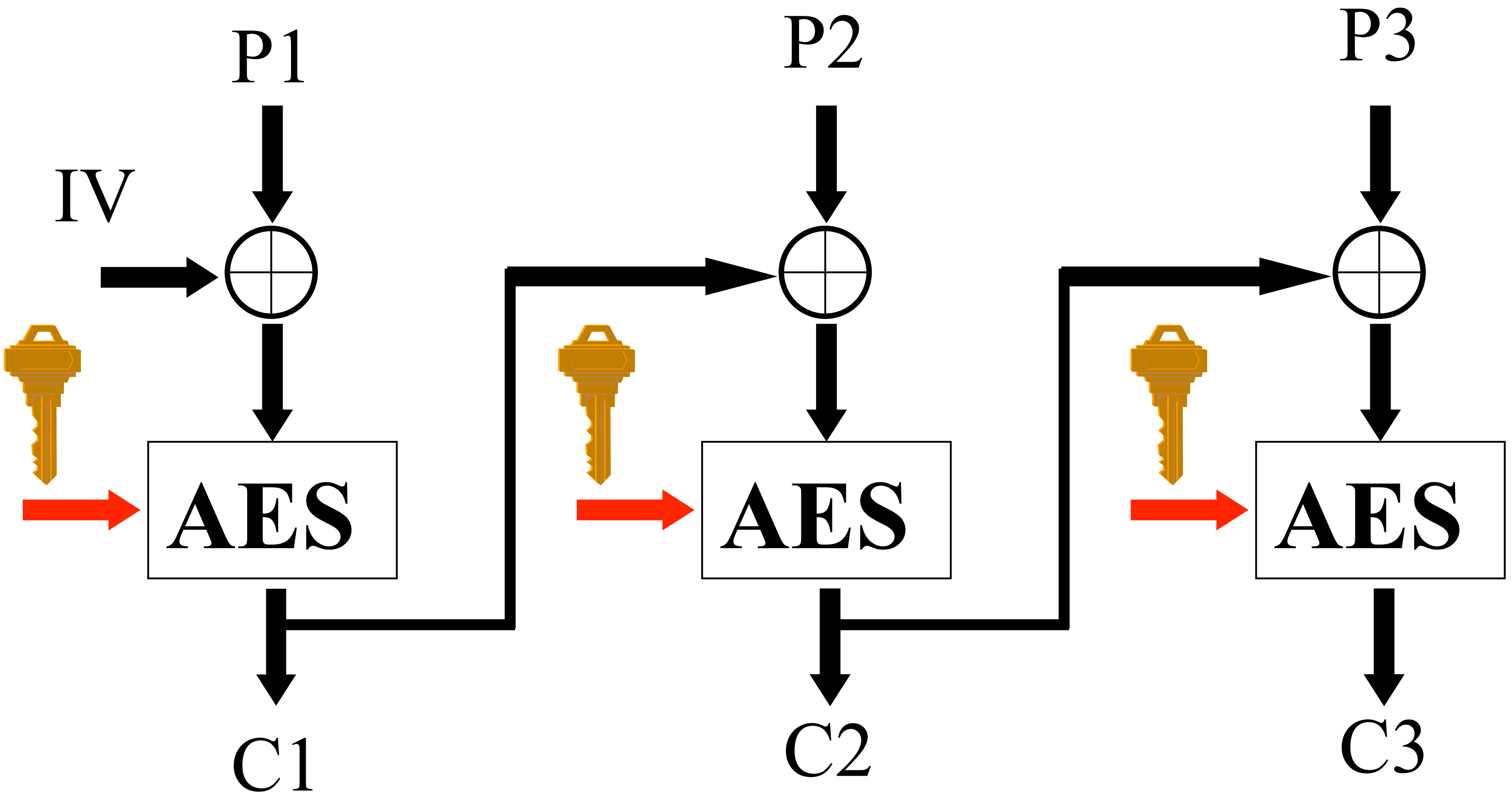


# Encrypted with AES in ECB and CBC mode

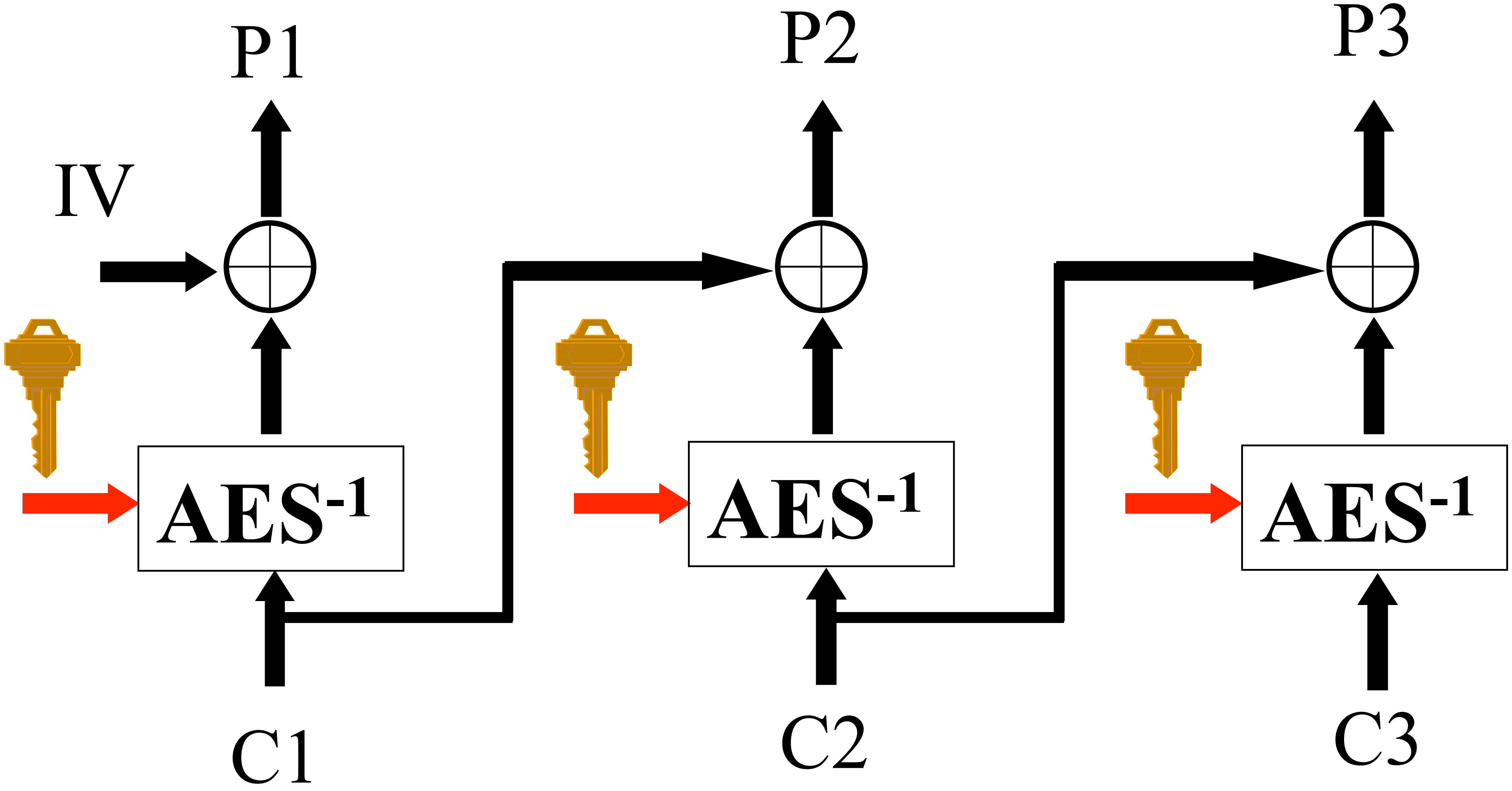




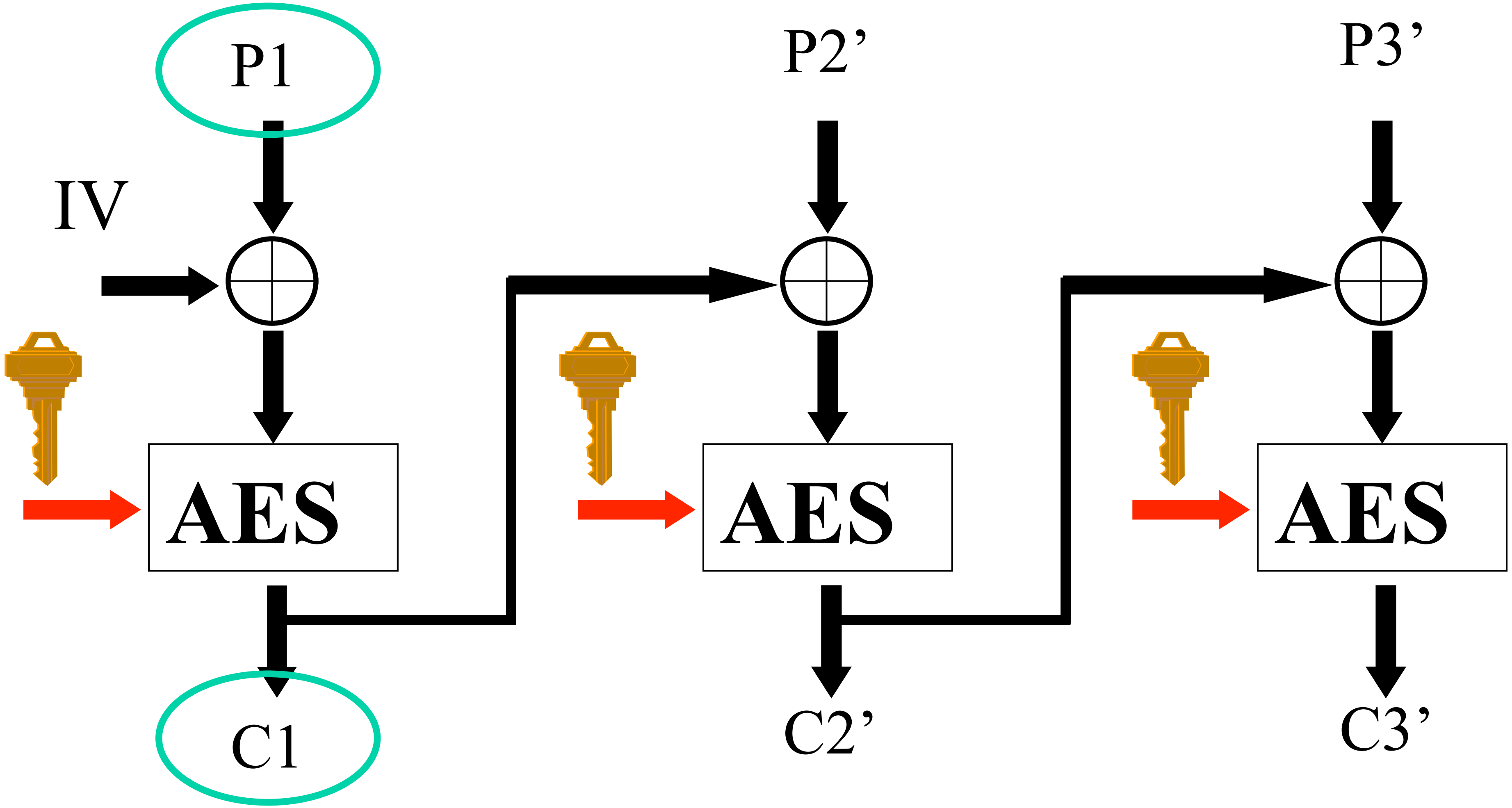
# How to use a block cipher: CBC mode



# CBC mode decryption



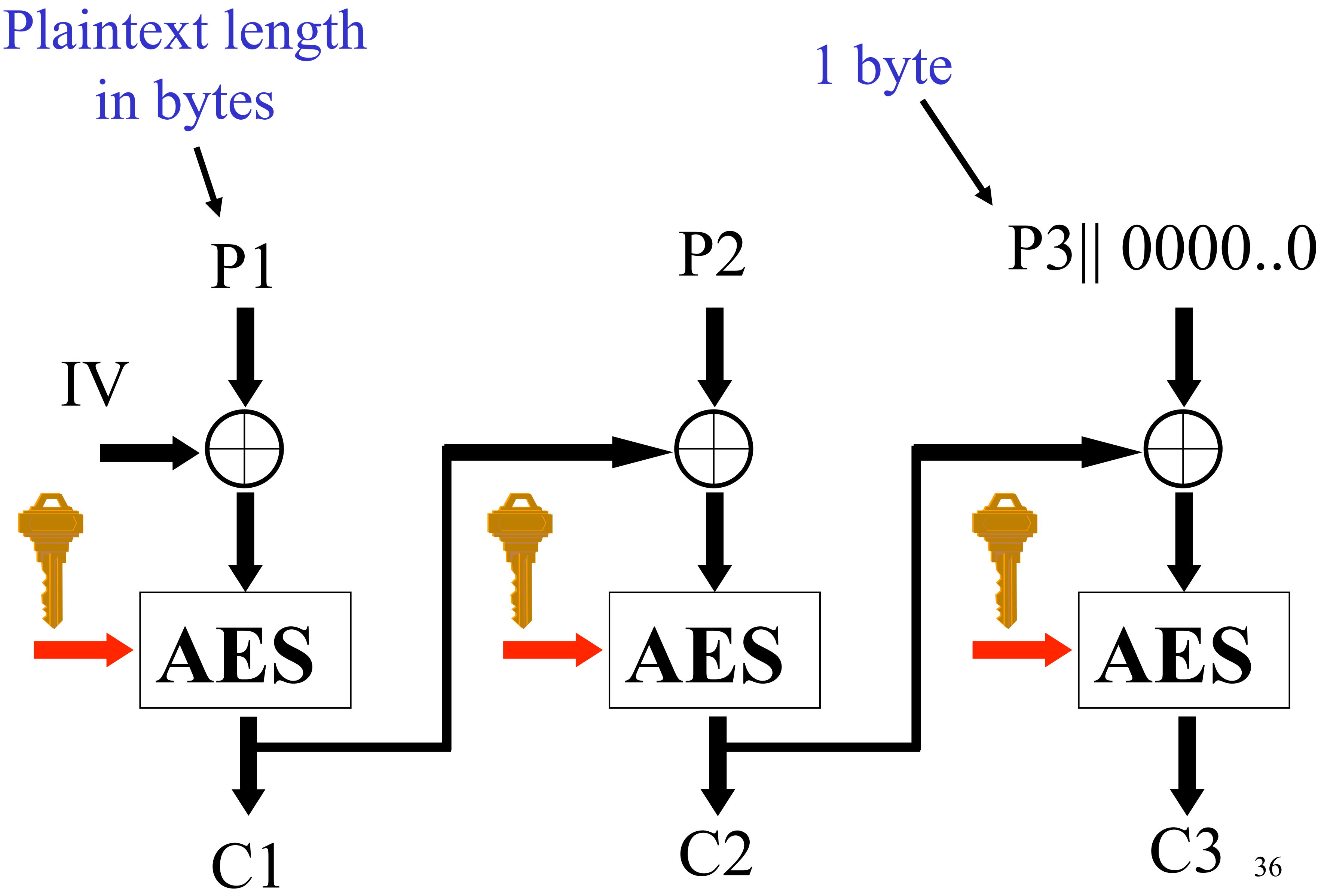
# What if IV is constant?



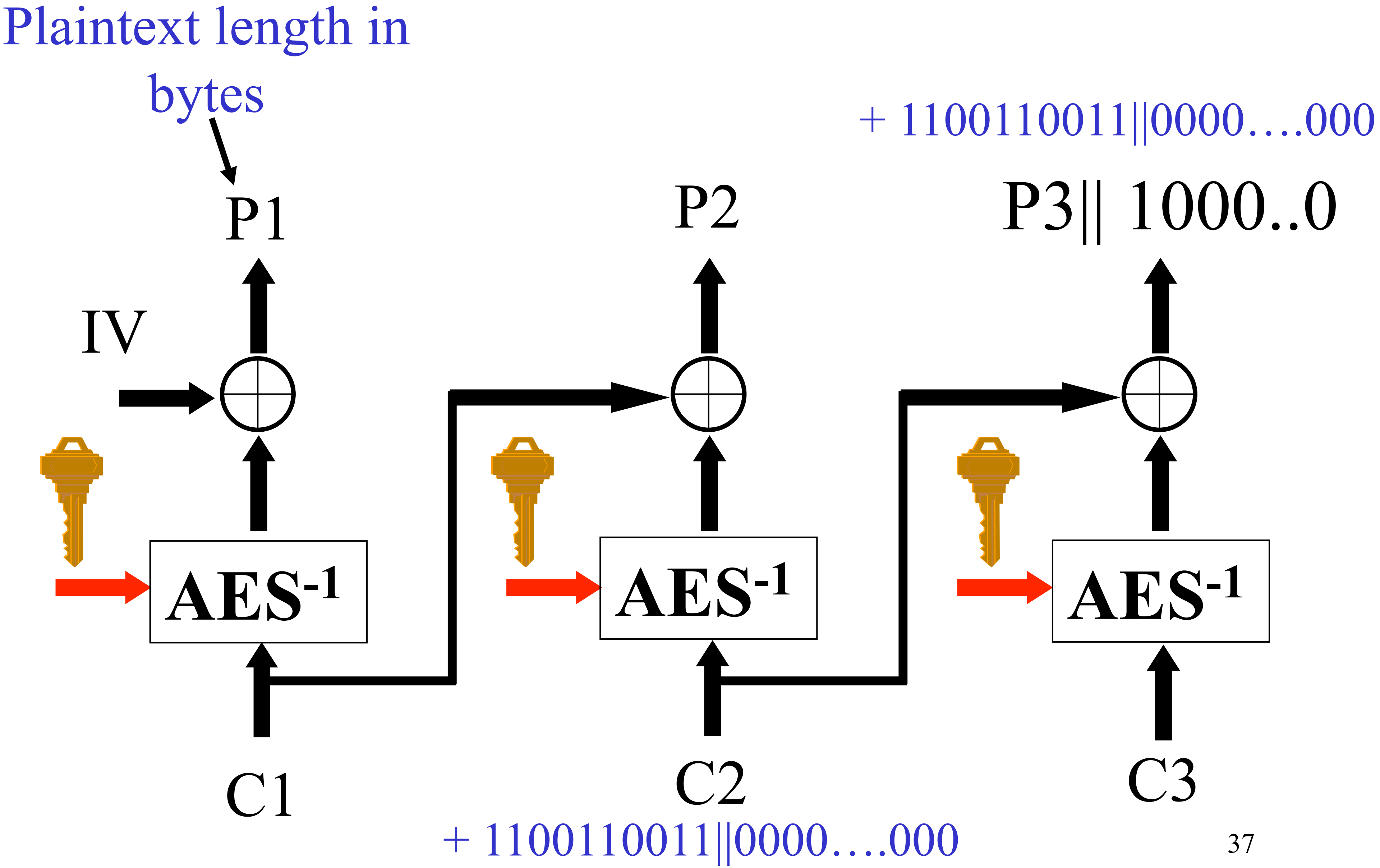
Repetition in P results in repetition in C: ⇒ information leakage

need random and secret IV

# CBC with incomplete plaintext (1)



# CBC with incomplete plaintext (2)



# CBC with incomplete plaintext (3)

Plaintext length in bytes

+ 1100110011||0000….000

P1 P2 P3|| 1000..0

- If the first 10 bits of P3 are equal to 1100110011 then after the modification P3’ will be equal to 0
- The decryption will then produce an error message because the plaintext length field is incorrect
- Conclusion: information on 1 byte of P3 can be obtained using on average 128 chosen ciphertexts
- Protection: random padding or authenticated encryption

# Modes of Operation

- CTR mode allows for pipelining
  - Better area/speed trade-off
- authentication: E-MAC and CMAC
  - E-MAC is CBC-MAC with extra encryption in last block
  - NIST prefers CMAC (was OMAC)
- authenticated encryption:
  - most applications need this primitive (ssh, TLS, IPsec, …)
  - for security against chosen ciphertext this is essential
  - NIST solution: GCM (very fast but lacks robustness)

# Authenticated encryption

Inefficient solution: encrypt then MAC

We can do better

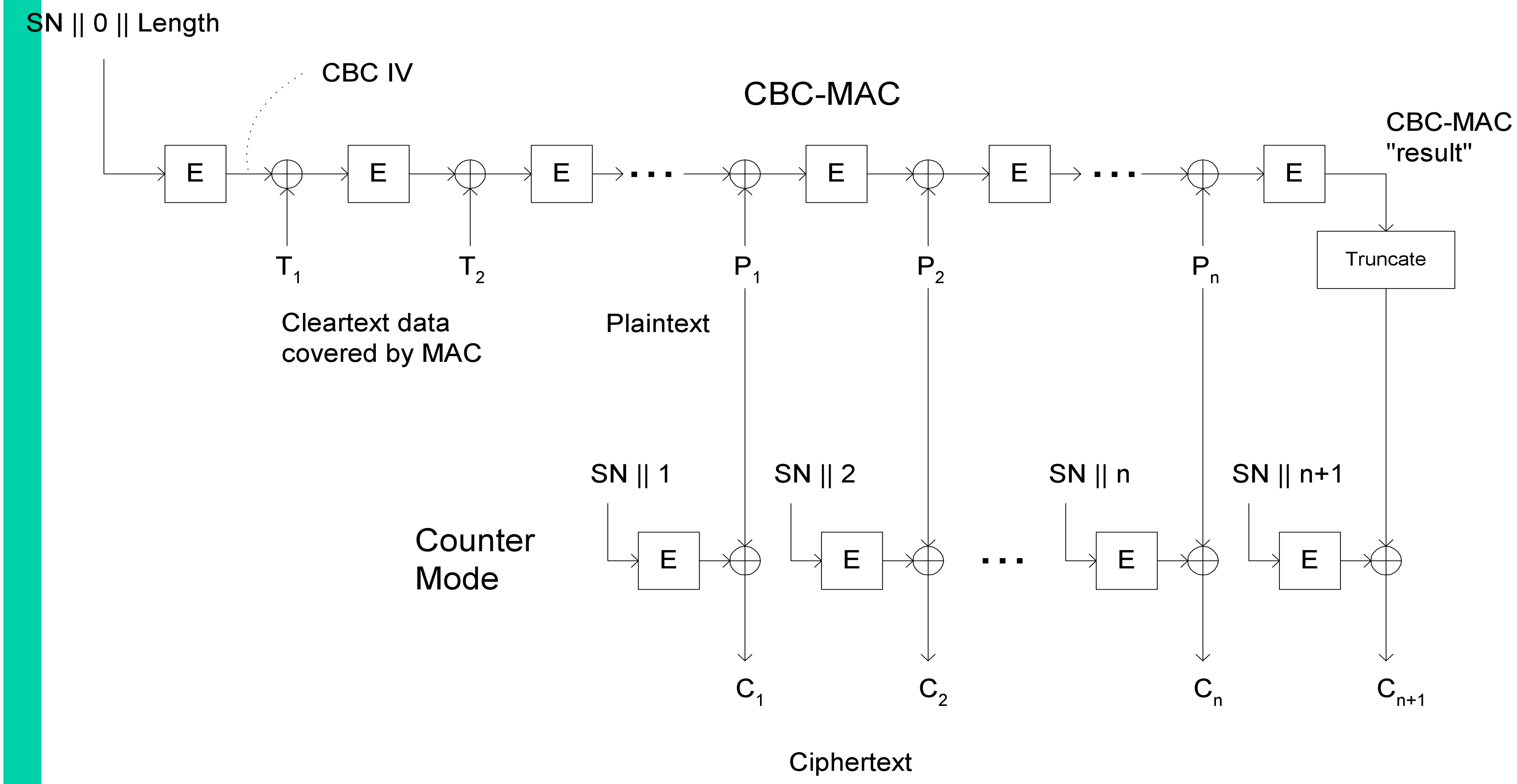
Issues:

- associated data
- parallelizable
- on-line
- patent-free
- provable security

- IAPM
- XECB
- OCB

- CCM
- EAX
- CWC
- GCM

# Example: CCM: CTR + CBC-MAC



SN = packet sequence number (WEP "IV")

# Public-Key Cryptology

- new factorization record in January 2010: 768 bits
- upgrade your RSA-1024 keys by 2010

- increased acceptance of ECC
  - example NSA Suite B in USA
  - Certicom challenge: ECC2K-130: 1 year with 60 KEURO (a large effort is underway)
- progress on pairings leading to more efficient protocols

# Attack on ISO 9796-2 [Coron+’09]

- History:
  - ISO 9796-1 (1991) was broken and withdrawn in 2001
  - ISO 9796-2 was repaired in 2002 after a first attack in 1999

- New forgery attack on 9796-2 that works for very long RSA moduli (2048 bits)
  - any160-bit hash function: 800$ on Amazon cloud
  - the specific EMV variant: 45K$

- Not a practical threat to 750 million EMV cards since the attack requires a large number of chosen texts (600,000)

# Quantum computers?

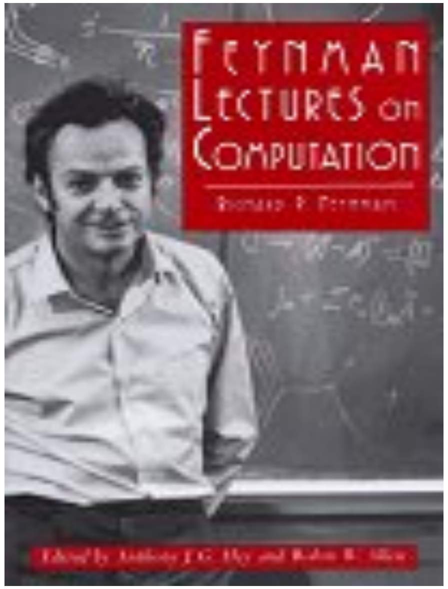
- exponential parallelism

*n* coupled quantum bits

⇓

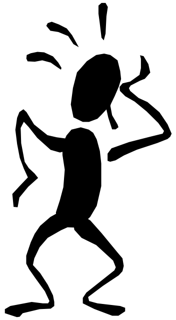
$2^n$ degrees of freedom !

- Shor 1994: perfect for factoring
- But: can a quantum computer be built?


Feynman Lectures on Computation
Richard P. Feynman
Edited by Anthony J.G. Hey and Robin W. Allen

# If a large quantum computer can be built...

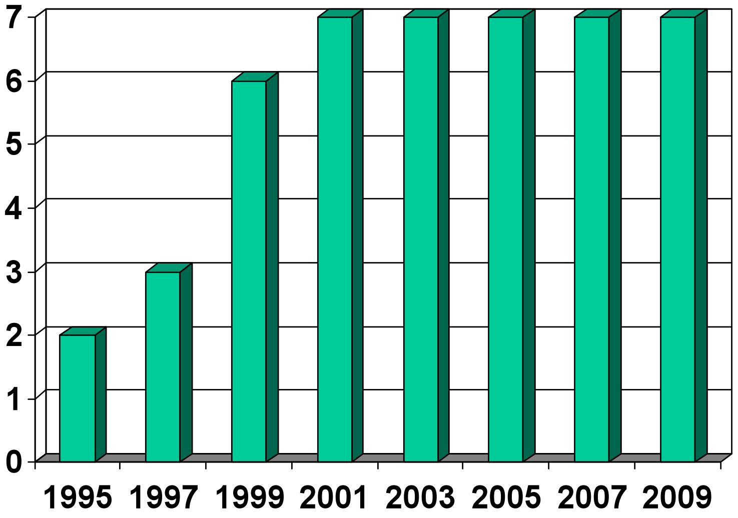
- All schemes based on factoring (such as RSA) will be insecure
- Same for discrete log (ECC)
- Symmetric key sizes: x2
- Hash sizes: x1.5 (?)



- Alternatives: McEliece, NTRU,…
- So far it seems very hard to match performance of current systems while keeping the security level against conventional attacks

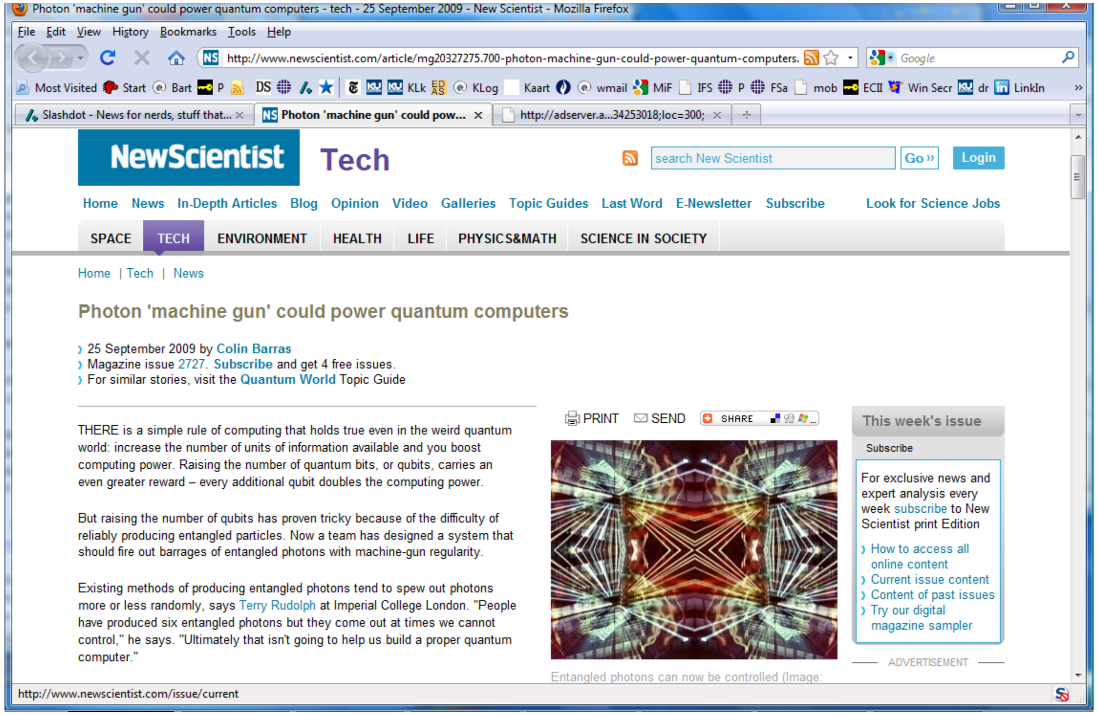
# Quantum computers

- Size of quantum computer does not (yet) matter!



- More important is to keep a few qubits with high reliability for a sufficiently long time (decoherence)

**Photon machine gun, New scientist, Sept. 09**



Photon 'machine gun' could power quantum computers - tech - 25 September 2009 - New Scientist - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.newscientist.com/article/mg20327275.700-photon-machine-gun-could-power-quantum-computers.
Google
Most Visited Start Bart P DS KLk KLog Kaart wmail MiF IFS P FSa mob ECII Win Secr dr LinkIn
Slashdot - News for nerds, stuff that...
Photon 'machine gun' could pow...
http://adserver.a...34253018;loc=300;
NewScientist
Tech
search New Scientist
Go »
Login
Home News In-Depth Articles Blog Opinion Video Galleries Topic Guides Last Word E-Newsletter Subscribe
Look for Science Jobs
SPACE TECH ENVIRONMENT HEALTH LIFE PHYSICS&MATH SCIENCE IN SOCIETY
Home | Tech | News
PRINT
SEND
SHARE
This week's issue
Subscribe
ADVERTISEMENT
http://www.newscientist.com/issue/current

Photon 'machine gun' could power quantum computers

- 25 September 2009 by Colin Barras
- Magazine issue 2727. Subscribe and get 4 free issues.
- For similar stories, visit the Quantum World Topic Guide

THERE is a simple rule of computing that holds true even in the weird quantum world: increase the number of units of information available and you boost computing power. Raising the number of quantum bits, or qubits, carries an even greater reward – every additional qubit doubles the computing power.

But raising the number of qubits has proven tricky because of the difficulty of reliably producing entangled particles. Now a team has designed a system that should fire out barrages of entangled photons with machine-gun regularity.

Existing methods of producing entangled photons tend to spew out photons more or less randomly, says Terry Rudolph at Imperial College London. "People have produced six entangled photons but they come out at times we cannot control," he says. "Ultimately that isn't going to help us build a proper quantum computer."

Entangled photons can now be controlled (Image:

For exclusive news and expert analysis every week subscribe to New Scientist print Edition

- How to access all online content
- Current issue content
- Content of past issues
- Try our digital magazine sampler

# How to encrypt with RSA?

- Assume that the RSA problem is hard
- … so a fortiori we assume that factoring is hard

- How to encrypt with RSA?
  - Hint: ensure that the plaintext is mapped to a random element of [0,n-1] and then apply the RSA Encryption Permutation (RSAEP)

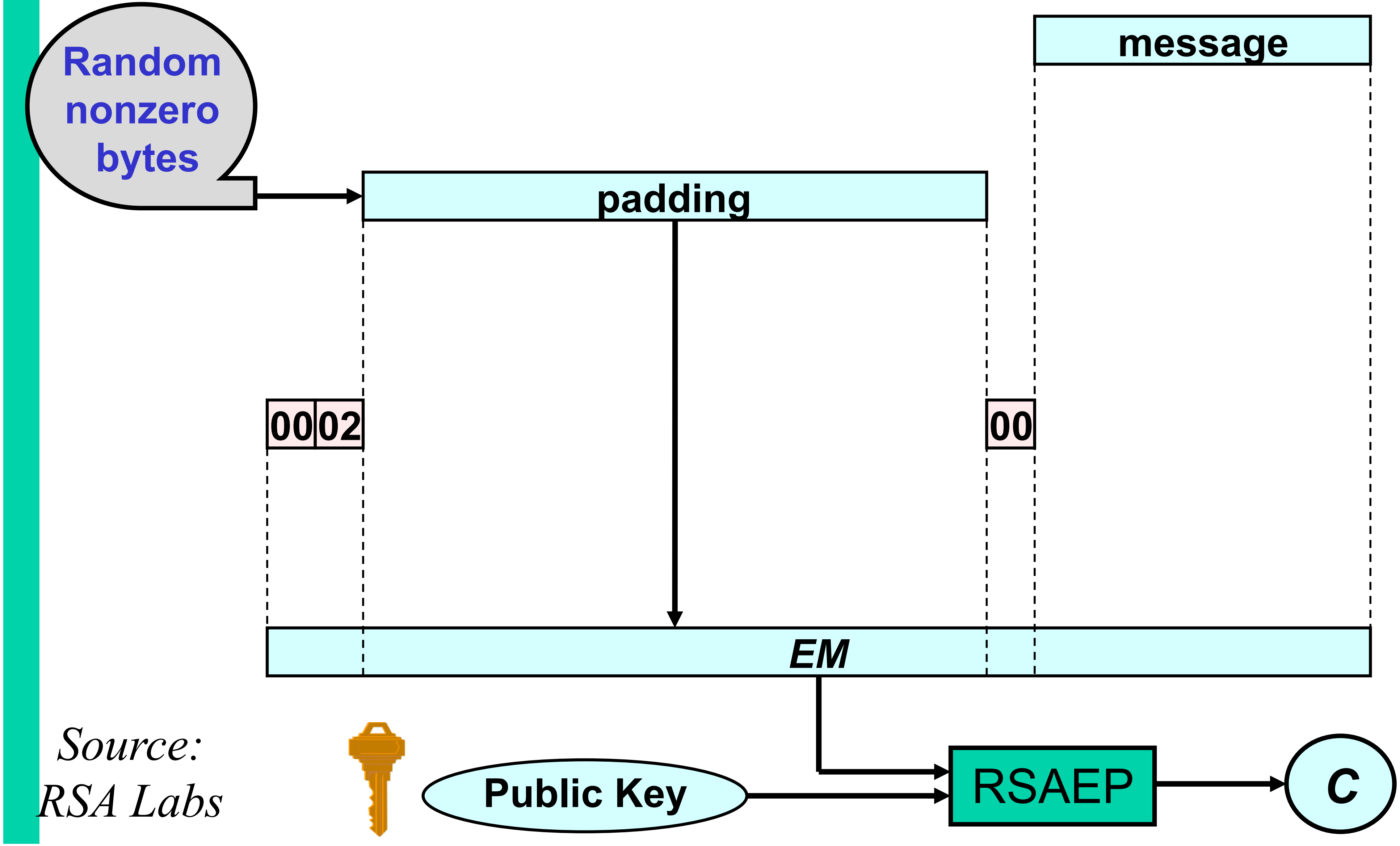
# How (not) to encrypt with RSA?

- Non-hybrid schemes
  - RSA-PKCS-1v1_5 (RSA Laboratories, 1993)
  - RSA-OAEP (Bellare-Rogaway, 1994)
  - RSA-OAEP+ (Shoup, 2000)
  - RSA-SAEP (Johnson et al., 2001)
  - RSA-SAEP+ (Boneh, 2001)
- Hybrid schemes
  - RSA-KEM (Zheng-Seberry, 1992)
    - RSA-KEM-DEM (Shoup, 2001)
    - RSA-REACT (Okamoto-Pointcheval, 2001)
  - RSA-GEM (Coron et al., 2002)

# RSA PKCS-1v1_5

- Introduced in 1993 in PKCS #1 v1.5
- *De facto* standard for RSA encryption and key transport
  - Appears in protocols such as TLS, S/MIME, ...

# RSA-PKCS-1v1_5 Diagram



*Source: RSA Labs*

# RSA-PKCS-1v1_5 Cryptanalysis

- Low-exponent RSA when very long messages are encrypted [Coppersmith+ ‘96/Coron ‘00]
  - large parts of a plaintext is known or similar messages are encrypted with the same public key
- Chosen ciphertext attack [Bleichenbacher ’98]
  - decryption oracle: ciphertext valid or not?
  - 1024-bit modulus: 1 million decryption queries
- These attacks are precluded by fixes in TLS

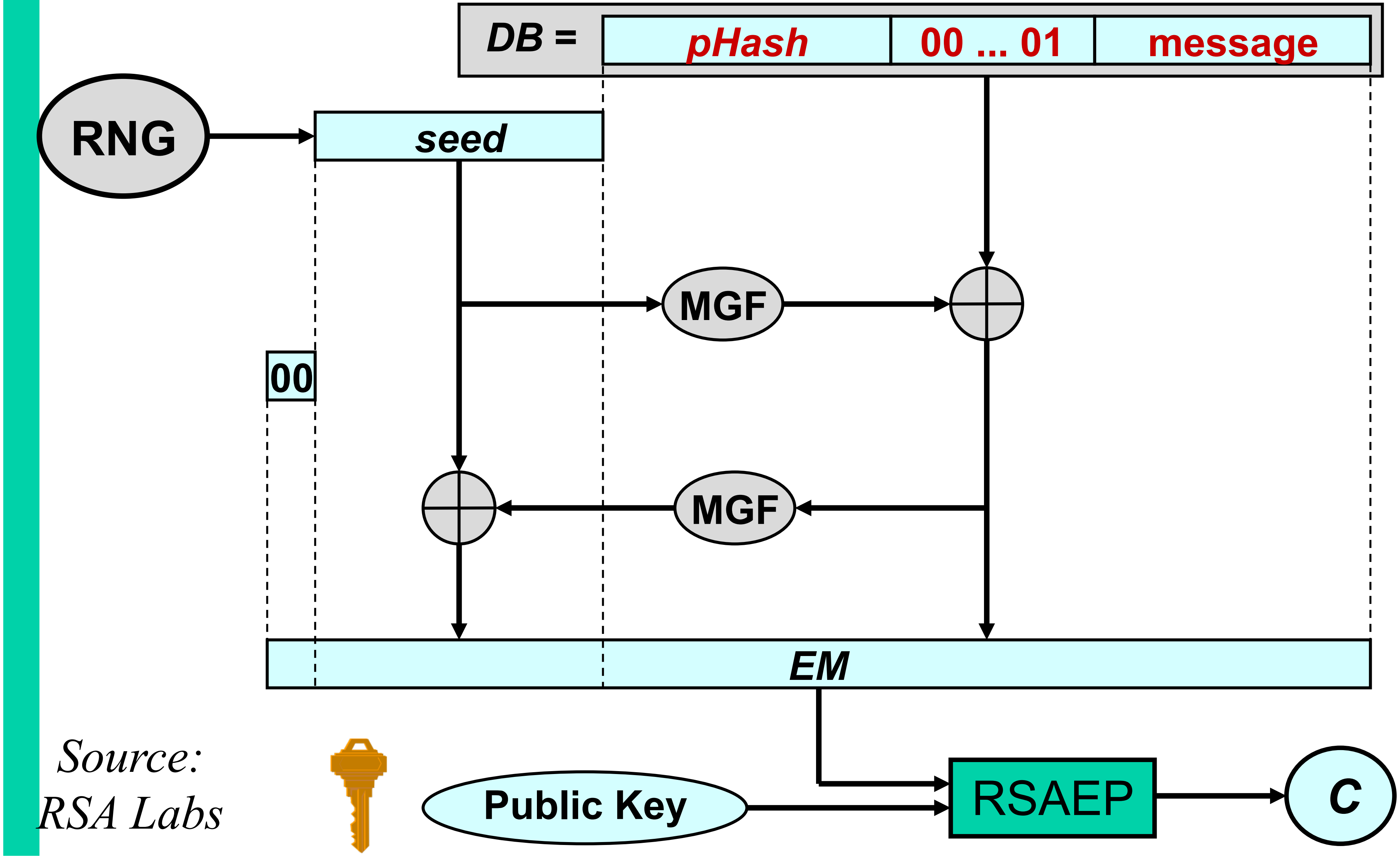
# Bleichenbacher’s attack

- Goal: decrypt c
  - choose random s, $0 < s < n$
  - computer $c' = c\, s^e \bmod n$
  - ask for decryption of c’: m’
  - compute m as $m'/s \bmod n$
- but m’ does not have the right format!
- idea: try many random choices for s:
  - if no error message is received, we know that
    $2B < (m\, s \bmod n) < 3B$
  - with $B = 2^{8(k-2)}$ (k length in bytes of the modulus)

# RSA-OAEP

- designers: Bellare and Rogaway 1993
- enhancements by Johnson and Matyas in 1996 (“encoding parameters”)
- already widely adopted in standards
  - IEEE P1363 draft
  - ANSI X9.44 draft
  - PKCS #1 v2.0  (PKCS #1 v2.1 draft)
  - ISO 18033-2 working draft 2000

# RSA-OAEP Diagram



*Source: RSA Labs*

# RSA OAEP - security

~~*[BR'93] RSA-OAEP is IND-CCA2 secure under RSA assumption in ROM*~~

*Shoup '00: the proof is wrong*

*[FOPS 01] RSA-OAEP is IND-CCA2 secure under partial domain one-wayness RSA assumption in ROM*

*for RSA: partial domain one-wayness ⇔ one-wayness*

*Reduction is very weak*

*ROM assumption is questionable*

# RSA OAEP - security

- Improved chosen ciphertext attack [Manger, Crypto ‘01]

- requires a few thousand queries (1.1 $\log_2 n$)
- opponent needs oracle that tells whether there is an error in the integer-to-byte conversion or in the OAEP decoding

- overall conclusion: RSA Inc. is no longer recommending the use of RSA-OAEP

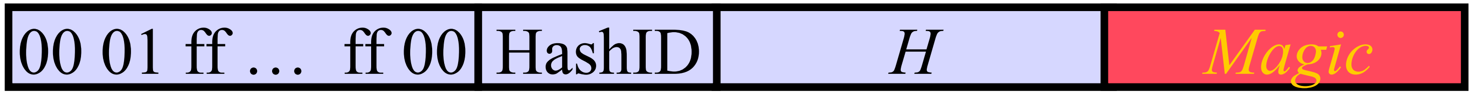
if it’s provable secure, it probably isn’t

# How to encrypt with RSA

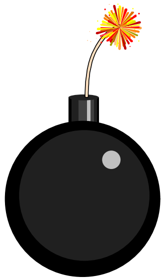
- RSA-KEM
  - encrypt 2 session keys with  RSA
  - encrypt and MAC data with these 2 keys
- Recommended in NESSIE report (http://www.cryptonessie.org) and to be included in ISO 18033
- Similar problems for signatures:
  ISO 9796-1 broken, PKCS#1 v1.0 questionable

# Attack on PKCS #1 v1.5 implementations (1)

[Bleichenbacher06]



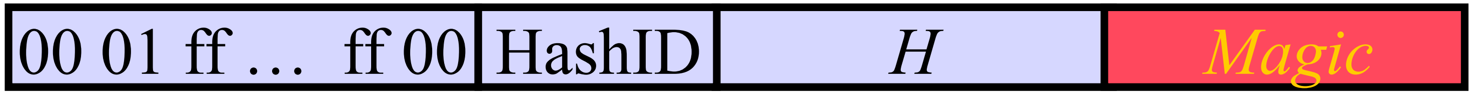
00 01 ff … ff 00 | HashID | *H* | *Magic*

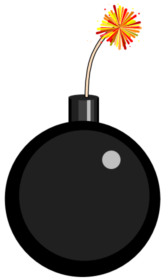


- Consider RSA with public exponent 3
- For any hash value H, it is easy to compute a string "Magic" such that the above string is a perfect cube of 3072 bits
- Consequence:
  - One can sign any message (H) without knowing the private key
  - This signature works for any public key that is longer than 3072 bits
- Vulnerable: OpenSSL, Mozilla NSS, GnuTLS

# Attack on PKCS #1 v1.5 implementations (2)

[Bleichenbacher06]





- Fix
  - Write proper verification code (but the signer cannot know which code the verifier will use)
  - Use a public exponent that is at least 32 bits
  - Upgrade – finally – to RSA-PSS

# Cryptographic algorithm selection

- Standards?
- Public domain versus proprietary
- Upgrades

# Cryptographic standards

- Algorithms historically sensitive (e.g., GSM)
- Choices with little technical motivation (e.g., RC2 and MD2)
- Little or no coordination effort (even within IETF)
- Technically difficult

A.S. Tanenbaum: “The nice thing about standards is there's so many to choose from”

# Major Standardization Bodies in Cryptography

- International
  - ISO and ISO/IEC International Organization for Standardi
  - ITU: International Telecommunications Union
  - IETF: Internet Engineering Task Force
  - IEEE: Institute of Electrical and Electronic Engineers
- National
  - ANSI: American National Standards Institute
  - NIST: National Institute of Standards and Technology
- European
  - CEN: Comité Européen de Normalisation
  - ETSI: European Telecommunications Standards Institute
- Industry
  - PKCS, SECG
  - W3C, OASIS, Liberty Alliance, Wi-Fi Alliance, BioAPI, WS-Security, TCG
  - GP, PC/SC, Open Card Framework, Multos

# Independent evaluation efforts

- NIST (US) (1997-2001): block cipher AES for FIPS 197 (http://csrc.nist.gov/CryptoToolkit/aes/)
- CRYPTREC (Japan) (2000-2003): cryptographic algorithms and protocols for government use in Japan (http://www.ipa.go.jp/security)
- EU-funded IST-NESSIE Project (2000-2003): new cryptographic primitives based on an open evaluation procedure (http://www.cryptonessie.org)
- ECRYPT eSTREAM (2004-2007): stream cipher competition

# Proprietary/secret algorithms

- No “free” public evaluations
- Risk of snake oil
- Cost of (re)-evaluation very high
- No economy of scale in implementations
- Reverse engineering

- Fewer problems with rumors and “New York Times” attacks
- Extra reaction time if problems
- Fewer problems with implementation attacks
- Can use crypto for IPR and licensing

# Many insecure algorithms in use

- Do it yourself (snake oil)
- Export controls
- Increased computational power for attacks (64-bit keys are no longer adequate)
- Cryptanalysis progress - including errors in proofs
- Upgrading is often too hard by design
  - cost issue
  - backward compatibility
  - version roll-back attacks

# Upgrade problem

- GSM: A5/3 takes a long time
- Bluetooth: E0 hardwired
- TCG: chip with fixed algorithms
- MD5 and SHA-1 widely used

- Negotiable algorithms in SSH, TLS, IPsec,…
- But even then these protocols have problems getting rid of MD5/SHA-1

Make sure that you do not use the same key with a weak and a strong variant (e.g. GSM A5/2 and A5/3)

# And the good news

- Many secure and free solutions available today: AES, RSA,…
- With some reasonable confidence in secure
- Cost of strong crypto decreasing except for “niche applications” (ambient intelligence)

In spite of all the problems, cryptography is certainly not the weakest link in our security chain

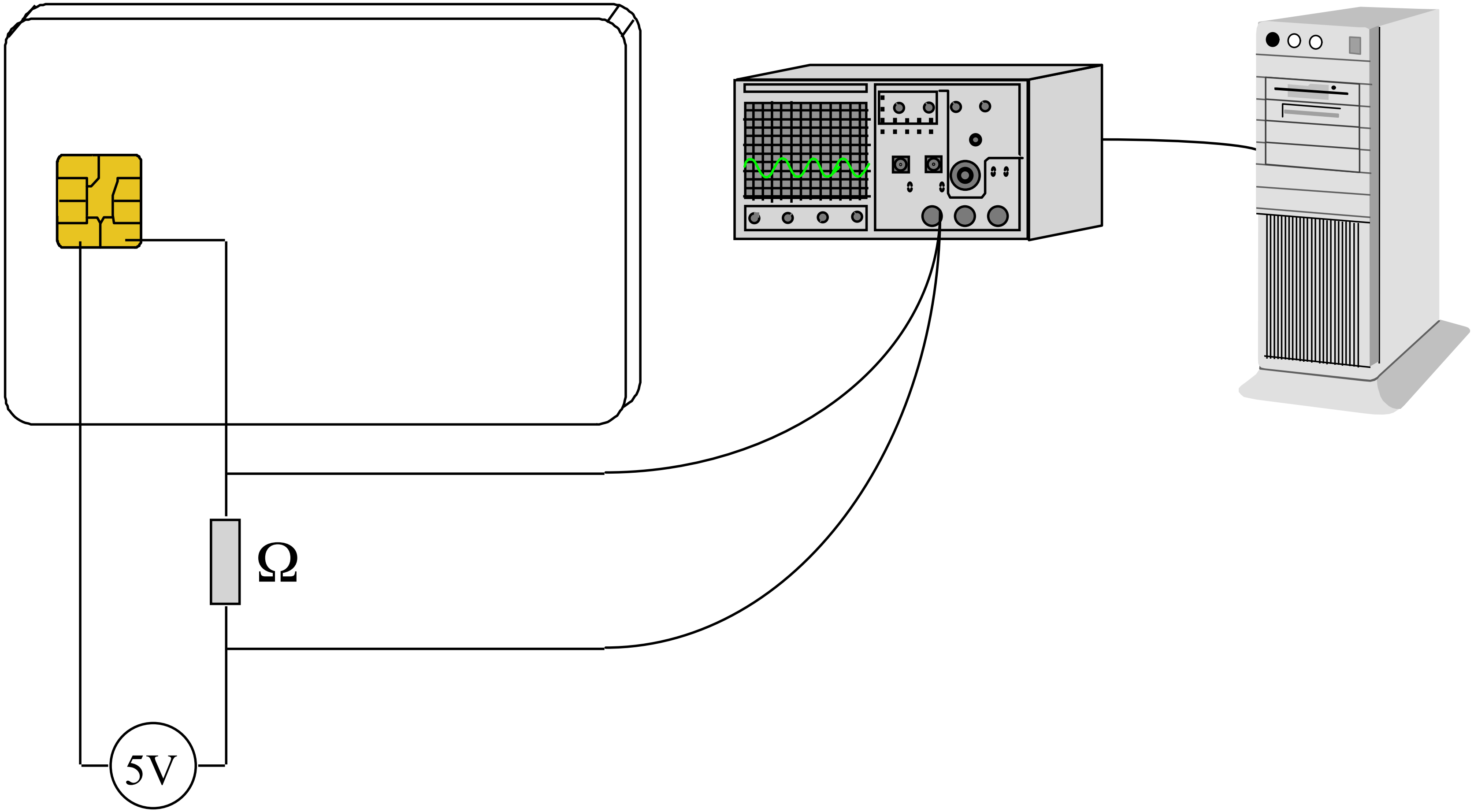
# What to use (generic solutions)

- Authenticated encryption mode (OCB, CWC, CCM, or even GCM) with 3-key 3-DES or AES
- Hash functions: RIPEMD-160, SHA-256, SHA-512 or Whirlpool
- Public key encryption: RSA-KEM or ECIES
- Digital signatures: RSA-PSS or ECDSA
- Protocols: TLS, SSH, IKE(v2)

# Secure implementations of cryptography

- Error messages and APIs (cf. supra)
- Side channels
  - Timing attacks
  - Power attacks
  - Acoustic attacks
  - Electromagnetic attacks
- Fault attacks

# Power analysis tools for smart cards



# Software: constant time is crucial

- PIN verification
- Square and multiply for RSA
- Variable rotations in RC5 and RC6
- Swaps in RC4
- Problems with cache misses in ciphers with S-boxes such as DES and AES

# PIN verification

```
input (PIN_U[0..k-1],PIN[0..k-1])
i=0;
while (i < k) do {
  if (PIN_U[i] != PIN[i]) return (0);
  i = i+1;
  }
return(1);
```

Problem?

# Timing attack on RSA

- “square and multiply” algorithm
- exponent bits scanned from MSB to LSB (left to right)

```
Let k = bitsize of d (say 1024)

Let s = m

For i = k-2 down to 0

  Let s = s*s mod n (SQUARE)

  If (bit i of d) is 1  then
    Let s = s*m mod n (MULTIPLY)
  End if

End for
```

| **Example :** | **$s = m^9 = m^{1001b}$** |
|---|---|
| **init (MSB 1)** | **$s = m$** |
| **round 2 (bit 0)** | **$s = m^2$** |
| **round 1 (bit 0)** | **$s = (m^2)^2 = m^4$** |
| **round 0 (bit 1)** | **$s = (m^4)^2 * m = m^9$** |

# Cache attack on crypto algorithms with S-boxes (DES, AES,…)

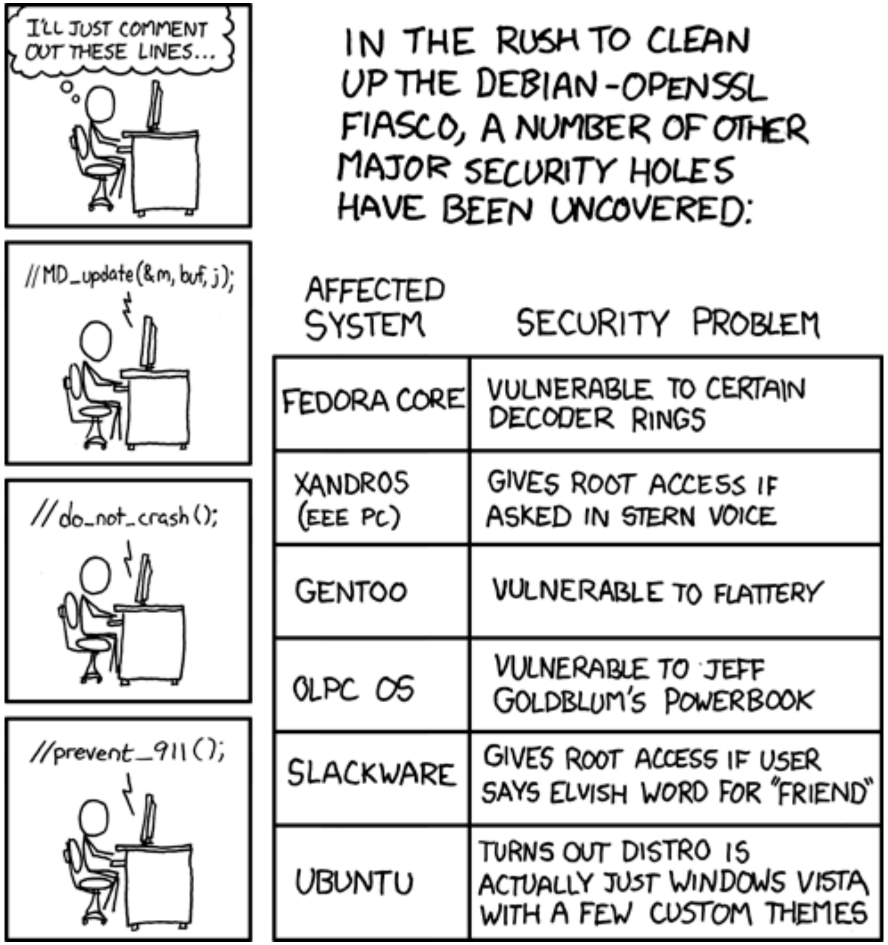
- Cache misses influence execution time
- Uses HyperThreading to monitor the encrypting process in real time and observe its use of shared resources.

- [Tsunoo-Saito-Suzaki-Shigeri-Miyauchi 03] Cryptanalysis of DES implemented on computers with cache, CHES 2003, LNCS 2779, 62-76, 2003
- [Osvik-Shamir-Tromer 05] Cache Attacks and Countermeasures: the Case of AES, RSA CT 2006
- [Bernstein 05] Cache-timing attacks on AES

# Implementation attacks (13 May ’08)

## Debian-OpenSSL incident

- Weak key generation:
  only 32K keys
  - easy to generate all private keys
  - collisions
- Between 13-17 May:
  280 bad keys out of 40K
  (0.6%)
- Revocation problematic



IN THE RUSH TO CLEAN UP THE DEBIAN-OPENSSL FIASCO, A NUMBER OF OTHER MAJOR SECURITY HOLES HAVE BEEN UNCOVERED:

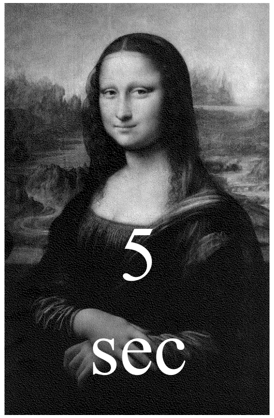
| AFFECTED SYSTEM | SECURITY PROBLEM |
|---|---|
| FEDORA CORE | VULNERABLE TO CERTAIN DECODER RINGS |
| XANDROS (EEE PC) | GIVES ROOT ACCESS IF ASKED IN STERN VOICE |
| GENTOO | VULNERABLE TO FLATTERY |
| OLPC OS | VULNERABLE TO JEFF GOLDBLUM'S POWERBOOK |
| SLACKWARE | GIVES ROOT ACCESS IF USER SAYS ELVISH WORD FOR "FRIEND" |
| UBUNTU | TURNS OUT DISTRO IS ACTUALLY JUST WINDOWS VISTA WITH A FEW CUSTOM THEMES |

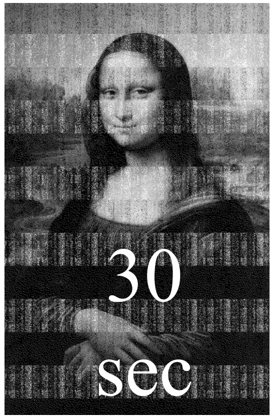
# Implementation attacks
# cold boot attack

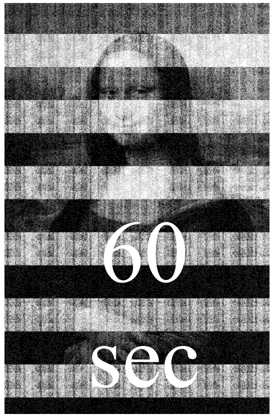
- Why break cryptography? Go for the key, stupid!
- Data reminence in DRAMs

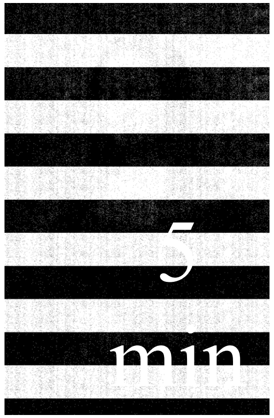
  Lest We Remember: Cold Boot Attacks on Encryption Keys [Halderman-Schoen-Heninger-Clarkson-Paul- Calandrino-Feldman- Appelbaum-Felten’08]

  - Boot from USB device and dump RAM image
  - Works for AES, RSA,…
  - Products: BitLocker, FileVault, TrueCrypt, dm-crypt, loop-AES









# Implementation attacks cold boot attack (2)

- Countermeasures
  - Overwrite keys in memory
  - Shut down rather than sleep/hibernate
  - Limit boot options (network, USB)
  - resilient exposure cryptography (AONT)
  - physical protection of DRAM
  - encrypt in the disk controller
  - new architecture
- Ineffective: trusted computing as implemented today

# Some crypto libraries

- **OpenSSL: http://www.openssl.org/**
- **Cryptlib: http://www.cs.auckland.ac.nz/~pgut001/cryptlib/**
- **SSLeay: http://www2.psy.uq.edu.au/~ftp/Crypto/**
- **IAIK Java: http://jce.iaik.tugraz.at/products/index.php**
- **COSIC crypto library (contact B. Preneel)**
- **See also http://www.ssh.fi/support/cryptography/online_resources/practical.html**

# Novel applications of cryptography

- Whitebox crypto
- SPAM fighting

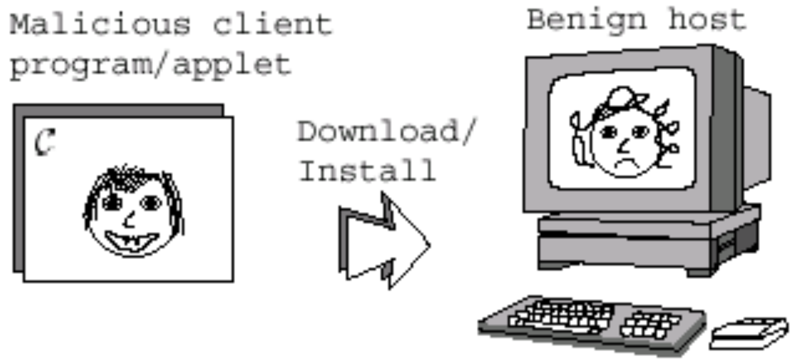
# Protection of software against whitebox attacks

- Software
  - Confidential information
  - Secret keys
  - Proprietary code
- Software and content distribution
- White-box setting
  - Complete accesss to implementation
  - Decompilation, reverse engineering, …

# Protection of software against whitebox attacks

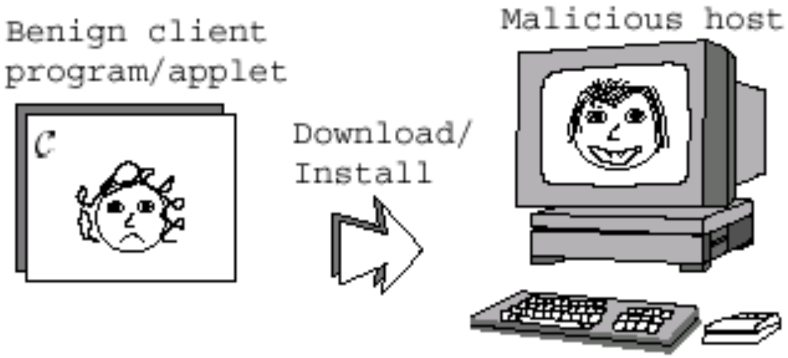
- “sandboxing”

  protect host against malware



- malicious hosts

  protect software against malicious hosts



# Techniques

- White-box cryptography
    - Extra input and output coding of encryption
- Code obfuscation
    - Obfuscate code and program flow
- Other techniques:
    - Integrity checks + error detection

      → Tamper resistant software (TRS)
    - Code encryption + 'on-the-fly' decryption

# White Box Cryptography

- Mathematical technique to hide keys in code

$$E'_K = \boxed{G \circ \boxed{E_K} \circ F^{-1}}$$

- *With:*
  - *EK* : encryption function, key *K*
  - *F* : arbitrary input coding
  - *G* : arbitrary output coding

# Pro and Cons

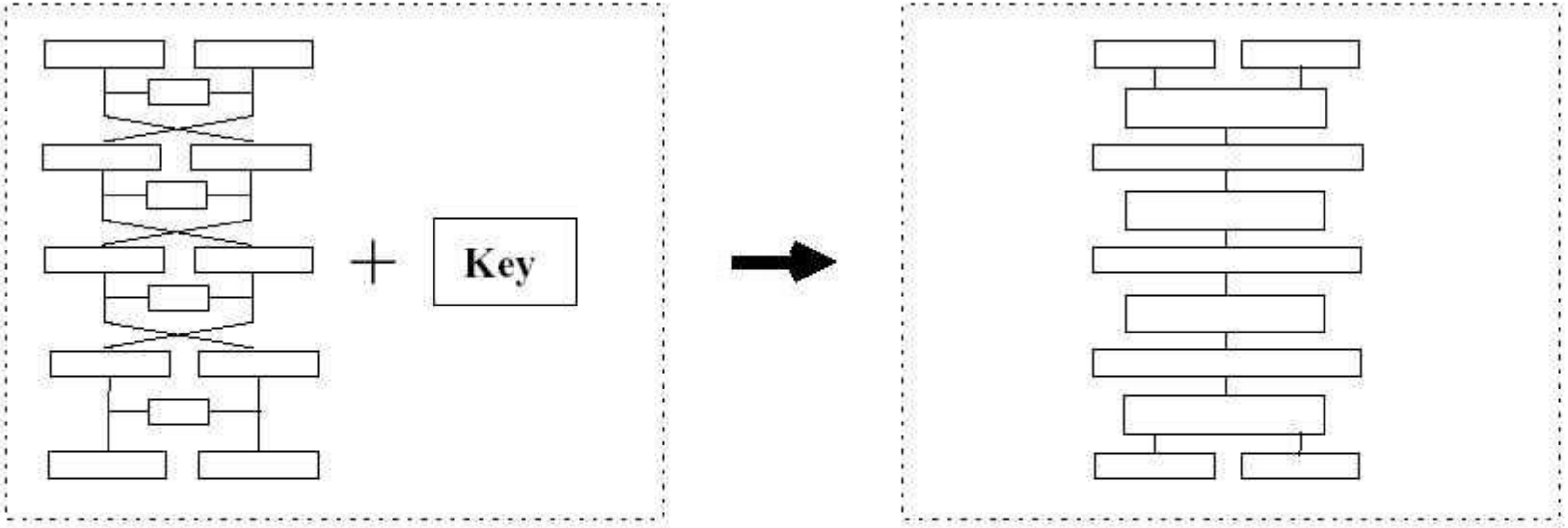
- Unique object code
  - Choose $F$ and $G$
  - Integrate key
- Protect key
  - No function that computes $E_K$ for an arbitrary key $K$
- Flexible
- Fast updates

- Increased memory
  - Tables for input and output coding and for function
- Increased execution time
- Security: very strong attack model
  - Trade-off with performance
- Fast key update open problem

# Example

- DES
  - 16-round Feistel
  - 8 S-boxes
  - 56-bit key

- White-box DES
  - General structure
  - 12 "T-boxes"
  - Key built in code



# The SPAM problem: it is about economics, stupid

- list of $10^7$-$10^8$ “good” names
- cost per message: ~$10^{-5}$ €; total cost 100-1000 €
- hit ratio: $10^{-6}$ to $10^{-4}$: 10-10000 responses

- Cost to society
  - Ruining e-mail as communication tool
  - Time and attention
  - ISP fees
  - Storage and bandwidth

# AND…

"The right to be left alone - the most comprehensive of rights, and the right most valued by civilized men."

**- Supreme Court Justice Louis Brandeis**

# Fighting SPAM

- Filtering
- Make sender pay
- Ephemeral email addresses
- Data/Sender Authentication

# Fighting SPAM (2)

- Filtering
  - Everyone: text-based
  - Brightmail: decoys; rules updates
  - Microsoft Research: (seeded) trainable filters
  - SpamCloud: collaborative filtering
  - SpamCop, Osirusoft, etc: IP addresses, proxies, …
- Make Sender Pay
  - Computation (CPU and/or memory)
  - Human attention
  - Cash, bonds, stamps (PennyBlack)

# Fighting SPAM (3)

- Ephemeral e-mail addresses
  - E.g. SPA: Single Purpose Addresses
- Data/Sender authentication
  - Sign all emails
  - Sender Permitted From (SPF): whitelist mail senders
  - Sign domain names (Yahoo's DomainKeys)
  - Authenticated mail: AMTP (TLS)

Often bypass for friends on whitelist

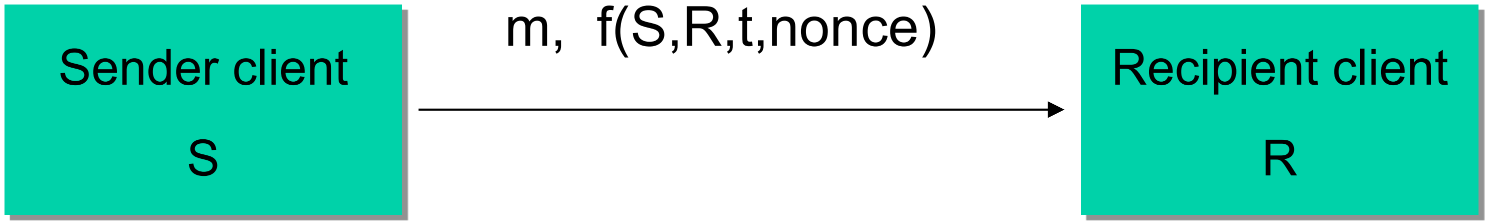
# Filtering: limitations

- Still high cost if too late in the chain
- Spammers generate more sophisticated emails…
  - "Daphnia blue-crested fish cattle, darkorange fountain moss, beaverwood educating, eyeblinking advancing, dulltuned amazons...."
  - FWD: Many On Stocks. Vali/u/m + V1codin+ ; V| @GRa + /Xanax/ ; Pnter.m.in ? Som|a| muKPs

# Computational Approach

- If I don’t know the sender:
  - Prove sender spent 10 seconds CPU time,
  - just for me, and just for this message
- Checking proof by receiver:
  - automatically in the background
  - very efficient
- All unsolicited mail treated equally

# Point-to-Point Architecture



(Ideal Message Flow)

- Single-pass “send-and-forget”
- Can augment with helper to handle slow machines
- Can add post office / pricing authority to handle money payments
- Time mostly used as nonce for avoiding replays (cache tags, discard duplicates; time controls size of cache)

# Economics

- 10 seconds CPU cost a few hundreds of a cent
- (80,000 s/day) / (10s/message) = 8,000 msgs/day
- Hotmail's billion daily spams:
  - 125,000 CPUs
  - Up front capital cost just for hardware: $150 million
- The spammers can't afford it.

# Cryptographic Puzzles

- Hard to compute; f(S,R,t,nonce) can’t be amortized
  - lots of work for the sender
- Easy to check “z = f(S,R,t,nonce)”
  - little work for receiver
- Parameterized to scale with Moore's Law
  - easy to exponentially increase computational cost, while barely increasing checking cost
- Can be based on (carefully) weakened signature schemes, hash collisions
- Can arrange a “shortcut” for post office

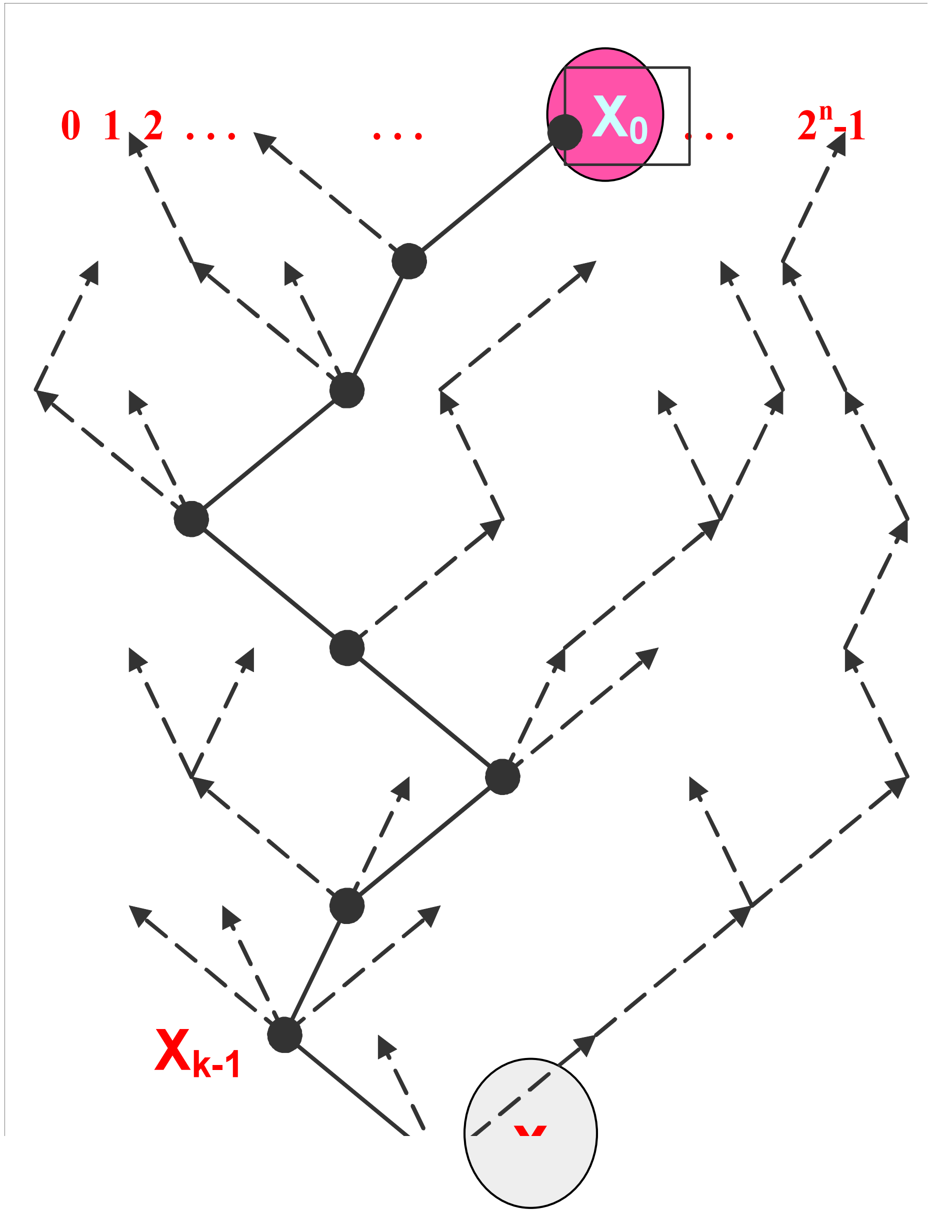
# Idea: replace CPU by memory

- CPU speeds vary widely across machines, but memory latencies vary much less (20-100 vs 2-6)
  - 33 MHz PDA vs. 3 GHz PC

- design a puzzle leading to a large number of cache misses
- Concrete schemes: [ABMW02] and [DGN03]

# Easy Functions [ABMW02]

- f: n bits to n bits, easy
- Given $x_k \in \text{range}(f^{(k)})$, find a pre-image with certain properties
- Hope: best solved by building table for $f^{-1}$ and working back from $x_k$
- Choose n=22 so $f^{-1}$ fits in small memory, but not in cache
- Optimism: $x_k$ is root of tree of expected size $k^2$



# Social Issues

- Who chooses f?
  - One global f? Who sets the price?
  - Autonomously chosen f's?
- How is f distributed (ultimately)?
  - Global f built into all mail clients? (1-pass)
  - Directory? Query-Response? (3-pass)

# Technical Issues

- Distribution lists
- Awkward introductory period
  - Old versions of mail programs; bounces
- Very slow/small-memory machines
  - Can implement "post office" (CPU), but:
  - Who gets to be the Post Office? Trust?
- Cache Thrashing (memory-bound)
- The Subverters or Zombies

# Conclusions: cryptography

- Can only move and simplify your problems
- Solid results, but still relying on a large number of unproven assumptions and beliefs
- Not the bottleneck or problem in most security systems

- *To paraphrase Laotse, you cannot create trust with cryptography, no matter how much cryptography you use -- Jon Callas.*

# Conclusions (2): cryptography

- Leave it to the experts
- Do not do this at home
- Make sure you can upgrade
- Implementing it correctly is hard

- Secure computation very challenging and promising: reduce trust in individual building blocks

# SPAM

- L. F. Cranor, B.A. LaMacchia: Spam!; Communications of the ACM 1998, 21 (8), 74-83.
- C. Dwork, M. Naor: Pricing via Processing or Combatting Junk Mail; Crypto '92, LNCS 740, Springer-Verlag, Berlin 1992, 139-147.
- E. Gabber, M. Jacobsson, Y. Matias, A. Mayer: Curbing Junk E-Mail via Secure Classification; 2nd International Conference on Financial Cryptography (FC '98), LNCS 1465, Springer-Verlag, Berlin 1998, 198-213.
- A. Juels, J. Brainard: Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks; 6th ISOC Symposium on Network and Distributed System Security (NDSS '99), IEEE Press, 1999, 151-165.
- M. Abadi, M. Burrows, M. Manasse, E. Wobber, Moderately hard, memory-bound functions, Proceedings of the 10th Annual Network and Distributed System Security Symposium (February 2003), 25-39.
- C. Dwork, A. Goldberg, M. Naor, On Memory-Bound Functions for Fighting Spam, Crypto 2003, 426-444.

# Selected books on cryptology

- D. Stinson, Cryptography: Theory and Practice, CRC Press, 3rd Ed., 2005. Solid introduction, but only for the mathematically inclined.
- A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997. The bible of modern cryptography. Thorough and complete reference work – not suited as a first text book. Freely available at http://www.cacr.math.uwaterloo.ca/hac
- N. Smart, Cryptography, An Introduction: 3rd Ed., 2008. Solid and up to date but on the mathematical side. Freely available at http://www.cs.bris.ac.uk/~nigel/Crypto_Book/
- B. Schneier, Applied Cryptography, Wiley, 1996. Widely popular and very accessible – make sure you get the errata.
- Other authors: Johannes Buchmann, Serge Vaudenay