# New developments in cryptology

Prof. Bart Preneel
COSIC
Bart.Preneel(at)esatDOTkuleuven.be
http://homes.esat.kuleuven.be/~preneel

March 2013

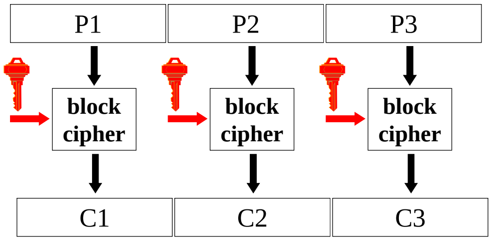
## Outline

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

## Outline

- Block ciphers/stream ciphers
- Hash functions/MAC algorithms
- Modes of operation and authenticated encryption
- How to encrypt/sign using RSA
- Multi-party computation
- Concluding remarks

## Block ciphers



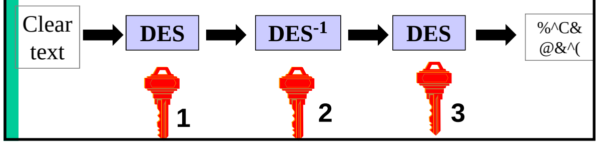
- larger data units: 64…128 bits
- memoryless
- repeat simple operation (round) many times

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

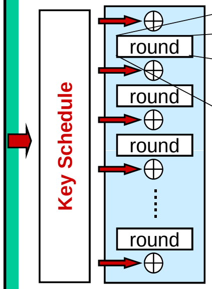
(May 2004)

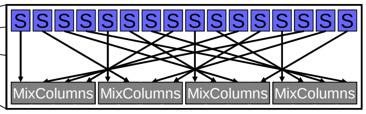
- Single DES abandoned
- two-key triple DES: until 2009 (80 bit security)
- three-key triple DES: until 2030 (100 bit security)

Highly vulnerable to a related key attack



## AES (2001)





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

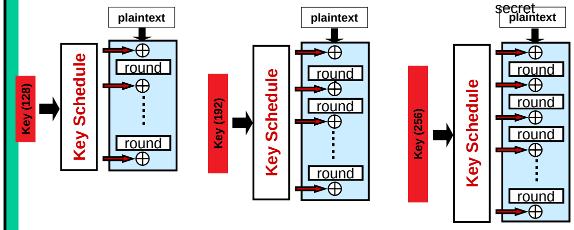
A $ 10M machine that cracks a DES key in 1 second would take **149 trillion years** to crack a 128-bit key

## 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: 1953 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

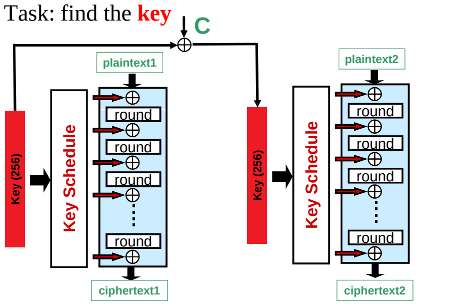
## AES-256 security

- 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

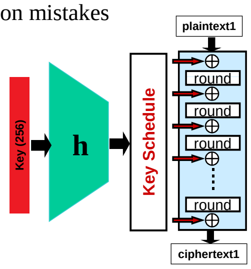
## What is a related key attack?

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

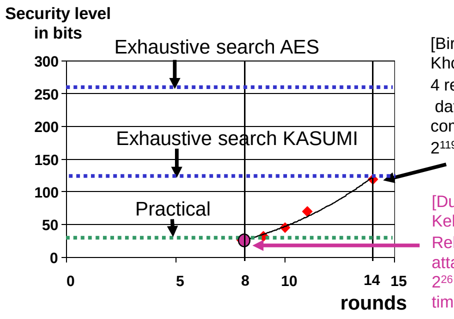


## Should I worry about a related key attack?

- Very hard in practice (except some old US banking schemes and IBM control vectors)
- 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)

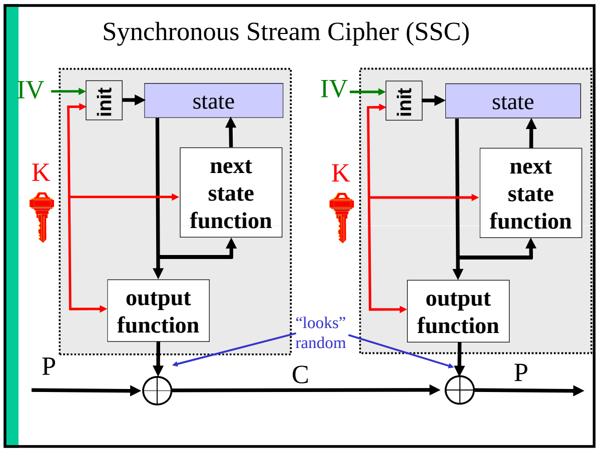
## New announcement: August 2011

[Bogdanov-Khovratovich-Rechberger]

- No related keys (attack in 2005)
- For AES-128: with $2^{88}$ plaintext/ciphertext pairs, the effective key size can be reduced by 2 bits (AES-126)
- For AES-192: only $2^{80}$ plaintext/ciphertext pairs
- For AES-256: only $2^{40}$ plaintext/ciphertext pairs

Very minor impact on security and very hard to extend

## 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

## GSM

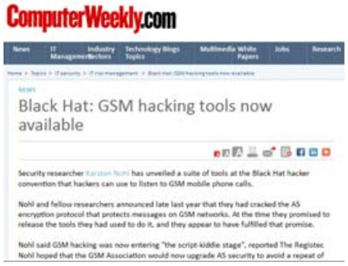
- A5/1 weak
  - [Barkan+03] requires seconds (software not available so requires math)
  - [Nohl10]: Kraken = 2 Terabyte of Rainbow tables http://reflextor.com/trac/a51
- A5/2 trivially weak (milliseconds) – withdrawn in 2007 (took 8 years)
- A5/3 (= Kasumi) seems ok but slow adoption (even if in 1.2 billion out of 3 billion handsets)
- Simpler attacks on GSM
  - eavesdrop after base station (always cleartext)
  - switch off encryption (can be detected)
  - SMS of death

## GSM

ISSE 2010

- growing number of open source tools to intercept: GnuRAdio, Airprobe, OpenBTS
- but needs more work (1-2 years?)

Jan 2011



ComputerWeekly.com

News

Black Hat: GSM hacking tools now available

Security researcher Karsten Nohl has unveiled a suite of tools at the Black Hat hacker convention that hackers can use to listen to GSM mobile phone calls.

Nohl and fellow researchers announced late last year that they had cracked the A5 encryption protocol that protects messages on GSM networks. At the time they promised to release the tools they had used to do it, and they appear to have fulfilled that promise.

Nohl said GSM hacking was now entering "the script-kiddie stage", reported The Register. Nohl hoped that the GSM Association would now upgrade A5 security to avoid a repeat of

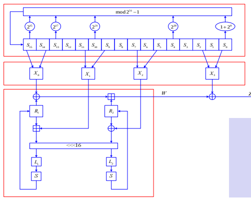
## GSM

- be careful when rolling out 2-factor authentication via SMS
- war texting hacks on car systems and SCADA systems [Black Hat, Aug'11]

intercepting mobile phone traffic is illegal

## International

- China:
  - ZUC as 3rd algorithm in LTE (also SMII, SMIII)
  - National Cryptography Industry Standards Technical Committee established on 19/10/2011



Every algorithm used in China needs to be designed in China

- US: NIST is very active in standardization

## Open competition for stream ciphers

http://www.ecrypt.eu.org

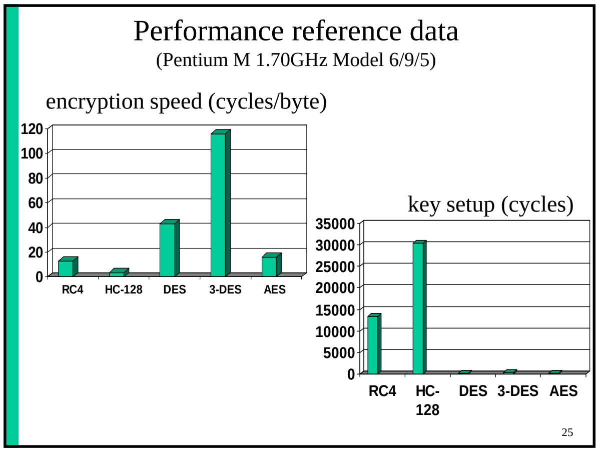
- run by ECRYPT
  - high performance in software (32/64-bit): 128-bit key
  - low-gate count hardware (< 1000 gates): 80-bit key
  - variants: authenticated encryption
- April 2005: 33 submissions
- many broken in first year
- April 2008: end of competition

## 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 |



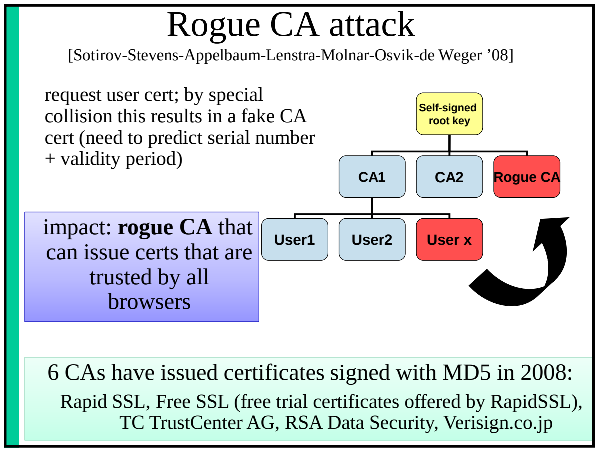
## Performance reference data

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

encryption speed (cycles/byte)

key setup (cycles)

25



## Rogue CA attack

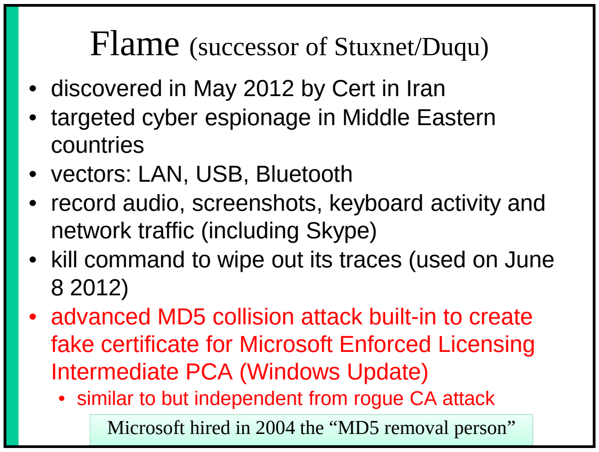
[Sotirov-Stevens-Appelbaum-Lenstra-Molnar-Osvik-de Weger '08]

request user cert; by special collision this results in a fake CA cert (need to predict serial number + validity period)

impact: **rogue CA** that can issue certs that are trusted by all browsers

6 CAs have issued certificates signed with MD5 in 2008:

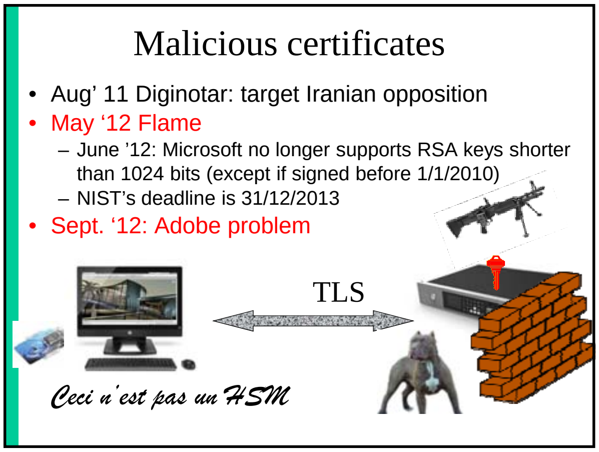
Rapid SSL, Free SSL (free trial certificates offered by RapidSSL), TC TrustCenter AG, RSA Data Security, Verisign.co.jp



## Flame (successor of Stuxnet/Duqu)

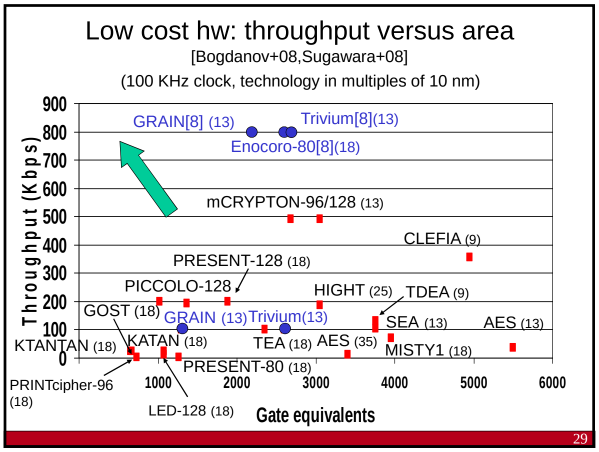
- discovered in May 2012 by Cert in Iran
- targeted cyber espionage in Middle Eastern countries
- vectors: LAN, USB, Bluetooth
- record audio, screenshots, keyboard activity and network traffic (including Skype)
- kill command to wipe out its traces (used on June 8 2012)
- advanced MD5 collision attack built-in to create fake certificate for Microsoft Enforced Licensing Intermediate PCA (Windows Update)
  - similar to but independent from rogue CA attack

Microsoft hired in 2004 the "MD5 removal person"



## Malicious certificates

- Aug' 11 Diginotar: target Iranian opposition
- May '12 Flame
  - June '12: Microsoft no longer supports RSA keys shorter than 1024 bits (except if signed before 1/1/2010)
  - NIST's deadline is 31/12/2013
- Sept. '12: Adobe problem

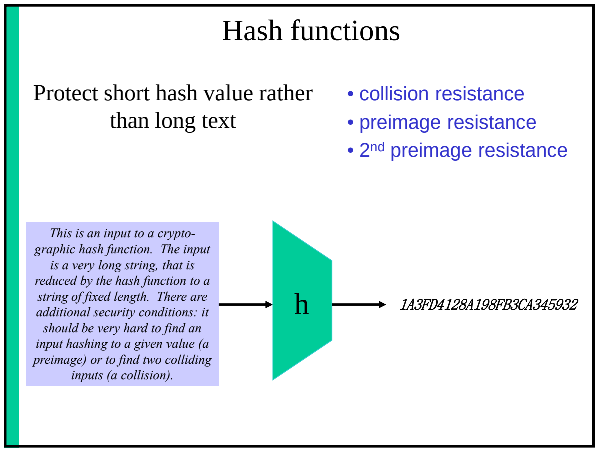


## Low cost hw: throughput versus area

[Bogdanov+08,Sugawara+08]

(100 KHz clock, technology in multiples of 10 nm)

29

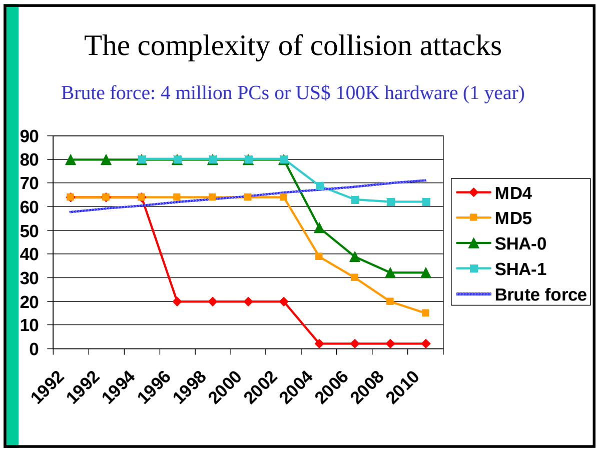


## Hash functions

Protect short hash value rather than long text

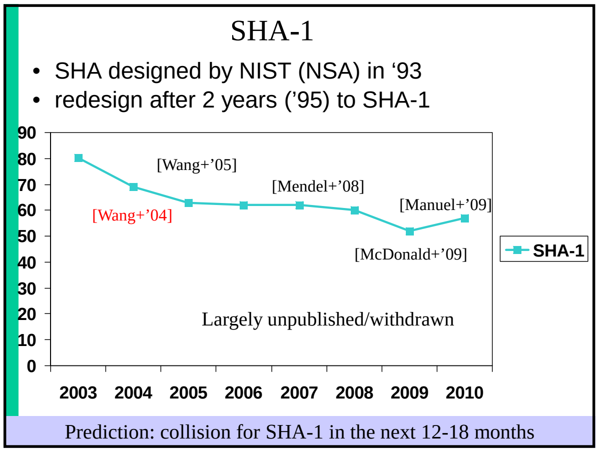
- collision resistance
- preimage resistance
- 2nd preimage resistance

*This is an input to a cryptographic hash function. The input is a very long string, that is reduced by the hash function to a string of fixed length. There are additional security conditions: it should be very hard to find an input hashing to a given value (a preimage) or to find two colliding inputs (a collision).*



### The complexity of collision attacks

Brute force: 4 million PCs or US$ 100K hardware (1 year)

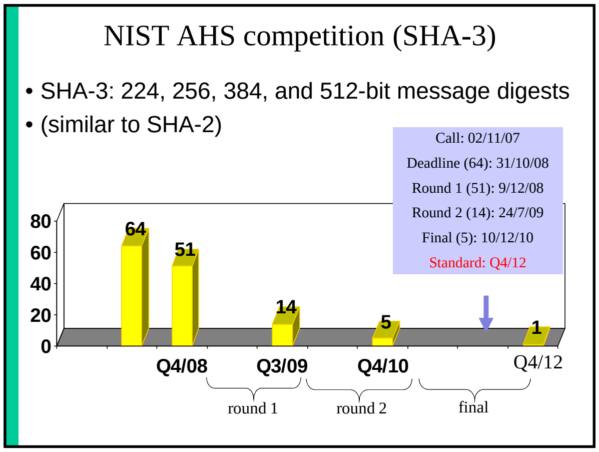


### SHA-1

- SHA designed by NIST (NSA) in ‘93
- redesign after 2 years (’95) to SHA-1

Largely unpublished/withdrawn

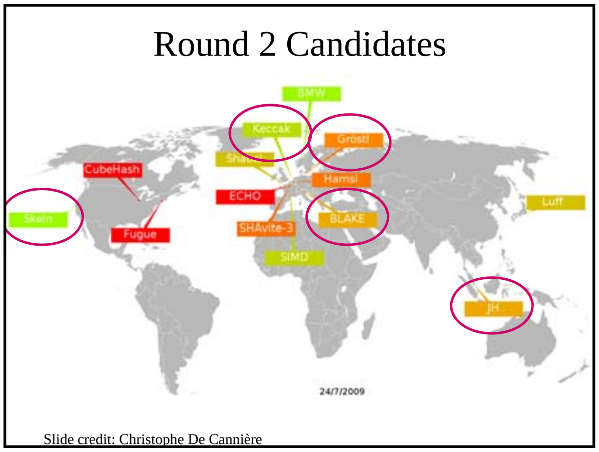
Prediction: collision for SHA-1 in the next 12-18 months



### NIST AHS competition (SHA-3)

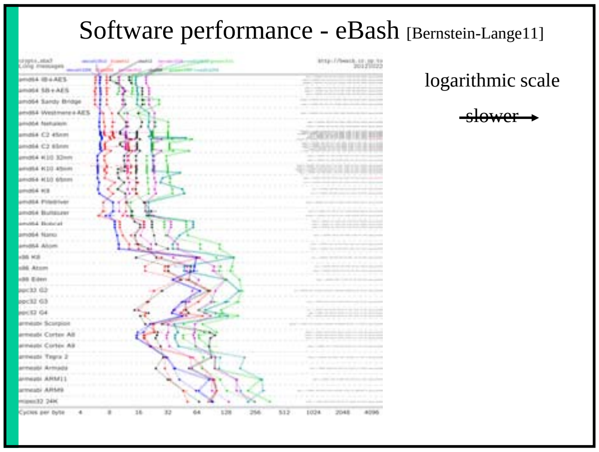
- SHA-3: 224, 256, 384, and 512-bit message digests
- (similar to SHA-2)

Call: 02/11/07
Deadline (64): 31/10/08
Round 1 (51): 9/12/08
Round 2 (14): 24/7/09
Final (5): 10/12/10
Standard: Q4/12



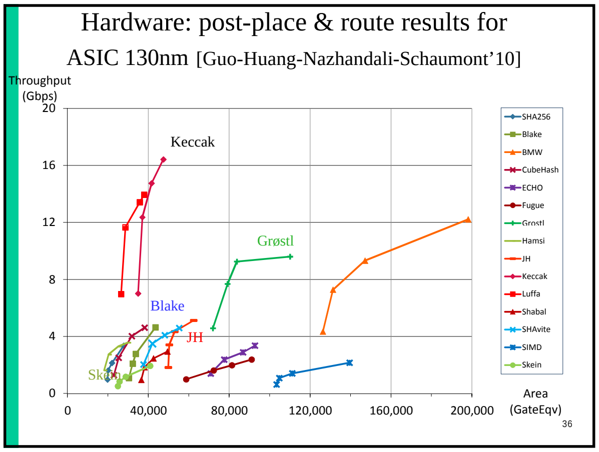
### Round 2 Candidates

Slide credit: Christophe De Cannière



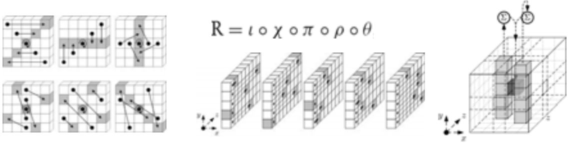
### Software performance - eBash [Bernstein-Lange11]

logarithmic scale



### Hardware: post-place & route results for ASIC 130nm [Guo-Huang-Nazhandali-Schaumont’10]

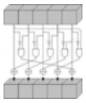
## Keccak



permutation: 25, 50, 100, 200, 400, 800, 1600

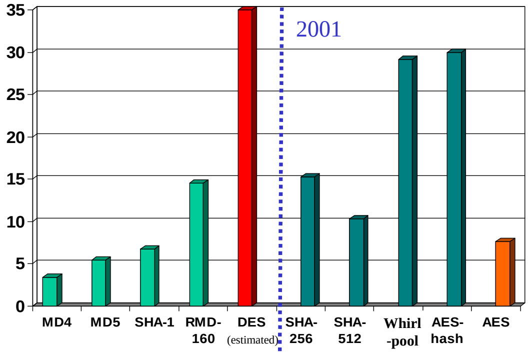
nominal version:

- 5x5 array of 64 bits
- 18 rounds of 5 steps



## Performance of hash functions - Bernstein

(cycles/byte) Intel Intel Core 2 Quad Q9550; 4 x 2833MHz



## MAC algorithms

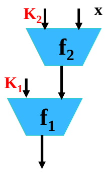
- EMAC based on AES
- HMAC based on MD5/SHA-1
- GMAC
- UMAC

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

39

## 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 |

40

## 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 $K_1$ = 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

41

## 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 $x_i \in GF(2^{32})$
- $g(x) = prf_K(h(x))$
- $h(x) = (\Sigma_{i=1}^{512}\ (x_{2i-1} + k_{2i-1}) \bmod 2^{32} \cdot (x_{2i} + k_{2i}) \bmod 2^{32})\bmod 2^{64}$

- 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

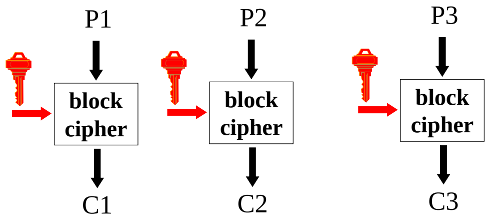
42

## How to use cryptographic algorithms

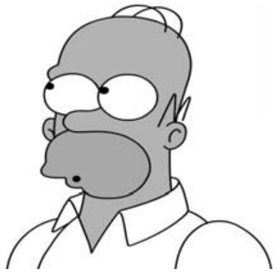
- Modes of operation
- Padding and error messages
- Authenticated encryption

- How to encrypt with RSA

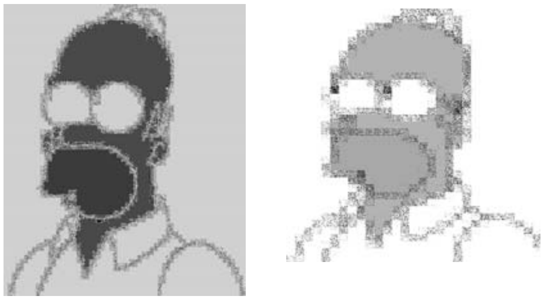
## How NOT to use a block cipher: ECB mode



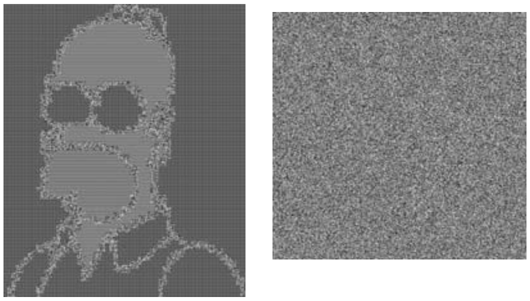
## An example plaintext



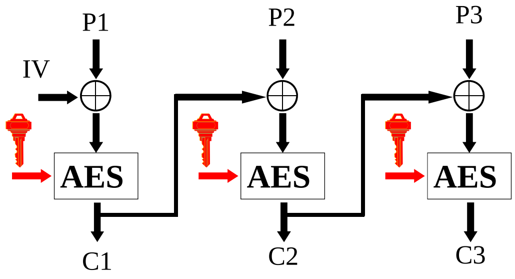
## Encrypted with substitution and transposition cipher

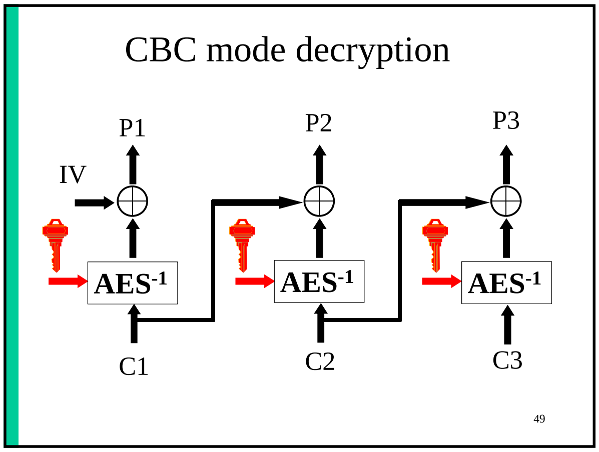


## Encrypted with AES in ECB and CBC mode



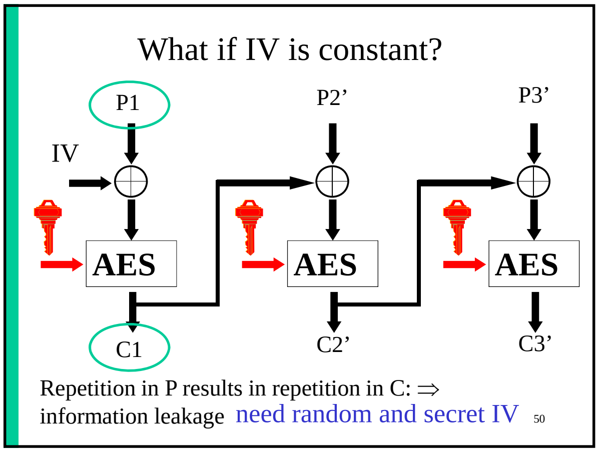
## How to use a block cipher: CBC mode





## CBC mode decryption

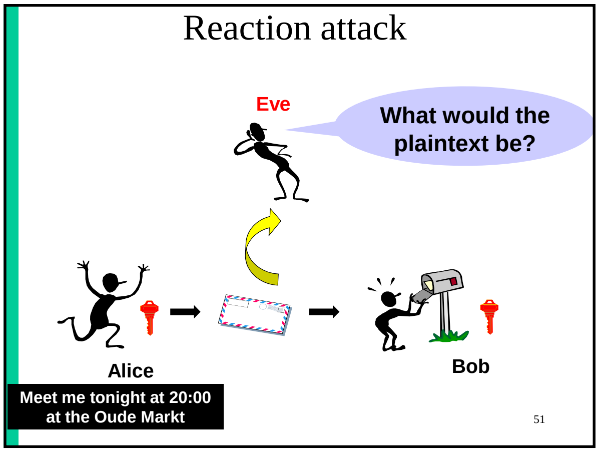
49



## What if IV is constant?

Repetition in P results in repetition in C: ⇒ information leakage need random and secret IV

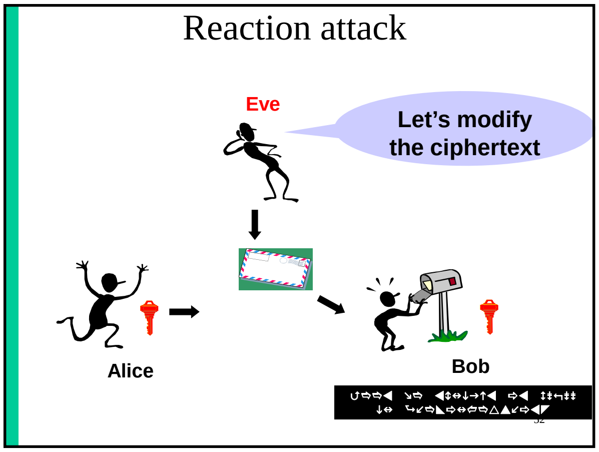
50



## Reaction attack

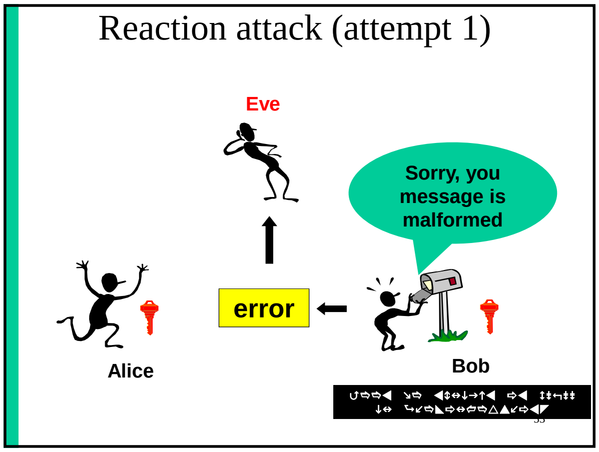
Meet me tonight at 20:00 at the Oude Markt

51



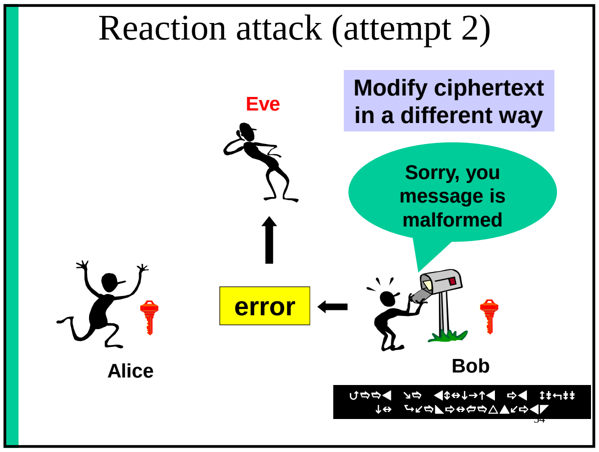
## Reaction attack

52



## Reaction attack (attempt 1)

53

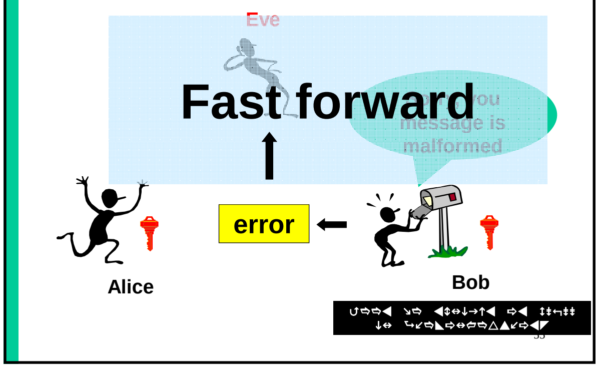


## Reaction attack (attempt 2)

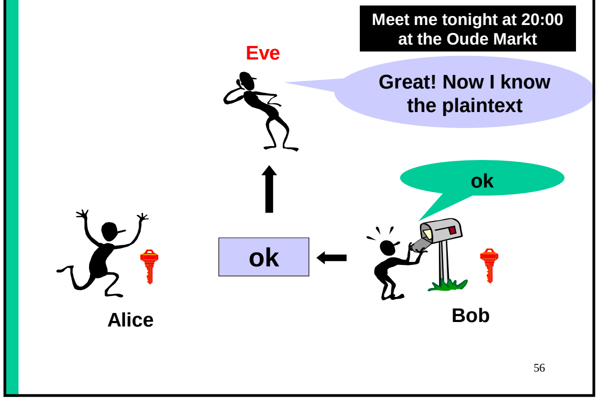
Modify ciphertext in a different way
Eve
Sorry, you message is malformed
error
Alice
Bob
↺⇨⇨◀ ↘⇨ ◀⇕⇔↓→↑◀ ⇨◀ ↨⇟↰⇟⇟ ↓⇔ ↳↙⇨◣⇨⇔⇦⇨△▲↙⇨◀◤

54

## Reaction attack (attempt 3)



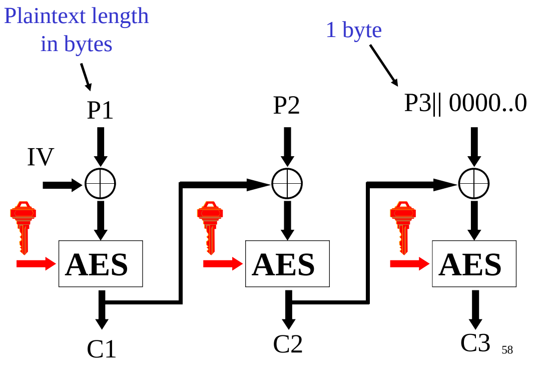
## Reaction attack (attempt 973)


Meet me tonight at 20:00 at the Oude Markt
Eve
Great! Now I know the plaintext
ok
ok
Alice
Bob

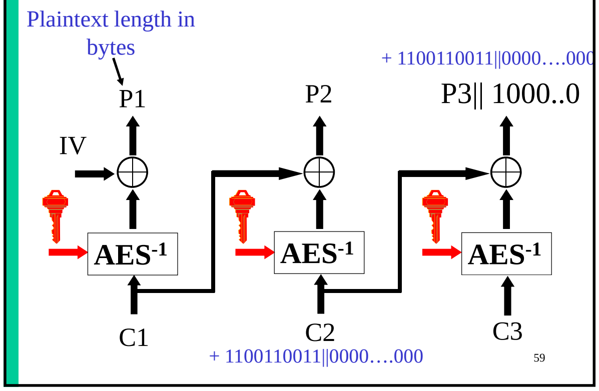
## Reaction attacks: well known but…

- [Bleichenbacher98] PKCS #1v1.5 – 1 million chosen ciphertexts; improved by [Klima-Pokorny-Rosa03]
- [Manger01] OAEP PKCS #1v2 – a few 1000 chosen ciphertexts
- [Bellare-Kohno-Namprempre 02]: SSH
- [Vaudenay'02] SSL, IPsec, WTLS...
- [Canvel-Hiltgen-Vaudenay-Vuagnoux03]: SSL/TLS
- 2010: ASP.NET
- 2011 XML encryption
- Solutions:
  - don't send error messages (bad engineering practice)
  - authenticated encryption
  - MAC the ciphertexts and do not decrypt if MAC is incorrect

## 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: a careful implementation of random padding or authenticated encryption

## XML Encryption attack

- **Reaction attack:** chosen plaintext (decryption queries) and observe error message
- XML decryption checks validity of plaintext (specific character encoding)
- [Jager-Somorovsky11] decrypt 160 bytes using 2000 decryption queries (100 seconds)

- Countermeasure:
  - unified error message
  - changing mode
  - authenticated encryption: non-trivial

## 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

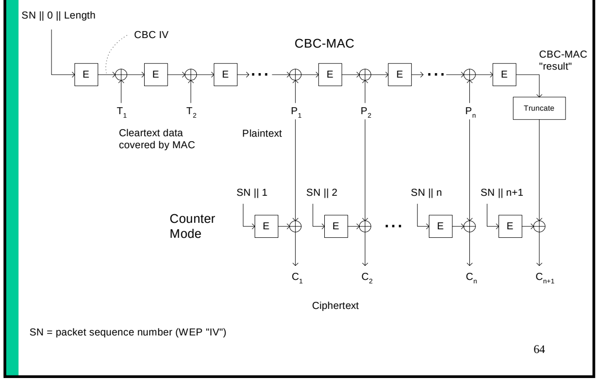
- needed for network security, but only fully understood by crypto community around 2000 (too late)
- **dump CBC mode!!**
- standards:
  - CCM: CTR + CBC-MAC [NIST SP 800-38C]
  - GCM: CTR + GMAC [NIST SP 800-38D]
- both are suboptimal but patent free

- properties
  - associated data
  - parallelizable
  - on-line
  - “provable” security

- IAPM
- XECB
- OCB

patented

## Example: CCM: CTR + CBC-MAC



SN = packet sequence number (WEP "IV")

## Outline

- Block ciphers/stream ciphers
- Hash functions/MAC algorithms
- Modes of operation and authenticated encryption
- How to encrypt/sign using RSA
- Multi-party computation
- Concluding remarks

## RSA (‘78)

- choose 2 “large” prime numbers p and q
- modulus n = p.q
- compute $\lambda(n) = \mathrm{lcm}(p-1,q-1)$
- choose e relatively prime w.r.t. $\lambda(n)$
- compute $d = e^{-1} \bmod \lambda(n)$

- public key = (e,n)
- private key = d of (p,q)

The security of RSA is based on the “fact” that it is easy to generate two large primes, but that it is hard to factor their product

- encryption: $c = m^e \bmod n$
- decryption: $m = c^d \bmod n$

try to factor 2419

## Public-Key Cryptology

- new factorization record in January 2010: 768 bits
- upgrade your RSA-1024 keys (should have been doen in 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)
  - limited commercial deployment outside government
- progress on pairings leading to more efficient protocols

## Key lengths for confidentiality

http://www.ecrypt.eu.org

| **duration** | symmetric | RSA | ECC |
|---|---|---|---|
| days/hours | 50 | 512 | 100 |
| 5 years | 73 | 1024 | 146 |
| 10-20 years | 103 | 2048 | 206 |
| 30-50 years | 141 | 4096 | 282 |

Assumptions: no quantum computers; no breakthroughs; limited budget

## Generation of key pairs

“Ron was wrong, Whit is right”

*http://print.iacr.org/2012/064.pdf*

- 11.7 million openly accessible public keys
- 6.4 million distinct RSA moduli
- rest: ElGamal/DSA (50/50) and 1 ECDSA

- 1.1% of RSA keys occur in >1 certificate
- 0.2% (12934 moduli) are easy to factor, because they form pairs like: n = p.q and n’ = p’.q so gcd(n,n’)=q
  - 40% of these have valid certs
  - reason: only 40-bit randomness in key generation combined with the birthday paradox
- less of a problem for ElGamal/DSA: need to know how randomness is produced and complexity is $2^{40}$ key generations
- ethical problem: how to report this?

## Quantum computers?

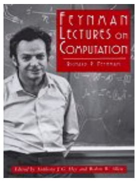
- exponential parallelism

*n* coupled quantum bits

⇩

$2^n$ degrees of freedom !

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



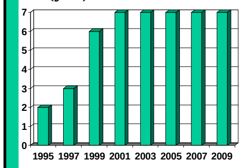
## If a large quantum computer can be built...

- 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**

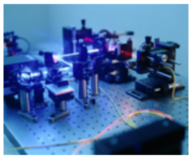
Photon 'machine gun' could power quantum computers

## Quantum cryptography

- no solution for entity authentication problem (bootstrapping needed with secret keys)
- no solution (yet) for multicast
- dependent on physical properties of communication channel
- cost
- implementation weaknesses (side channels)

## Quantum cryptography

- Security based
  - on the assumption that the laws of quantum physics are correct
  - rather than on the assumption that certain mathematical problems are hard



## Quantum hacking

http://www.iet.ntnu.no/groups/optics/qcr/

## 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)

77

## 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)

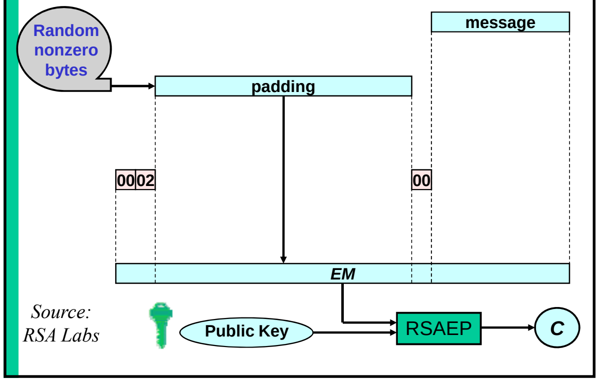
78

## 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, ...

79

## 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 $s^e$ mod n
  - ask for decryption of c’: m’
  - compute m as m’/s mod 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 mod n) < 3B
  - with B = $2^{8(k-2)}$ (k length in bytes of the modulus)

## “Efficient padding oracle attacks on cryptographic hardware” (PKCS#11 devices)

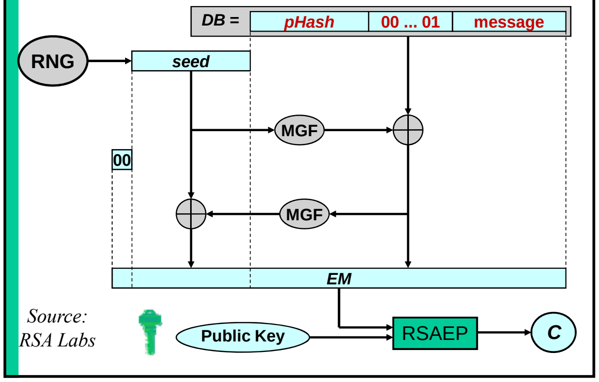
[Bardou+ 12] most attacks take less than 100 milliseconds

| Device | PKCS#1v1.5 | | CBC pad | |
|---|---|---|---|---|
| | token | session | token | session |
| Aladdin eTokenPro | 🟥 | 🟥 | 🟥 | 🟥 |
| Feitian ePass 2000 | OK | OK | N/A | N/A |
| Feitian ePass 3003 | OK | OK | N/A | N/A |
| Gemalto Cyberflex | 🟥 | N/A | N/A | N/A |
| RSA Securid 800 | 🟥 | N/A | N/A | N/A |
| Safenet iKey 2032 | 🟥 | 🟥 | N/A | N/A |
| SATA dKey | OK | OK | OK | OK |
| Siemens CardOS | 🟥 | 🟥 (89 secs) | N/A | N/A |

## 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 included in ISO 18033
- Similar problems for signatures:
  ISO 9796-1 broken, PKCS#1 v1.0 questionable

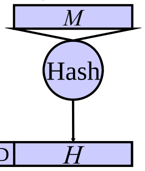
## How to sign with RSA?

- public key: (n,e)
- private key: d
- $s = t^{d} \bmod n = t^{1/e} \bmod n$
- But
  - message M is often larger than modulus n
  - RSA(x*y) = RSA(x)*RSA(y)
  - RSA(0) = 0, RSA(1) = 1,…
- Solution: hash and add redundancy
  - PKCS #1
  - RSA-PSS

## How (not) to sign with RSA: an 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)

## RSA Signatures: PKCS #1 v1.5 [source: RSA Labs]



public key: (n,e)

private key: d

t = | 00 01 ff ff ff ff ff … ff ff ff 00 | HashID | *H* |

Generation of RSA signature on M: $s = t^{d} \bmod n = t^{1/e} \bmod n$

Verification of RSA signature s on M

Compute $t = s^{e} \bmod n$ *and check that t has the required format*

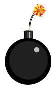
Problem: most signature verification software would accept a signature on M of the following form:

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

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

[Bleichenbacher06]

| 00 01 ff... ff 00 | HashID | H | Magic |
|---|---|---|---|



- consider RSA with public exponent e = 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
  - example of a perfect cube 1728 = $12^3$
- 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

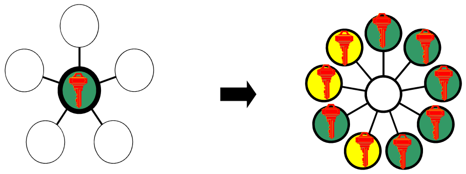
## Fix of Bleichenbacher's attack

- 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

## Secure implementations of cryptography

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

## Secure Computation



- PKI
- Banking
- Credit card
- Google
- …

Multi-party computation

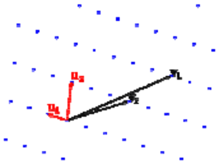
"you can trust it because you don't have to"

## Multi-party computation becomes "truly practical"

- Similar to first public key libraries 20 years ago
  - EU: CACE project (Computer Aided Cryptography Engineering), www.cace-project.eu
  - US: Brown Univ. + UCSD (Usenix 2010)
- Examples
  - efficient zero-knowledge proofs
  - 2-party computation of AES (Bristol)
  - secure auction of beetroots in Denmark (BRICS)
  - oblivious transfer for road pricing (COSIC)

## Fully homomorphic encryption

- From E(x) and E(y), you can compute E(x+y), E(c.x) and E(x.y) **without decrypting**
- Many cool applications including cloud computing
- [Gentry'09] ideal lattices = breakthrough
- First implementations require only seconds [Vercauteren-Smart'10], [Gentry-Halevi'10]….
  - but to ciphertext for 1 bit is 3 million bits and public key is several Mbyte
- Substantial improvements if restricted operations



lattice ←   lettuce →

## Cryptographic algorithm selection

- Standards?
- Public domain versus proprietary
- Upgrades

98

## 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

100

## Independent evaluation efforts

- NIST (US) (1997-2001): block cipher AES for FIPS 197 (http://csrc.nist.gov/CryptoToolkit/aes/)
- CRYPTREC (Japan) (2000-2003 and 2009-2012): 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
- NIST (US) (2007-2012): hash function SHA-3 for FIPS 197 (http://csrc.nist.gov/CryptoToolkit/aes/)

101

## 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

102

## 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

103

## 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)

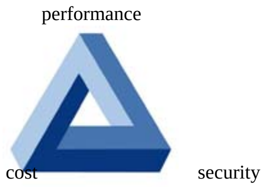
- Authenticated encryption mode (OCB, CWC, CCM, or even GCM) with 3-key 3-DES or AES
- Hash functions: RIPEMD-160, SHA-256, SHA-512 or Whirlpool
- Public key encryption: RSA-KEM or ECIES
- Digital signatures: RSA-PSS or ECDSA
- Protocols: TLS 1.2, SSH, IKE(v2)

## Challenges for crypto

security for 50-100 years
authenticated encryption of Terabit/s networks
ultra-low power/footprint

secure software and hardware implementations

algorithm agility



## 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