Cryptology Principles

Prof. Bart Preneel
COSIC
Firstname.lastname(at)esatDOTkuleuven.be
http://homes.esat.kuleuven.be/~preneel
December 2014

Outline

- Concepts and algorithms
  - symmetric algorithms for confidentiality
  - symmetric algorithms for data authentication
  - public-key cryptology
- Cryptology: protocols
  - identification/entity authentication
  - key establishment
- Network security protocols

Definitions

Confidentiality
Integrity
Availability

	data	entities
confidentiality	encryption	anonymity
authentication	data authentication	identification

Authorisation

Non-repudiation of origin, receipt

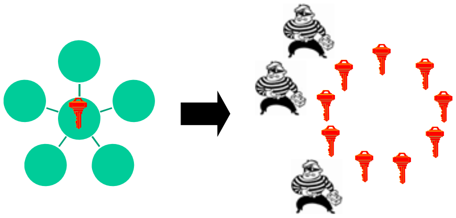
Contract signing

Notarisation and Timestamping

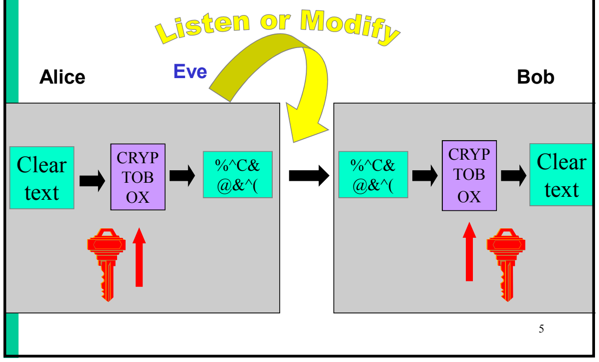
Don't use the word authentication without defining it

Secure multi-party computation

- auctions
- medical statistics and advice
- e-voting
- road pricing
- (social) search



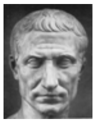
Cryptology: basic principles



Old cipher systems (pre 1900)

- Caesar cipher: shift letters over k positions in the alphabet (k is the secret key)

```
THIS IS THE CAESAR CIPHER
WKLV LV WKH FDHVDU FLSKHU
```



- Julius Caesar never changed his key (k=3).

Cryptanalysis example:

```
TIPGK RERCP JZJZJ WLE     GVCTX EREPC WMWMW JYR
UJQHL SFSDQ KAKAK XMF     HWDUY FSFQD XNXNX KZS
VKRIM TGTER LBLBL YNG     IXEVZ GTGRE YOYOY LAT
WLSJN UHUFS MCMCM ZOH     JYFWA HUHSF ZPZPZ MBU
XDTKO VOVGT NDNDN API     KZGXB IVITG AQAQA NCV
YNULP WKWHU OEOEO BQJ     LAHYC JWJUH BRBRB ODW
ZOVMQ XKXIV PFPFP CRK     MBIZD KXKVI CSCSC PEX
APWNR YLYJW QGQGQ DSL     NCJAE LYLWJ DTDTD QFY
BQXOS ZMXKX RHRHR ETM     ODKBF MZMXK EUEUE RGZ
CRYPT ANALY SISIS FUN     PELCG NANYL FVFVF SHA
DSZQU BOBMZ TJTJT GVO     QFMDH OBOZM GWGWG TIB
ETARV CPCNA UKUKU HWP     RGNEI PCPAN HXHXH UJC
FUBSW DQDOB VLVLV IXQ     SHOFJ QDQBO IYIYI VKD
```

Plaintext?   k = 17

Old cipher systems (pre 1900) (2)

- Substitutions
  - ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - MZNJSOAXFQGYKHLUCTDVWBIPER

! Easy to break using statistical techniques

- Transpositions

```
TRANS      OIS R
POSIT      NOTIT
IONS       OSANP
```

Security

- there are n! different substitutions on an alphabet with n letters
- there are n! different transpositions of n letters
- n=26: **n!=403291461126605635584000000 = 4 . 10^{26} keys**
- trying all possibilities at 1 nanosecond per key requires....

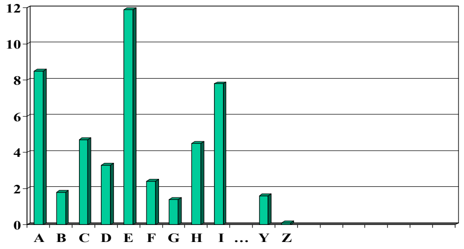
4.10^{26} /(10^{9} . 10^{5} . 4 10^{2}) = 10^{10} years

keys per second

seconds per day

days per year

Letter distributions

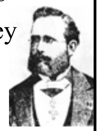


Assumptions on Eve (the opponent)

- A scheme is broken if Eve can deduce the key or obtain additional plaintext
- Eve can always try all keys till “meaningful” plaintext appears: a brute force attack
  - solution: large key space
- Eve will try to find shortcut attacks (faster than brute force)
  - history shows that designers are too optimistic about the security of their cryptosystems

Assumptions on Eve (the opponent)

- Cryptology = cryptography + cryptanalysis
- Eve knows the algorithm, except for the key (Kerckhoffs’s principle)
- increasing capability of Eve:
  - knows some information about the plaintext (e.g., in English)
  - knows part of the plaintext
  - can choose (part of) the plaintext and look at the ciphertext
  - can choose (part of) the ciphertext and look at the plaintext

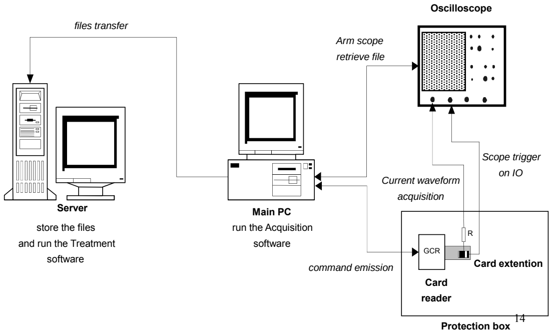


New assumptions on Eve

- Eve may have access to side channels
  - timing attacks
  - simple power analysis
  - differential power analysis
  - acoustic attacks
  - electromagnetic interference
- Eve may launch (semi-)invasive attacks
  - differential fault analysis
  - probing of memory or bus

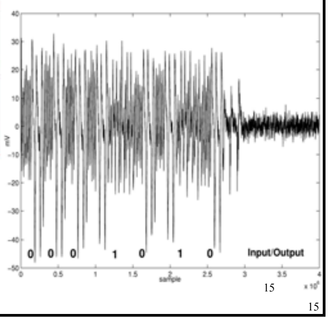
13

Side channel analysis



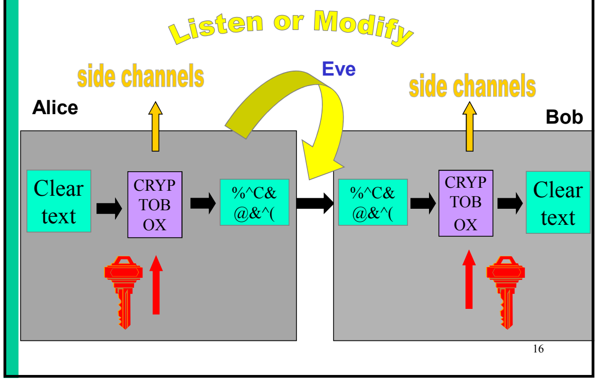
14

Side channel analysis: EMA



15

Cryptology + side channels

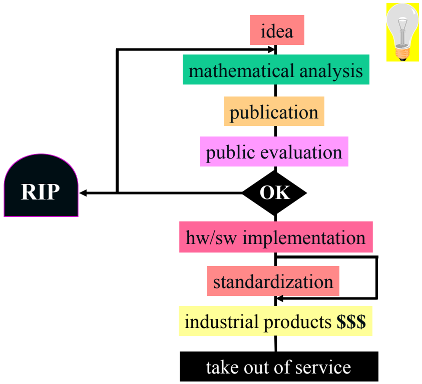


16

The Rotor machines (WW II)

17

Life cycle of a cryptographic algorithm



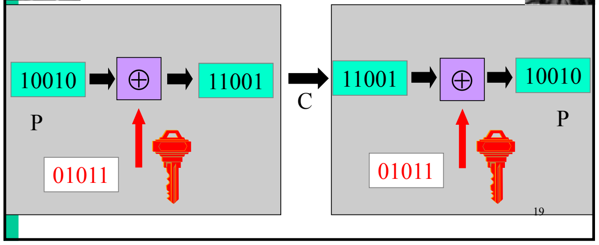
18

Vernam scheme (1917)
Mauborgne: one time pad (1917+x)

Shannon (1948)

F. Miller (1882)

key is random string, as long as the plaintext
information theoretic proof of security



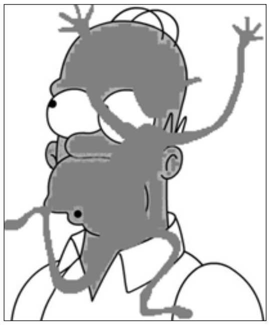
Vernam scheme: Venona

http://www.nsa.gov/public_info/declass/venona/

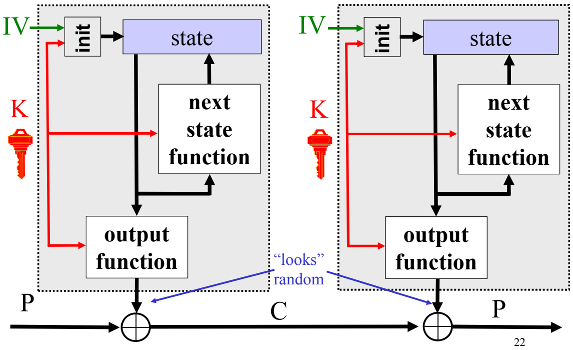
- $c_1 = p_1 + k$
- $c_2 = p_2 + k$
- then $c_1 - c_2 = p_1 - p_2$

- a skilled cryptanalyst can recover p_1 and p_2 from $p_1 - p_2$ using the redundancy in the language

Example: c1 V c2 (not +)



Synchronous Stream Cipher (SSC)



Exhaustive key search

- 2014: 2^{40} instructions is easy, 2^{60} is somewhat hard, 2^{80} is hard, 2^{128} is completely infeasible
  - 1 million machines with 16 cores and a clock speed of 4 GHz can do 2^{56} instructions per second or 2^{80} per year
  - trying 1 key requires typically a few 100 instructions

- Moore's "law": speed of computers doubles every 18 months: key lengths need to grow in time
  - but adding 1 key bit doubles the work for the attacker

- Key length recommendations in 2014
  - < 70 bits: insecure
  - 80 bits: one year (but not for NSA)
  - 100 bits: 20 years

- More details http://www.ecrypt.eu.org

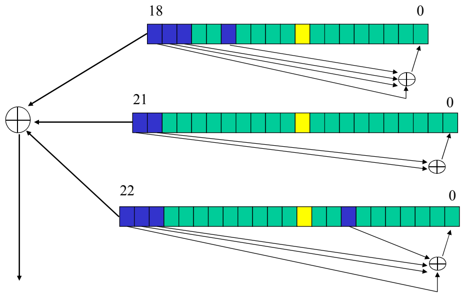
SSC: Specific properties

- Recipient needs to be synchronized with sender
- No error-propagation
  - excellent for wireless communications
- Key stream independent of data
  - key stream can be precomputed
  - particular model for cryptanalysis: attacker is not able to influence the state
  - Big concern is reuse of key stream:

Practical stream ciphers

- A5/1 (GSM) (64 or 54) - broken
- E0 (Bluetooth) (128) - broken
- RC4 (browser) (40-128) - insecure
- SNOW-3G (3GSM) (128) – ok
- Salsa20/12 (256)
- HC-128 (128)
- Grain (80/128)
- Trivium (80)

A5/1 stream cipher (GSM)



Clock control: registers agreeing with majority are clocked (2 or 3) 🟨

A5/1 stream cipher (GSM)

A5/1 attacks

- exhaustive key search: 2^{64} (or rather 2^{54})
- search 2 smallest registers: 2^{43} values – a few steps to verify a guess

- [BB05]: 10 minutes on a PC,
  - 3-4 minutes of ciphertext only

A simple cipher: RC4 (1987)

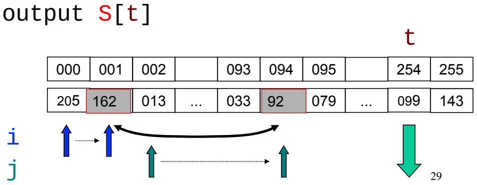
- designed by Ron Rivest (MIT)
- leaked in 1994
- S[0..255]: secret table derived from user key K

```
for i=0 to 255 S[i]:=i
j:=0
for i=0 to 255
    j:=(j + S[i] + K[i]) mod 256
    swap S[i] and S[j]
i:=0,  j:=0
```

A simple cipher: RC4 (1987)

Generate key stream which is added to plaintext

```
i:=(i+1) mod 256
j:=(j + S[i]) mod 256
swap S[i] and S[j]
t:=(S[i] + S[j]) mod 256
output S[t]
```



An improved version: Spritz (2014)

Generate key stream which is added to plaintext

```
i:= (i + w) mod 256
j:= (j + k + S[j + S[i]]) mod 256
k:= (i + k + S[j]) mod 256
swap S[i] and S[j]
z:= S[j + S[i + S[z+k]]] mod 256
output z
```

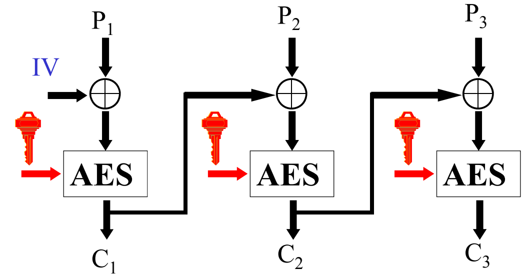
note: gcd(w,256)=1

Block cipher

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

Block cipher in CBC mode

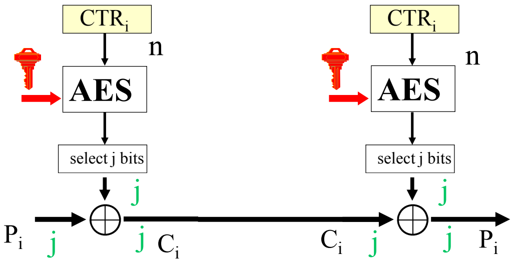
Cipher Block Chaining $C_i = E_K(P_i \oplus C_{i-1})$



need random and secret IV

CounTer Mode (CTR)

$C_i = P_i \oplus$ leftmost j bits of $E_K(CTR_{i-})$, CTR_i ++

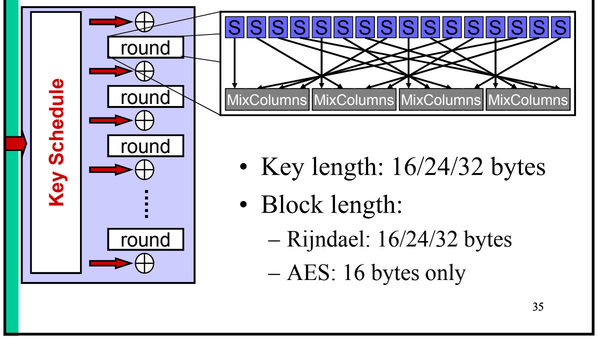


state initialized with random IV, or CTR_0 = IV, $j \le n$

Practical block ciphers

- 32-bit block ciphers
  - Keeloq (remote control for cars, garage doors)
- 64-bit block ciphers
  - DES: outdated
  - 3-DES: financial sector
  - KASUMI (3GSM)
  - GOST
- 128-bit block ciphers
  - AES: main standard

AES: Rijndael



- Key length: 16/24/32 bytes
- Block length:
  - Rijndael: 16/24/32 bytes
  - AES: 16 bytes only

Encryption

- Hides the content of the plaintext (confidentiality)
- But does **NOT**
  - protect against modifications (active eavesdropping)
  - hide the length of the plaintext (solution: random padding)
  - Hides who is communicating with whom (solution: many dummy messages)

Symmetric cryptology: data authentication

- the problem
- hash functions without a key
  - MDC: Manipulation Detection Code
- hash functions with a secret key
  - MAC: Message Authentication Code

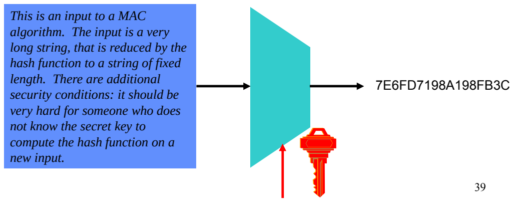
Data authentication: the problem

- encryption only provides confidentiality (passive eavesdropping)
- Bob wants to know:
  - the **source** of the information (data origin)
  - that the information has not been **modified**
  - (optionally) **timeliness** and **sequence**
- data authentication:
  - more complex than data confidentiality
  - more important for commercial applications

Data authentication: Message Authentication Code (MAC)

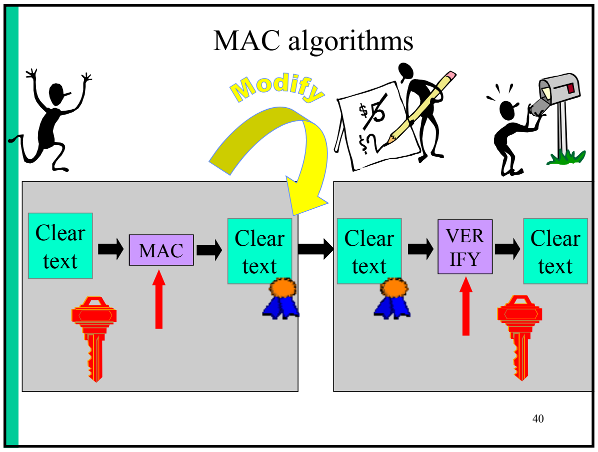
- Replace protection of authenticty of (long) message by protection of secrecy of (short) key
- Add MAC to the plaintext

- CBC-MAC
- HMAC
- GMAC



This is an input to a MAC algorithm. 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 for someone who does not know the secret key to compute the hash function on a new input.

MAC algorithms



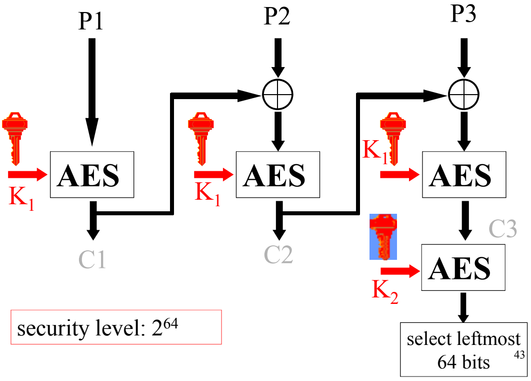
Data authentication: MAC

- typical MAC lengths: 32..96 bits
  - Forgery attacks: 2^m steps with m the MAC length in bits
- typical key lengths: (56)..112..160 bits
  - Exhaustive key search: 2^k steps with k the key length in bits
- birthday attacks: security level smaller than expected

Practical MAC algorithms

- Banking: CBC-MAC based on triple-DES
- Internet: HMAC, CBC-MAC based on AES
- information theoretic secure MAC algorithms (authentication codes): GMAC/GCM
  - highly efficient but rather long keys
  - part of the key refreshed per message: this is problematic (value “H” should also be refreshed)

CBC-MAC based on AES

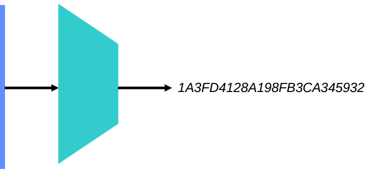


security level: 2^{64}

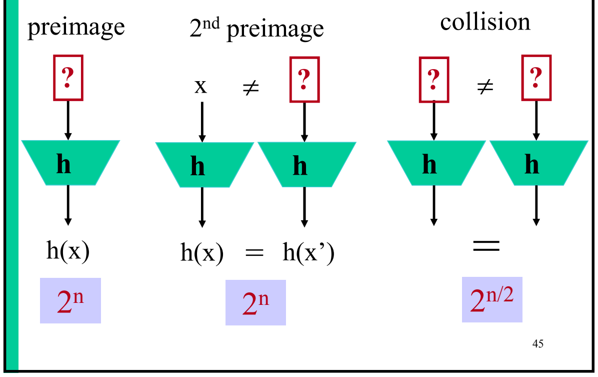
Data authentication: MDC

- MDC (manipulation detection code)
- Protect short hash value rather than long text

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).



MDC Security requirements (n-bit result)



Practical hash functions

- MD2: legacy (not very secure)
- MD4: broken
- MD5: broken
- SHA-1: broken
- RIPEMD-160: ok but legacy output
- SHA-2: ok
- SHA-3: new NIST standard in 2014
- Stribog (2013)

NIST's Modes of Operation for AES

- ECB/CBC/CFB/OFB + CTR (Dec 01)
- MAC algorithm: CMAC (May 05)
- **Authenticated encryption:**
  - CCM: CTR + CBC-MAC
  - GCM: Galois Counter Mode

Issues:
- associated data
- parallelizable
- on-line
- provable security

- IAPM
- XECB
- OCB

patented

- CCM
- GCM
- (EAX)
- (CWC)

Concrete recommendations

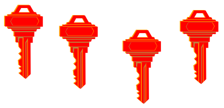
- AES-128 in CCM or EAX mode
  - CCM = CTR mode + CBC-MAC
  - change key after 2^{40} blocks
- Stream ciphers (better performance)
  - hardware: SNOW-3G or Trivium
  - software: HC-128
- CAESAR: open competition from 2013-2017 will come up with better solutions
  - http://competitions.cr.yp.to/caesar.html

Public-key cryptology

- the problem
- public-key encryption
- digital signatures
- an example: RSA
- advantages of public-key cryptology

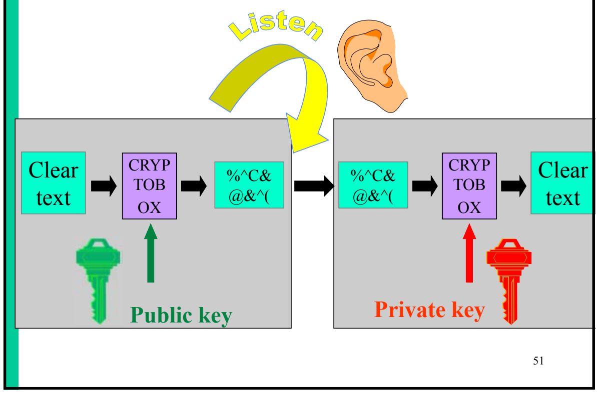
Limitation of symmetric cryptology

- Reduce security of information to security of keys

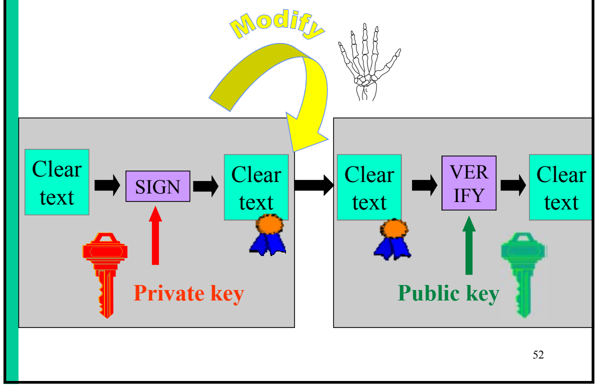


- But: how to establish these secret keys?
  - cumbersome and expensive
  - or risky: all keys in 1 place
- Do we really need to establish secret keys?

Public key cryptology: encryption



Public key cryptology: digital signature



A public-key agreement protocol: Diffie-Hellman

- Before: Alice and Bob have never met and share no secrets; they know a public system parameter α

generate x *compute α^x*	$\xrightarrow{\alpha^x}$ $\xleftarrow{\alpha^y}$	*generate y* *compute α^y*
compute $k=(\alpha^y)^x$		*compute $k=(\alpha^x)^y$*

- After: Alice and Bob share a short term key k
  - Eve cannot compute $\mathbf{k}$: in several mathematical structures it is hard to derive x from α^x (this is known as the discrete logarithm problem)

RSA ('78)

- Choose 2 "large" prime numbers p and q
- modulus n = p.q
- compute λ(n) = lcm(p-1,q-1)
- choose e relatively prime w.r.t. λ(n)
- compute d = e^{-1} mod λ(n)

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

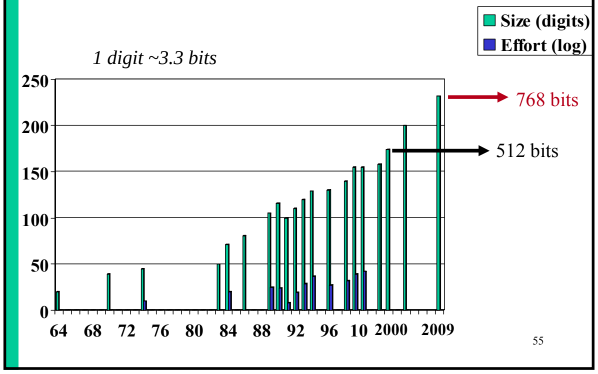
The security of RSA is based on the "fact" that it is easy to generate two large primes, but that it is hard to factor their product

- encryption: c = m^e mod n
- decryption: m = c^d mod n

try to factor 2419

Factorisation records

2009: 768 bits or 232 digits



55

Problematic public keys (1/3)

[Lenstra-Hughes+ Crypto 12]

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

- 1.1% of RSA keys occur in >1 certificate

- easy to factor: 0.2% of RSA keys
  - 12,000 keys!
  - 40% have valid certs

[Heninger+ Usenix Sec. 12]

12 million openly accessible public keys (5.8 TLS/6.2 SSH)
23 million hosts (12.8/10.2)

1%: 512-bit RSA keys

- 5.6% of TLS hosts share public keys
- 5.2% default manufacturer keys
- 0.34% have by accident the same key

- easy to factor: 0.5% of TLS hosts and 0.03% of SSH hosts
- DSA key recovery: 1.6% of DSA hosts

Problematic public keys (2/3)

- low entropy during key generation
- RSA keys easy to factor, because they form pairs like: n = p.q and n' = p'.q so gcd(n,n')=q
- DSA keys: reuse of randomness during signing or weak key generation

- why ???

- embedded systems
  - routers, server management cards, network security devices
- key generation at first boot

RSA versus DSA

Ron was wrong, Whit is right or vice versa?

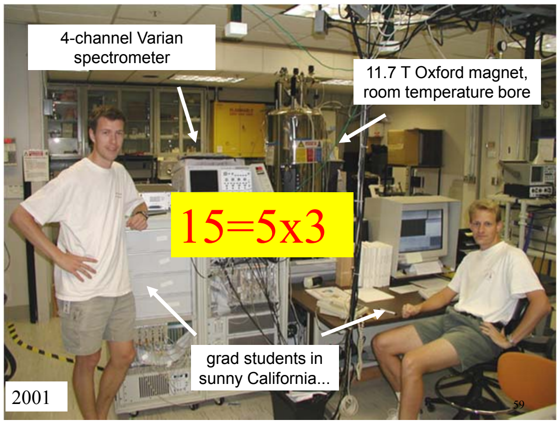
Problematic public keys (3/3)

ethical problem: how to report this?

details:

Lenstra, Hughes, Augier, Bos, Kleinjung, Wachter, "Ron was wrong, Whit is right" http://print.iacr.org/2012/064.pdf, *or with as title* "Public keys," Crypto 2012.

Heninger, Durumeric, Wustrow, Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices," Usenix Security 2012, https://www.usenix.org/conference/usenixsecurity12/tech-schedule/technical-sessions



4-channel Varian spectrometer

11.7 T Oxford magnet, room temperature bore

15=5x3

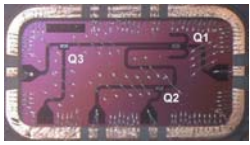
grad students in sunny California...

2001

59

- 2001: 7-bit quantum computer factors 15
- 2007: two new 7-bit quantum computers
- 2012: 143 has been factored in Apr. '12

- 2012: 10 to 15 years for a large quantum computer



Quantum Computing: An IBM Perspective

Steffen, M.; DiVincenzo, D. P.; Chow, J. M.; Theis, T. N.; Ketchen, M. B.

Quantum physics provides an intriguing basis for achieving computational power to address certain categories of mathematical problems that are completely intractable with machine computation as we know it today. We present a brief overview of the current theoretical and experimental works in the emerging field of quantum computing. The implementation of a functioning quantum computer poses tremendous scientific and technological challenges, but current rates of progress suggest that these challenges will be substantively addressed over the next ten years. We provide a sketch of a quantum computing system based on superconducting circuits, which are the current focus of our research. A realistic vision emerges concerning the form of a future scalable fault-tolerant quantum computer.

60

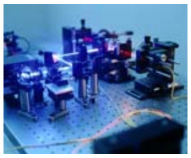
Quantum-computer resistant public key cryptology

- Error-correcting codes: McEliece
- Multivariate polynomial equations: HFE
- Lattices: NTRU
- Hash functions: Merkle scheme and variant for digital signatures

- So far it seems very hard to match performance of current systems while keeping the security level against conventional attacks

Quantum cryptography [BB84]

- 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 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 (e.g. side channels)

Advantages of public key cryptology

- Reduce protection of information to protection of authenticity of public keys
- Confidentiality without establishing secret keys
  - extremely useful in an open environment
- Data authentication without shared secret keys: digital signature
  - sender and receiver have different capability
  - third party can resolve dispute between sender and receiver

Disadvantages of public key cryptology

- Calculations in software or hardware two to three orders of magnitude slower than symmetric algorithms
- Longer keys: 1024 bits rather than 56…128 bits
- What if factoring is easy?

Reading material

- B. Preneel, Modern cryptology: an introduction.
  - This text corresponds more or less to this lecture
  - It covers in more detail how block ciphers are used in practice, and explains how DES works.
  - It does not cover identification, key management and application to network security.

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, online
- Other authors: Serge Vaudenay

Books on network security and more

- W. Stallings, *Network and Internetwork Security: Principles and Practice*, Prentice Hall, 5th Ed., 2010. Solid background on network security. Explains basic concepts of cryptography.
- W. Diffie, S. Landau, *Privacy on the line. The politics of wiretapping and encryption*, MIT Press, 2nd Ed., 2007. The best book so far on the intricate politics of the field.
- Ross Anderson, *Security Engineering*, Wiley, 2nd Ed., 2008. Insightful. A must read for every information security practitioner. Available for free at http://www.cl.cam.ac.uk/~rja14/book.html
- David Basin, Patrick Schaller, Michael Schläpfer, Applied Information Security. A Hands-on Approach, Springer-Verlag, 2011, 202 pages
- IACR (International Association for Cryptologic Research): www.iacr.org

Crypto software libraries

http://ece.gmu.edu/crypto_resources/web_resources/libraries.htm

C/C++/C#

- Botan (C++)
- Cryptlib (C)
- Crypto++ (C++)
- CyaSSL (C) embedded
- GnuTLS (C)
- Libgcrypt (C++)
- MatrixSSL (C++) embedded
- Miracl (binaries)
- OpenSSL (C++)
- PolarSSL (C)

Java

- SunJCA/JCE
- BouncyCastle (BC, C#)
- CryptixCrypto (until ’05)
- EspreSSL
- FlexiProvider
- GNU Crypto
- IAIK
- Java SSL
- RSA JSafe