

Introduction to Cryptology

Arjen K. Lenstra

Laboratory for cryptologic algorithms

What is Cryptology?

‘The art and science of secret writing’

~~What this talk could be:~~

~~How the basics of cryptology work~~

What this talk is:

How the basics of cryptology don't work

Context

Cryptology is crucial to achieve Information Security

Some other issues on which Information Security depends:

users, employees, passwords, confusion, lethargy, incompetence, stupidity, inertia, policies and their enforcement, regulations, legislation, jurisdiction, juries, monitoring, auditing, risk management, profits/losses, liabilities, business considerations, access control, verification, operating systems, implementation, software, patches, networks, legacy systems, errors, hackers, viruses, public relations, public perception, conventions, physical protection, standards, fear, ...

Why is cryptology interesting?

- Crypto: strongest link in information security
- Obviously: like to keep it that way
- But: many aspects we have **no clue** about!

Interesting because:

- Lots of challenging problems with impact on real life
- Covers broad range of mathematics and computer science
- Nothing can be taken for granted: lots of surprises

Examples of current cluelessness

1. The current hashing nightmare
2. The case of the Advanced Encryption Standard
3. Public Key Crypto: mathematics or religion?
4. Cryptography related products

The current hashing nightmare

‘Hashing’:

- A way to quickly, uniquely identify a document
- Comparable to a fingerprint: of fixed small size
- Tiny change in document leads to a completely different hash

The current hashing nightmare

‘Hashing’:

- A way to quickly, uniquely identify a document
- Comparable to a fingerprint: of fixed small size
- Tiny change in document leads to a completely different hash
- Lots of other nice properties:
 - Given the hash, can’t construct the document
 - Can’t make two documents with same hash
 - ...

Aside: hash versus encryption

Hashing and encrypting are totally different things:

- Hashing a document of any size:
 - always results in a fingerprint of the same small size
 - fingerprint cannot be used to reconstruct document
- Encrypting a document:
 - results in an encryption of about the same size
 - encryption used to reconstruct original document

Aside: hash versus encryption

Hashing and encrypting are totally different things:

- Hashing a document of any size:
 - always results in a fingerprint of the same small size
 - fingerprint cannot be used to reconstruct document
- Encrypting a document:
 - results in an encryption of about the same size
 - encryption used to reconstruct original document

So, what are hashes good for?

to identify data/docs/software succinctly

‘Popular’ hashes

Popular?

- Early 1990s:
 - MD4
 - MD5

Both by
Ron Rivest



- Mid 1990s:
 - SHA
 - SHA1

Both by
NSA, based
on MD4/MD5



Relevant related events

- Almost right away: MD4 considered weak, not used
- mid 1990s: MD5 ‘suspicious’, but widely used
- SHA mysteriously updated to SHA1
- Everyone happy with SHA1 (and some with MD5)
- 2002: announcement of SHA2, extension of SHA1

Relevant related events

- Almost right away: MD4 considered weak, not used
- mid 1990s: MD5 ‘suspicious’, but widely used
- SHA mysteriously updated to SHA1
- Everyone happy with SHA1 (and some with MD5)
- 2002: announcement of SHA2, extension of SHA1
- Fall of 2004:
 - MD4 disastrously weak
 - MD5 very weak
 - SHA weak
- February 7 '05, NIST: don't worry, SHA1&2 are fine!



(US) National
Institute of
Standards and
Technology

Relevant related events

- Almost right away: MD4 considered weak, not used
- mid 1990s: MD5 ‘suspicious’, but widely used
- SHA mysteriously updated to SHA1
- Everyone happy with SHA1 (and some with MD5)
- 2002: announcement of SHA2, extension of SHA1
- Fall of 2004:
 - MD4 disastrously weak
 - MD5 very weak
 - SHA weak
- February 7 '05, NIST: don't worry, SHA1&2 are fine!
- February 14 '05: **SHA1 weaker than expected**



(US) National
Institute of
Standards and
Technology

What happened?

2004/2005:

Xiaoyun Wang

‘broke’ almost
all hashes in sight



(and, strangely, all cryptologists loved it!)

Something weird in cryptology

- Why did Xiaoyun Wang break all our hashes?
- Shouldn't she be locked up?

Something weird in cryptology

- Why did Xiaoyun Wang break all our hashes?
- Shouldn't she be locked up?

If, in crypto, you manage to destroy others' toys:

- (research-)people love and appreciate it
- it's a sign of progress
(even if results may be disastrous)

The 'after Wang' era

- SHA1 definitely on the way out
- SHA2 no longer fully trusted either

But: SHA1 and SHA2 are essentially all we have

- Good hashes are crucial for applications
- At this point no one has a clue what to do

This just in from NIST

March 15, 2006: The SHA-2 family of hash functions (i.e., SHA-224, SHA-256, SHA-384 and SHA-512) may be used by Federal agencies for all applications using secure hash algorithms. Federal agencies should stop using SHA-1 for digital signatures, digital time stamping and other applications that require collision resistance as soon as practical, and must use the SHA-2 family of hash functions for these applications after 2010.

The case of AES

AES (Advanced Encryption Standard):

Intermezzo

What is an Encryption Standard supposed to do?

- Communicating parties A and B share a key K
- A uses K to quickly encrypt any volume of data
- The encrypted data is sent over public channels
- Only B can decrypt it and retrieve the data

(Most likely you've used it often)

The case of AES

AES (Advanced Encryption Standard):

The case of AES

AES (Advanced Encryption Standard):

The successor of DES (Data Encryption Standard)

DES:

- Designed in the mid 1970s, mostly by the NSA
- Regarded with utmost suspicion for a long time
- Still ‘unbroken’, but by late 1990s too weak
due to increasing computer speed

Finding a successor for DES

1997, NIST opted for open public design competition:

- Free exploitation of public know-how
- Avoid suspicion about cooked design

Finding a successor for DES

1997, NIST opted for open public design competition:

- Free exploitation of public know-how
- Avoid suspicion about cooked design

This turned out to be very successful approach

- At least 20 proposals from researchers worldwide
- Proposals presented to each other – some cracked
- Resulted in 5 finalists in 2000

The five finalists

- **MARS: IBM team** with Don Coppersmith 
- **RC6: RSA team** with Ron Rivest 
- **Rijndael: BE team** Vincent Rijmen & Joan Daemen 
- **Serpent: DK/IL/UK team** with Eli Biham 
- **Twofish: private US team** with Bruce Schneier 

And the winner was...

Rijndael, the one no one can pronounce

(other names considered: 'koeieulier' and 'angstschreeuw')



- Soon 'all' our communications will be protected by a Belgian cipher
- Let's keep our fingers crossed that AES = Rijndael is indeed as strong as we hope it to be

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

How infeasible?

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

How infeasible? Effort 2^{128} , that's more than 3×10^{38}

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

How infeasible? Effort 2^{128} , that's more than 3×10^{38}

Why would that be hard?

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

How infeasible? Effort 2^{128} , that's more than 3×10^{38}

Why would that be hard?

PCs run at 4GHz, say 1000GHz: 10^{12} ops/sec

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

How infeasible? Effort 2^{128} , that's more than 3×10^{38}

Why would that be hard?

PCs run at 4GHz, say 1000GHz: 10^{12} ops/sec

fewer than 3×10^7 sec/year: 3×10^{19} ops/year

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

How infeasible? Effort 2^{128} , that's more than 3×10^{38}

Why would that be hard?

PCs run at 4GHz, say 1000GHz:	10^{12} ops/sec
fewer than 3×10^7 sec/year:	3×10^{19} ops/year
10^{10} people, each 1000 PCs:	3×10^{32} ops/year

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

How infeasible? Effort 2^{128} , that's more than 3×10^{38}

Why would that be hard?

PCs run at 4GHz, say 1000GHz: 10^{12} ops/sec

fewer than 3×10^7 sec/year: 3×10^{19} ops/year

10^{10} people, each 1000 PCs: 3×10^{32} ops/year

If we all can afford the electric bill: a million years

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

But

cache

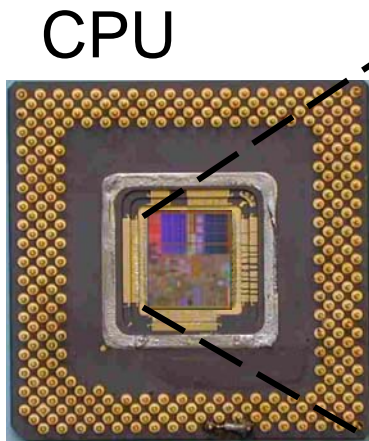
(slide shamelessly stolen from Eran Tromer)



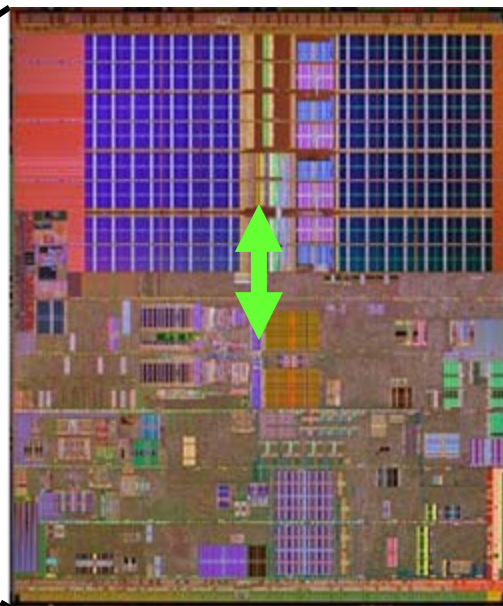
Main memory

(7-9% latency decrease per year)

Typical latency: 50-150ns



CPU



CPU cache memory

Typical latency: 0.3ns



CPU core

(60% speed increase per year)

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

But:

- New type of attack looks at cache behavior
- AES surprisingly susceptible:
if attacker can access machine where AES runs,
secret key retrieved in a **fraction of a second**.
- What now?

Cryptanalytic progress against AES?

No effective breaks affecting the AES algorithm yet:
finding a secret key is computationally infeasible

But:

- New type of attack looks at cache behavior
- AES surprisingly susceptible:
if attacker can access machine where AES runs,
secret key retrieved in a **fraction of a second**.
- What now? **Would NSA still select Rijndael?**

PKC: math or religion?

AES: How to arrange a common key for A and B ?

Traditionally: key management nightmare

PKC: math or religion?

AES: How to arrange a common key for A and B ?

Traditionally: key management nightmare

Since 1970 we use **public key cryptography (PKC)**

- An entirely new way of doing cryptography that revolutionized the field
- Based on **(trapdoor) one way functions**

Trapdoor one way functions?

- No one knows for sure how to construct them
- We hope/believe/pray we have some:
 - Integer factorization
Idea: **multiplication of integers is easy, but factoring is hard**
 - Discrete Logarithms
 - Lattice based, ...

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- What about 91?

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$
- What about 5283065753709209?

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$
- $5283065753709209 = 59957 \times 88114244437$
found in 25 minutes
in 1930s using the
bicycle chain sieve



Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$
- $5283065753709209 = 59957 \times 88114244437$
- What about $2^{128}+1$, a 39-digit number?

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$.
- $5283065753709209 = 59957 \times 88114244437$
- $2^{128}+1$ factor: 59649589127497217
found in 1970 in a few hours on an IBM 360/91

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$.
- $5283065753709209 = 59957 \times 88114244437$
- $2^{128}+1$ factor: 59649589127497217
- What about a 100-digit number?

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$.
- $5283065753709209 = 59957 \times 88114244437$
- $2^{128}+1$ factor: 59649589127497217
- 1988: first hard 100-digit number factorization, in two weeks on the Internet, generated lots of publicity

Why is factoring integers hard?

- Obviously false: given 15, easy to find 3 or 5
- $91 = 7 \times 13$.
- $5283065753709209 = 59957 \times 88114244437$
- $2^{128}+1$ factor: 59649589127497217
- 1988: first hard 100-digit number factorization
- Current state of the art:
 - 200-digit numbers take months on huge networks
 - 300-digit numbers are safely out of reach

But why is factoring integers hard?

- No one knows
- Maybe factoring is not hard at all
(and: factoring is easy on quantum computer!)
- Hardly any progress since the late 1980s
1989: Number Field Sieve made it less hard
- All we do now is throw more hardware at it
- We need more theory, but we have no clue...

Cryptography related products

- Lots of very crappy crypto products for sale
- No one seems to care much:
 - it is still better than having nothing at all
 - even though they give a false sense of protection

Example of crappy product

- A one-time password generator
- Generates a new unpredictable password
- For additional protection during authentication



E9AE88 E8A7F9 4EPEHP C288C0 48P064 C48682

Example of crappy product

- A one-time password generator
- Generates a new unpredictable password
- For additional protection during authentication



E9AE88 E8A7F9 4EPEHP C288C0 48P064 C48682

**This turns out not to be a coincidence:
After more outputs only 2 significant digits**

Conclusion

Looked at the basic ingredients of cryptology

- Hashing
- AES
- Public Key Crypto

Conclusion

Looked at the basic ingredients of cryptology

- Hashing: **no one knows what is going on**
- AES: **new attack model: unpleasant surprise**
- Public Key Crypto: **situation stable and unclear**

Conclusion

Looked at the basic ingredients of cryptology

- Hashing: **no one knows what is going on**
- AES: **new attack model: unpleasant surprise**
- Public Key Crypto: **situation stable and unclear**

Help is desperately needed!

