Analysis and design of symmetric cryptographic algorithms

Jean-Philippe Aumasson

University of Applied Sciences Northwestern Switzerland
School of Engineering

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

PhD public defense

Agenda

Introduction

- Cryptography and its applications
- Symmetric cryptographic algorithms
- What is a cryptographic attack?

This thesis

- Context
- Attacks on ciphers
- Attacks on hash functions
- New design of hash function

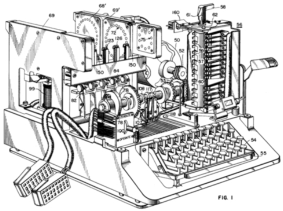
Conclusion

Introduction

Cryptography

"Science of secret"

Historical application: **encryption**



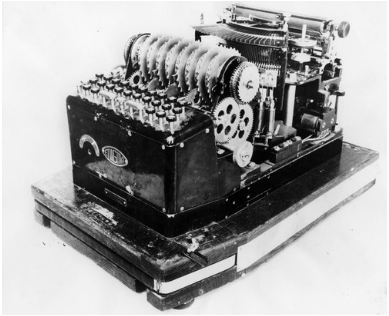
$\longleftarrow$ CONFIDENTIAL MESSAGE

$\longrightarrow$ K8DZQB1PAHYIGLETMD5T

Need to know the **secret key** to encrypt/decrypt

Crypto in practice

The **Enigma** machine (1920's)



Used by German army during WWII…
…broken by British intelligence

Modern crypto: different machines, more applications…

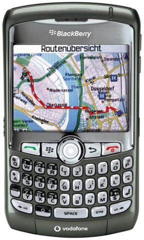
Secure communication

Goals

- Message privacy
- Sender & recipient authentication
- Non-repudiation

Tools

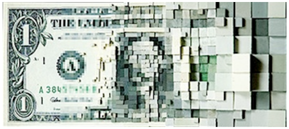
- Symmetric crypto
- Public-key crypto
- Key-agreement protocols
- Digital signatures
- Certificates



Digital money

Goals

- Anonymity
- Fairness
- Untraceability
- Transferability
- etc.



Tools

- Number theory mathematics
- Zero-knowledge protocols
- Secure hardware tokens

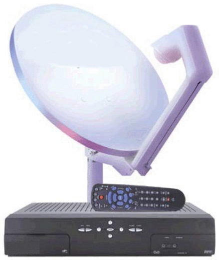
Conditional access TV

Goals

- Broadcast operation (satellite, etc.)
- Message privacy
- Selective reception

Tools

- Symmetric crypto
- Public-key crypto
- Secure hardware



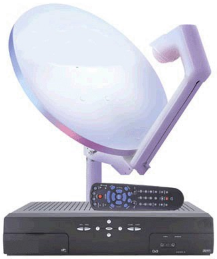
Conditional access TV

Goals

- Broadcast operation (satellite, etc.)
- Message privacy
- Selective reception

Tools

- Symmetric crypto
  - Ciphers
  - Hash functions
- Public-key crypto
- Secure hardware



Ciphers

Transform a **plaintext** to a **ciphertext**

Key K necessary to encrypt and to decrypt

$$\texttt{MESSAGE} \xrightarrow{\text{Encrypt}_K} \texttt{LSJFSDH}$$

$$\texttt{LSJFSDH} \xrightarrow{\text{Decrypt}_K} \texttt{MESSAGE}$$

Ciphers

Transform a **plaintext** to a **ciphertext**

Key *K* necessary to encrypt and to decrypt

$$\texttt{MESSAGE} \xrightarrow{\text{Encrypt}_K} \texttt{LSJFSDH}$$

$$\texttt{LSJFSDH} \xrightarrow{\text{Decrypt}_K} \texttt{MESSAGE}$$

$$\text{“meaningful text”} \xrightarrow{\text{Encrypt}_K} \text{“unreadable text”}$$

Cipher by substitution

Replace `A` by `D`, `B` by `E`, `C` by `F`, etc. Ex: `EPFL` $\longrightarrow$ `HSIO`

Cipher by substitution

Replace `A` by `D`, `B` by `E`, `C` by `F`, etc. Ex: $\texttt{EPFL} \longrightarrow \texttt{HSIO}$

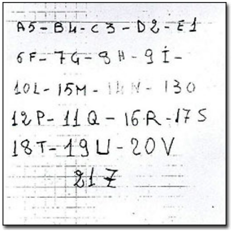
Used by Julius Caesar in -50…

Cipher by substitution

Replace `A` by `D`, `B` by `E`, `C` by `F`, etc. Ex: `EPFL` $\longrightarrow$ `HSIO`

Used by Julius Caesar in -50…

…and Sicilian Mafia bosses in 2000's



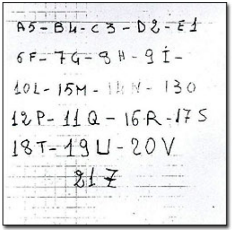
A5 - B4 - C3 - D2 - E1
6F - 7G - 8H - 9I -
10L - 15M - 14N - 13O
12P - 11Q - 16R - 17S
18T - 19U - 20V
21Z

Cipher by substitution

Replace `A` by `D`, `B` by `E`, `C` by `F`, etc. Ex: `EPFL` $\longrightarrow$ `HSIO`

Used by Julius Caesar in -50…

…and Sicilian Mafia bosses in 2000's (with less success)

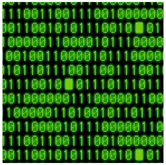


A5 - B4 - C3 - D2 - E1
6F - 7G - 8H - 9I -
10L - 15M - 14N - 13O
12P - 11Q - 16R - 17S
18T - 19U - 20V
21Z

Modern cryptography

- Uses **computers**, not pencil-and-paper

- Operates on **bits**, not on letters



- Is **hard to break** (in general)

Perfect cipher

$$\begin{array}{ll} \text{Plaintext:} & \texttt{0111011}\cdots\texttt{0101011} \\ & \quad\quad\quad\oplus \\ \text{Key:} & \texttt{1101011}\cdots\texttt{1001101} \\ & \quad\quad\quad= \\ \text{Ciphertext:} & \texttt{1010000}\cdots\texttt{1100110} \end{array}$$

"XOR" operation on bits

$$1 \oplus 1 = 0 \quad 0 \oplus 0 = 0 \quad 1 \oplus 0 = 1 \quad 0 \oplus 1 = 1$$

Perfect cipher

$$\begin{array}{ll} \text{Plaintext:} & \texttt{0111011}\cdots\texttt{0101011} \\ & \qquad\quad\;\oplus \\ \text{Key:} & \texttt{1101011}\cdots\texttt{1001101} \\ & \qquad\quad\;= \\ \text{Ciphertext:} & \texttt{1010000}\cdots\texttt{1100110} \end{array}$$

“XOR” operation on bits

$$1 \oplus 1 = 0 \quad 0 \oplus 0 = 0 \quad 1 \oplus 0 = 1 \quad 0 \oplus 1 = 1$$

Used during the cold war to encrypt the
Moscow-Washington telescripter liaison

Problem: the key must be as long as the plaintext

Solution: stream ciphers

Generate a long string of bits from a short key

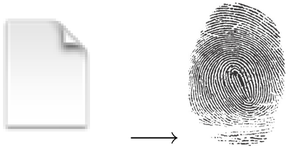
$$\begin{array}{lll} & \text{Plaintext:} & \texttt{0111011}\cdots\texttt{0101011} \\ & & \qquad\quad\oplus \\ \textbf{Key}\text{: } \texttt{101011} \longrightarrow & \textbf{Keystream}\text{:} & \texttt{1101011}\cdots\texttt{1001101} \\ & & \qquad\quad= \\ & \text{Ciphertext:} & \texttt{1010000}\cdots\texttt{1100110} \end{array}$$

Expands a short bit string to a long one

If the key of 128 bits, there are 2^{128} possible keystreams
$\Rightarrow$ ideally, an attack makes 2^{128} trials

Hash functions

Compresses a long bit string to a short one



Message: `0100···1101` $\longrightarrow$ **Hash**: `110101`

$$x \longrightarrow H(x)$$

Main application: **digital signatures**
(signing short documents is cheaper than long ones)

Hash functions security: preimage resistance

Given a hash y, it should be **difficult** to find x such that

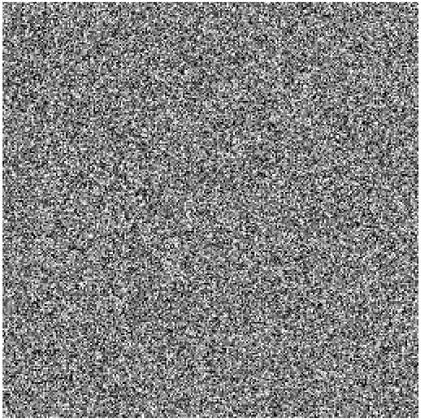
$$H(x) = y$$

Hash functions security: collision resistance

It should be difficult to find x_1 and x_2 such that

$$H(x_1) = H(x_2), \quad x_1 \neq x_2$$

Hash functions security: randomness



The hashes should look like random values

What is an attack?

Any **mathematical method** that finds either…

- the secret key (stream ciphers)
- preimages or collisions (hash functions)
- non-randomness

in less time than ideally expected

Bruteforce: works against any cipher or hash function

What is an attack?

128-bit keys are typical: finding the secret key should require 2^{128} operations

$2^{128} \approx 10^{38} = 100\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000$

Using 7 000 000 000 computers at 4 GHz in parallel:

It would take $10^{11.6}$ years to find the key
$\approx$ 28 times the age of the universe

What is an attack?

128-bit keys are typical: finding the secret key should require 2^{128} operations

$2^{128} \approx 10^{38} = 100\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000\,000$

Using 7 000 000 000 computers at 4 GHz in parallel:

It would take $10^{11.6}$ years to find the key
$\approx$ 28 times the age of the universe

Suppose we find an attack in 2^{120} only
It would only take 1 billion years!

What is an attack?

128-bit keys are typical: finding the secret key should require 2^{128} operations

$2^{128} \approx 10^{38} =$ 100 000 000 000 000 000 000 000 000 000 000 000 000

Using 7 000 000 000 computers at 4 GHz in parallel:

It would take $10^{11.6}$ years to find the key
$\approx$ 28 times the age of the universe

Suppose we find an attack in 2^{120} only
It would only take 1 billion years!

The cipher is **considered broken**

This thesis

Context: crypto public competitions

1. Cryptographers submit algorithms
2. They try to destroy competitors
3. The organizer picks a design that survived

Context: crypto public competitions

eSTREAM (2005-08)

http://www.ecrypt.eu.org/ecrypt1/

ECRYPT

Network of Excellence in Cryptology

IST-2002-507932

PROJECT
Home
Partners

eSTREAM - the ECRYPT Stream Cipher Project

- European Network of Excellence (EPFL, CNRS, etc.)
- New stream ciphers: Salsa20, Grain, etc.

SHA-3 Competition (2008-12)

http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

NIST National Institute of Standards and Technology
Information Technology Laboratory

SEARCH CSRC: GO

ABOUT MISSION CONTACT STAFF SITE MAP

Computer Security Division CSD
Computer Security Resource Center CSRC

CSRC HOME GROUPS PUBLICATIONS DRIVERS NEWS & EVENTS ARCHIVE

Cryptographic Hash Project
Cryptographic Hash Algorithm Competition
Timeline for Hash Algorithm Competition

CSRC HOME > GROUPS > ST > HASH PROJECT

CRYPTOGRAPHIC HASH ALGORITHM COMPETITION

NIST has opened a public competition to develop a new cryptographic hash algorithm, which converts a variable length message into a short "message digest" that can be used for digital signatures, message authentication and

- US Institute of Standards (NIST)
- Future hash function standard SHA-3

The stream cipher Salsa20

Operations on 32-bit words:

- **XOR**: words viewed as strings of bits

$$\mathtt{0101}\cdots\mathtt{0101} \oplus \mathtt{1010}\cdots\mathtt{1010} = \mathtt{1111}\cdots\mathtt{1111}$$

- **Rotation**: words viewed as strings of bits

$$\mathtt{1000}\cdots\mathtt{0000} \lll 1 = \mathtt{0000}\cdots\mathtt{0001}$$

- **Integer addition**: words viewed as integer numbers

$$\begin{aligned} 1000 + 1 &\equiv 1001 \bmod 2^{32} \\ (2^{32} - 1) + 1 &\equiv 0 \bmod 2^{32} \end{aligned}$$

The stream cipher Salsa20

1. Initialize a table of 32-bit words with 256 key bits

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix}$$

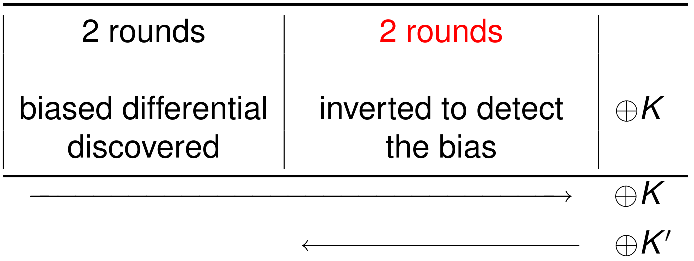
2. Repeat 10 times (rounds)

$$\begin{array}{llll}
x_4 \oplus= (x_0 + x_{12}) \lll 7\,; & x_9 \oplus= (x_5 + x_1) \lll 7\,; & x_{14} \oplus= (x_{10} + x_6) \lll 7\,; & x_3 \oplus= (x_{15} + x_{11}) \lll 7 \\
x_8 \oplus= (x_4 + x_0) \lll 9\,; & x_{13} \oplus= (x_9 + x_5) \lll 9\,; & x_2 \oplus= (x_{14} + x_{10}) \lll 9\,; & x_7 \oplus= (x_3 + x_{15}) \lll 9 \\
x_{12} \oplus= (x_8 + x_4) \lll 13; & x_1 \oplus= (x_{13} + x_9) \lll 13; & x_6 \oplus= (x_2 + x_{14}) \lll 13; & x_{11} \oplus= (x_7 + x_3) \lll 13 \\
x_0 \oplus= (x_{12} + x_8) \lll 18; & x_5 \oplus= (x_1 + x_{13}) \lll 18; & x_{10} \oplus= (x_6 + x_2) \lll 18; & x_{15} \oplus= (x_{11} + x_7) \lll 18 \\
x_1 \oplus= (x_0 + x_3) \lll 7\,; & x_6 \oplus= (x_5 + x_4) \lll 7\,; & x_{11} \oplus= (x_{10} + x_9) \lll 7\,; & x_{12} \oplus= (x_{15} + x_{14}) \lll 7 \\
x_2 \oplus= (x_1 + x_0) \lll 9\,; & x_7 \oplus= (x_6 + x_5) \lll 9\,; & x_8 \oplus= (x_{11} + x_{10}) \lll 9\,; & x_{13} \oplus= (x_{12} + x_{15}) \lll 9 \\
x_3 \oplus= (x_2 + x_1) \lll 13; & x_4 \oplus= (x_7 + x_6) \lll 13; & x_9 \oplus= (x_8 + x_{11}) \lll 13; & x_{14} \oplus= (x_{13} + x_{12}) \lll 13 \\
x_0 \oplus= (x_3 + x_2) \lll 18; & x_5 \oplus= (x_4 + x_7) \lll 18; & x_{10} \oplus= (x_9 + x_8) \lll 18; & x_{15} \oplus= (x_{14} + x_{13}) \lll 18
\end{array}$$

3. Add the initial table to the final table

Attack strategy for 4 rounds

Structure permut(K) $\oplus$ K, with K a 256-bit key



Invert 2 rounds to observe the bias, based on partial knowledge of the key

Can be used to verify the correctness of 220 key bits

$\Rightarrow$ can find the key 64 times faster than ideally

Summary and impact

We developed the **best known attacks** on the stream cipher Salsa20

Contributes to increase the confidence in the cipher

Salsa20 chosen as

- Cowinner of the eSTREAM competition
- Alternative to AES by programmers
- Basis for a new hash function…

New differential attacks

Differential equations play an important role in. . .

- Quantum mechanics (evolution of a quantum state)

$$i\hbar\partial_t|\psi\rangle = H|\psi\rangle$$

- Image processing (PDE-based techniques)

$$\frac{\partial I}{\partial t} = \mathrm{div}\,(c(x,y,t)\nabla I) = \nabla c \cdot \nabla I + c(x,y,t)\Delta I$$

- Economics (evolution of stock prices)

$$dS_t = \mu S_t\,dt + \sigma S_t\,dW_t$$

- Cryptography (differential attacks)

$$\hat{E}_k(m) \oplus \hat{E}_k(m \oplus \hat{\Delta}_{\text{in}}) = \hat{\Delta}_{\text{out}}$$

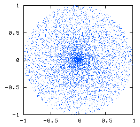
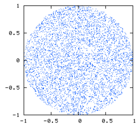
New attacks: cube testers

View the cipher as a **black-box**

Differentiate its implicit Boolean equations

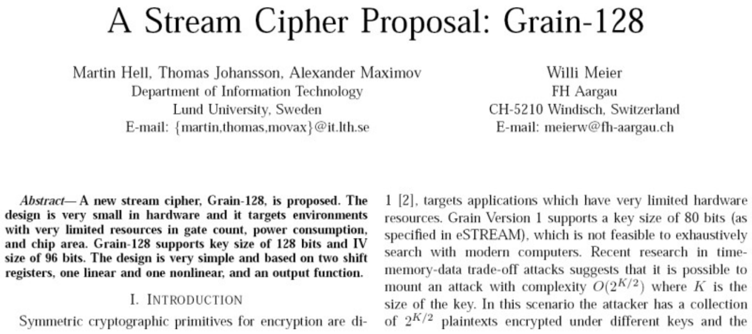
$$\bigoplus_{i=0}^{2^n-1} f_k(v_i)$$

Try to **detect a structure** in the differential equations

Ex:  vs. 

Application of cube testers

Goal: attacking the stream cipher Grain-128



A Stream Cipher Proposal: Grain-128

Martin Hell, Thomas Johansson, Alexander Maximov
Department of Information Technology
Lund University, Sweden
E-mail: {martin,thomas,movax}@it.lth.se

Willi Meier
FH Aargau
CH-5210 Windisch, Switzerland
E-mail: meierw@fh-aargau.ch

Abstract— **A new stream cipher, Grain-128, is proposed. The design is very small in hardware and it targets environments with very limited resources in gate count, power consumption, and chip area. Grain-128 supports key size of 128 bits and IV size of 96 bits. The design is very simple and based on two shift registers, one linear and one nonlinear, and an output function.**

I. Introduction

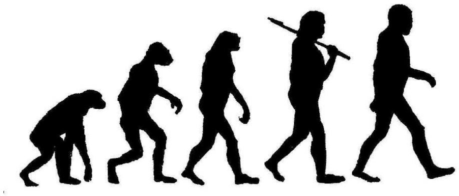
Symmetric cryptographic primitives for encryption are di- 1 [2], targets applications which have very limited hardware resources. Grain Version 1 supports a key size of 80 bits (as specified in eSTREAM), which is not feasible to exhaustively search with modern computers. Recent research in time-memory-data trade-off attacks suggests that it is possible to mount an attack with complexity $O(2^{K/2})$ where K is the size of the key. In this scenario the attacker has a collection of $2^{K/2}$ plaintexts encrypted under different keys and the

Application of cube testers

Implementation in programmable hardware

Application of cube testers

Optimize parameters with **evolutionary methods**



```
for(i=0;i<NUMBER_GENERATIONS;++i) {

  for(j=0;j<CHILDREN;++j) reproduction( rand()%POPULATION, rand()%POPULATION, j );
  for(j=0;j<POPULATION+CHILDREN;++j) perf[j] = (evaluate( j )<<8)^j;
  for(j=0;j<POPULATION+CHILDREN;++j)
    for(k=0;k<CUBE_SIZE;++k) buffer[j][k] = population[j][k];
  qsort( perf, POPULATION+CHILDREN, sizeof(int), compare );
  for(j=0;j<POPULATION;++j)
    for(k=0;k<CUBE_SIZE;++k) population[ j ][ k ] = buffer[ perf[j]&0xFF ][k];

}
```

Summary and impact

We showed that Grain-128 can be **broken**

- in time 2^{77} (1h30 with 7 000 000 000 computers)
- instead of 2^{128} (28 universe lifetimes)

Unexpected result!

Grain-128 should not be used (anymore)

Attacks on hash functions

Given the algorithm of $H()$

Preimage attack

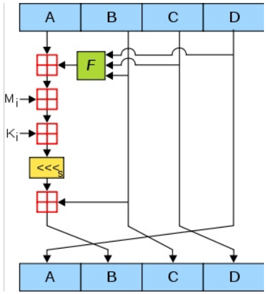
Given y,
find x such that $H(x) = y$

Collision attack

Find x_1, x_2
such that $H(x_1) = H(x_2)$

Preimage attacks on reduced MD5

MD5

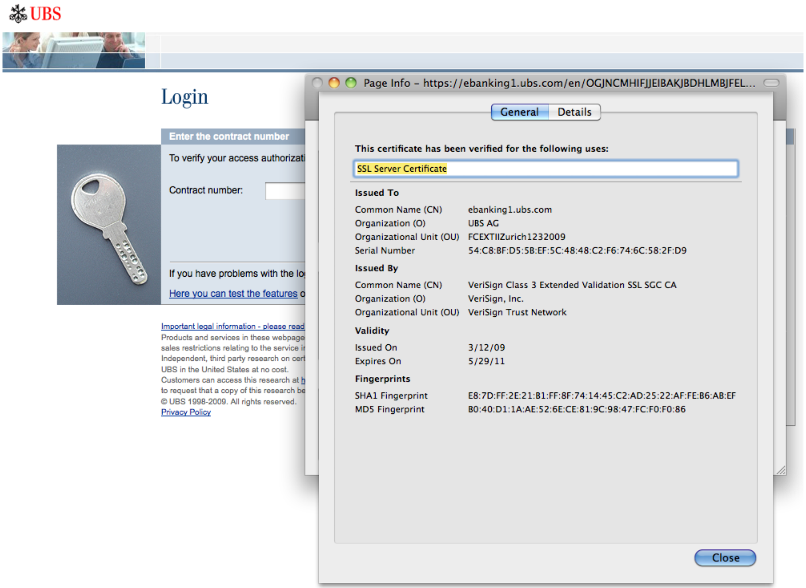


THE most ever studied hash function

Internet standard, designed in 1992

Collision attacks found in 2005

No preimage attack known



UBS

Login

Enter the contract number

To verify your access authorizati

Contract number:

If you have problems with the lo

Here you can test the features o

Important legal information - please read
Products and services in these webpage
sales restrictions relating to the service i
Independent, third party research on cert
UBS in the United States at no cost.
Customers can access this research at h
to request that a copy of this research be
© UBS 1998-2009. All rights reserved.
Privacy Policy

Page Info – https://ebanking1.ubs.com/en/OGJNCMHIFJJEIBAKJBDHLMBJFEL...

General | Details

This certificate has been verified for the following uses:

SSL Server Certificate

Issued To

Common Name (CN)	ebanking1.ubs.com
Organization (O)	UBS AG
Organizational Unit (OU)	FCEXTIIZurich1232009
Serial Number	54:C8:BF:D5:5B:EF:5C:48:48:C2:F6:74:6C:58:2F:D9

Issued By

Common Name (CN)	VeriSign Class 3 Extended Validation SSL SGC CA
Organization (O)	VeriSign, Inc.
Organizational Unit (OU)	VeriSign Trust Network

Validity

Issued On	3/12/09
Expires On	5/29/11

Fingerprints

SHA1 Fingerprint	E8:7D:FF:2E:21:B1:FF:8F:74:14:45:C2:AD:25:22:AF:FE:B6:AB:EF
MD5 Fingerprint	B0:40:D1:1A:AE:52:6E:CE:81:9C:98:47:FC:F0:F0:86

Close

```
//Note: All variables are unsigned 32 bits and wrap modulo 2^32 when calculating
var int[64] r, k

//r specifies the per-round shift amounts
r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}

//Use binary integer part of the sines of integers (Radians) as constants:
for i from 0 to 63
    k[i] := floor(abs(sin(i + 1)) × (2 pow 32))

//Initialize variables:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476

//Pre-processing:
append "1" bit to message
append "0" bits until message length in bits ≡ 448 (mod 512)
append bit /* bit, not byte */ length of unpadded message as 64-bit little-endian integer to message

//Process the message in successive 512-bit chunks:
for each 512-bit chunk of message
    break chunk into sixteen 32-bit little-endian words w[i], 0 ≤ i ≤ 15

    //Initialize hash value for this chunk:
    var int a := h0
    var int b := h1
    var int c := h2
    var int d := h3

    //Main loop:
    for i from 0 to 63
        if 0 ≤ i ≤ 15 then
            f := (b and c) or ((not b) and d)
            g := i
        else if 16 ≤ i ≤ 31
            f := (d and b) or ((not d) and c)
            g := (5×i + 1) mod 16
        else if 32 ≤ i ≤ 47
            f := b xor c xor d
            g := (3×i + 5) mod 16
        else if 48 ≤ i ≤ 63
            f := c xor (b or (not d))
            g := (7×i) mod 16

        temp := d
        d := c
        c := b
        b := b + leftrotate((a + f + k[i] + w[g]) , r[i])
        a := temp

    //Add this chunk's hash to result so far:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d

var int digest := h0 append h1 append h2 append h3 //(expressed as little-endian)
```

Attack strategy: birthday paradox

In a group of at least 23 random people, there is more than 50% probability that some pair of them will have the same birthday

Idea: $(23 \times 22)/2 = 253$ possible pairs…

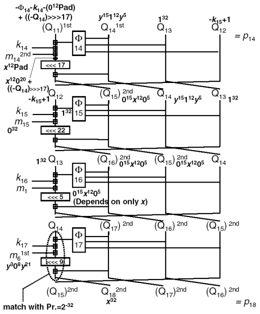
Summary and impact

First preimage attacks on (reduced) MD5

Introduce new cryptanalysis techniques

- Neutral words
- Local collisions

Techniques generalized and refined to attack full MD5



Other attacks found

Hash name	Type of attack	Broken
CHI	collision	☠
Codefish	preimage	☠
Dynamic SHA2	collision	☠
ESSENCE	collision	☠
Hamsi	dist.	?
HAVAL	preimage	☠
MCSSHA	preimage	☠
Shabal	dist.	?
Skein	dist.	?
Vortex	collision	☠

Details on the SHA-hash competition

- **64** designs submitted (academia, Intel, IBM, etc.)
- **51** selected for the “first round” (Dec 08)
- **14** selected for the “second round” (Jul 09)
- **5** finalists (2010)

The winner will be…

- A worldwide standard
- Implemented in all computers
- Supported for decades (ideally)

Our candidate: BLAKE

Design started in 2007, with as goals

- As simple as possible, but not simpler
- Stand on the shoulders of previous cryptographers
- Fast in software and hardware
- Secure against classical and quantum attacks

$\Rightarrow$ need good **tradeoff speed/security**

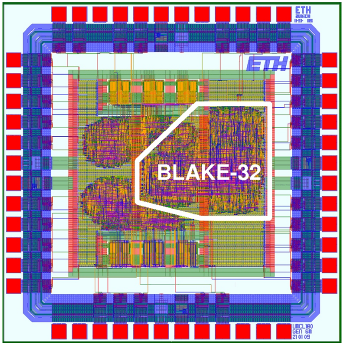
BLAKE's core algorithm

```
a += $m_i \oplus k_i$
a += b
d = $(a \oplus d) \ggg 16$
c += d
b = $(b \oplus c) \ggg 12$
a += $m_j \oplus k_j$
a += b
d = $(a \oplus d) \ggg 8$
c += d
b = $(b \oplus c) \ggg 7$
```

BLAKE's performance

Simple to implement, and fast on all platforms

- **450 Mb/sec** in a PC
- **20 Gb/sec** in integrated circuits



BLAKE in the SHA-3 competition

One of the 14 second round candidates

Researchers from Austria, Canada, Germany, Japan, Netherlands, and Portugal worked on efficient implementation of BLAKE (HW & SW)

Researchers from all over the world tried to attack BLAKE (without success so far)

“The best results against BLAKE (. . .) appear to pose no threat to the design” (NIST)

Final decision: 2012

Conclusion

Summary of contributions

New cryptanalytic techniques

Attacks on several ciphers and hash functions

Design of second round SHA-3 hash function candidate

Better understanding of symmetric crypto algorithms?

Dissemination of research results:

- Peer-reviewed articles
- Contributed talks in conferences
- Invited talks in seminars

Thanks to my co-authors from. . .

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

n|w
University of Applied Sciences Northwestern Switzerland
School of Engineering

Loughborough
University

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

Ens

METU

1425

DTU

TU
Graz

ingenico

INRIA

Thanks for your attention!

Questions?