CRYPTOGRAPHY LECTURE, 2023

Master level

Mirosław Kutyłowski

Cryptographic Hash Function

Ideal model: Random Oracle:

a blackbox D interacting with the external world:

- D maintains a hash table T
- D defines a function F and serves as an oracle for values of F
- query $F(a) = ?$ sent to D for a k-bit string a. It answers as follows:
  1. search for an entry $(a, *)$ in T
  2. if an entry (a, z) has been found, then return(z)
  3. else:
     i. choose z at random
     ii. insert (a, z) in T
     iii. return(z)

Properties of ROM: One-Way Function

input: z chosen at random

output: any x such that $F(x) = z$

F is **one-way function** if

there is no efficient algorithm to solve this problem with a non-negligible probability

Remarks:

i. if we have seen a such that the oracle D has returned z, then the algorithm can return a

ii. the number of arguments where the oracle D has answered is limited (say 2^{40}), while $k \gg 40$, then probability of this situation is negligible

(for $k = 160$: pbb=$1/2^{120}$)

ROM

at the moment of creation of the query $F^{-1}(z) = ?$ with high probability no entry f the form $(*, z)$ in T

an algorithm can not forsee the future events (coin tossing and choosing z as the image of F)

Application - cryptographic commitment

procedures:

commitment creation:

- input: x
- output: $C(x)$

commitment opening:

- input: c
- output: x such that $C(x) = c$

key property I:

for any nontrivial property $\mathcal{A}$ and a commitment c it is infeasible to say whether $\mathcal{A}(x)$ for x used to create the commitment ($c = C(x)$)

In other words: we know nothing about x if we learn only c

Target application of cryptographic commitment

interactive protocol between Alice and Bob

mimicking "simulataneous choice":

i. Alice chooses x, commits x to c, and shows c to Bob

ii. Bob chooses y

iii. Alice opens c to x

One more property needed to avoid cheating by Alice:

Alice may attempt to open c to a different value x

2nd pre-image resistance:

given x and $c = F(x)$ it is infeasible to find any x' such that $c = F(x')$

For ROM:

- it is extremely unlikely that for an entry $(x, c) \in T$ there is already an entry $(x', c) \in T$ for $x' \neq x$ ...
- ... so for producing such x' it is necessary to ask queries and hope to get c
- ... but this is extremely unlikely

weaker version: conflict-freeness :

given F

find and x, x' such that $F(x) = F(x')$

Existence issues

if $F: \{0,1\}^{2k} \longrightarrow \{0,1\}^{k}$ then the the expected number of x' such that $F(x) = F(x')$ is huge (2^k)! (Pigeon Hole Principle)

- deep difference between:

  "x' does exist" and "x' can be computed"

Example for 2nd preimage resistance without ROM:

Setup: random generators g and h of a cyclic group G of a prime order q such that it is infeasible to find r such that $g^r = h$

Function: $F: \{1, ..., q-1\}^2 \to G$ where

$$F(x_0, x_1) = g^{x_0} \cdot h^{x_1}$$

What if not 2nd preimage resistant:

i. algorithm A finds x_0' and x_1' such that $F(x_0, x_1) = F(x_0', x_1')$

then also $g^{x_0} \cdot h^{x_1} = g^{x_0'} \cdot h^{x_1'}$

ii. A returns $r := (x_0' - x_0)/(x_1 - x_1') \bmod q$

indeed

$$h = g^{(x_0' - x_0)/(x_1 - x_1')}$$

Application: coin tossing over internet

i. Alice chooses secret K and bit a

ii. Alice calculates: $c := F(K, a)$ and sends c to Bob

iii. Bob chooses bit b and sends it to Alice

iv. Alice computes $r := a \otimes b$ and responds with K, a

v. Bob checks the opening: $c = F(K, a)$? and computes $r := a \otimes b$

Dependencies:

conflict free $\Rightarrow$ 2nd preimage resistant

Equivalent to: $\neg$ 2nd preimage resistant $\Rightarrow \neg$ conflict free

this is obvious: if we can create a conflict for a $c = F(x)$then we can create a conflict!

2nd preimage resistant $\Rightarrow$ one-way

Equivalently: $\neg$ one-way $\Rightarrow$ $\neg$ 2nd preimage resistant

proof:

i. choose x at random

ii. $c := F(x)$

iii. apply inversion function: $x' := F^{-1}(c)$

iv. output (x, x')

v. with high probability $x \neq x'$ as there is a huge number of z such that $F(z) = c$

Correlated-input secure

input: given $c = F(K, x)$ for a known K and unknown random x, a "simple function" H

output: c' such that $c' = F(K, H(x))$

example: $H(x) = x + 1$

F is correlated-input secure if

if there is no computable function A solving this problem

ROM: it holds obviously

Yet-another-output secure

input: given $c_1 = F(K, H_1(x)) \ldots, c_k = F(K, H_k(x))$ for a known K, unknown random x, and known "simple circuits" $H_1, \ldots, H_k$

output: c' such that $c' = F(K, H_{k+1}(x))$

F secure if for any computable function A solving this problem:

either

- $H_{k+1}(x) = H_i(x)$ for some $i \leq n$
- or A can extract x

Application: pseudorandom number generation

Procedure:

i. choose K at random

ii. output: $F(K,0)\|F(K,1)\|F(K,2)\|...$

Property: unpredictability

Given a prefix of the output of such PRNG it is impossible to predict what will come next

(unless F is not one-way)

authenticating transmission over a second channel:

i. server A sends to server B a data D over an unprotected channel

ii. server A computes $h := F(D, K)$ where K is random and shared by A and B

iii. the operator of A calls the operator of B and dictates h

iv. server B checks whether h corresponds to the message received by recomputing it

An attack (for h transmitted in advance):

i. eavesdrop h

ii. as man-in-the-middle intercept D and manipulate getting D' such that $F(D', K) = h$ as well

iii. transmit D' to server B

(the attack does not work iff F has the property discussed)

Negative Example

$F(x_0, x_1) = g^{x_0} \cdot h^{x_1}$

- let $z = F(x_0, x_1)$ for unknown x_0 and x_1
- one can easily compute $F(C(x_0, i), C(x_1, j))$ where $C(x, i) = x + i \bmod q$

$$F(C(x_0, i), C(x_1, j)) = z \cdot g^i \cdot h^j$$

Lesson learnt: this F is **provably secure** concerning 2nd preimage resistance but ...

nevertheless **insecure** regarding another important property

Yet another property (for Blockchain)

input: M and a parameter a small integer k

task: find y such that k least significant bits of $F(M, y)$ are $0....0$

Assumption:

there is no better way to find y than brute force:

– take candidates for y, for each compute $F(M, y)$ and check the result

Mining in Bitcoin

the party that first finds y can extend the blockchain with transactions M (and get reward)

Hash functions for a fixed input length

say $H: \{0,1\}^{512} \longrightarrow \{0,1\}^{160}$

should behave as in ROM:

that is: **all properties mentioned so far should be satisfied**

the image must be long enough to withstand the **birthday attack** for $H: * \longrightarrow \{0,1\}^{2k}$:

i. choose at random $x_1, \ldots, x_m$ for $m = 2^k$

ii. compute $H(x_1), \ldots, H(x_m)$

iii. look for any repetition among the values computed

iv. if found: $H(x_i) = H(x_j)$ then return the collision values x_i and x_j

Observation: pbb of failure:

$$\left(1-\frac{1}{2^{2k}}\right)\left(1-\frac{2}{2^{2k}}\right)\cdots\left(1-\frac{2^k}{2^{2k}}\right)>\left(1-\frac{1}{2^k}\right)^{2^k}\approx\frac{1}{e}$$

$$\left(1-\frac{1}{2^{2k}}\right)\left(1-\frac{2}{2^{2k}}\right)\cdots\left(1-\frac{2^k}{2^{2k}}\right)<\left(1-\frac{2^k/2}{2^{2k}}\right)^{2^k/2}\approx\left(1-\frac{1}{2^{k+1}}\right)^{2^{k-1}}\approx\left(\frac{1}{e}\right)^{1/4}$$

Corollary:

no hash function $H: \{0,1\}^{\cdots} \longrightarrow \{0,1\}^{80}$ can be used

computing 2^{40} values is feasible,

while computing and storing 2^{80} values is impossible (2^{30} petabytes))

renting 1 petabyte storage: 367980 USD, so together about 370 000 000 000 000 USD

brutto social product USA 2020 about: 21 000 000 000 000

Hashing long arbitrarily long messages

assume that we can find a good $F: \{0,1\}^{512} \longrightarrow \{0,1\}^{160}$

what to do if we wish to get a function $F^*: \{0,1\}^* \longrightarrow \{0,1\}^{160}$ with similar properties (behaving like Random Oracle)?

Merkle-Damgard meta-construction based on function Γ:

1. input message M
2. add padding to get a full number of blocks: $M \| \text{padding} = m_1 \| m_2 \| \ldots \| m_N$
3. take initial vector: H_0
4. for $i = 1$ to N:

   $$H_i := \Gamma(H_{i-1}, m_i)$$

5. return H_N

Merkle-Damgard instantiations:

→ **Davies-Meyer:** $H_i := \mathrm{Enc}_{m_i}(H_{i-1}) \otimes H_{i-1}$ where Enc is an encryption function

→ **Matyas–Meyer–Oseas**: $H_i := \mathrm{Enc}_{H_{i-1}}(m_i) \otimes m_i$

→ **Miyaguchi-Preneel**: $H_i := \mathrm{Enc}_{H_{i-1}}(m_i] \otimes m_i \otimes H_{i-1}$

→ **dedicated constructions** based on fixed length input **compress-hashing**:

MD5 (obsolete - do not use except for nonsecurity applications!),

SHA-1,

SHA-2 (a few options, in use),

Keccak (current NIST standard SHA-3)

MD5 story

one of flagship algorithms, used even until 2012, replaced by SHA-1 (similar architecture), later by Keccak (different architecture)

DESIGN:

- pad to the length 448 mod 512 with 10..., add: the message length: a 64 bit number
- split into 512 bit blocks
- IHV_i is the intermediate hash value after block i,

  consisting of 32-bit numbers a_i, b_i, c_i, d_i.
- the initial value $\mathrm{IHV}_0 = (a_0, b_0, c_0, d_0)$ is fixed
- $\mathrm{IHV}_i = \mathrm{MD5Compress}(\mathrm{IHV}_{i-1}, M_i)$
- hash output= the last value IHV_N (after some reformatting)

MD5Compress function

- steps $i = 0,\ldots,63$
- each step involves modular addition, left rotation, non-linear function f_i, adding a constant t_i, rotation by constant s_i
- function $f_i(x,y,z)$ defined as:
  - $F(x,y,z) = (x \wedge y) \vee (\bar{x} \wedge z)$ for $i = 0,\ldots,15,$
  - $G(x,y,z) = (z \wedge x) \vee (\bar{z} \wedge y)$ for $i = 16,\ldots,31,$
  - $H(x,y,z) = x \oplus y \oplus z$ for $i = 32,\ldots,47,$
  - $I(x,y,z) = y \oplus (x \vee \bar{z})$ for $i = 48,\ldots,63$

512-bit message block split into 32-bit words $m_0, \ldots m_{15}$

- the same m_i used 4 times in different phases – this this compicates adjusting message blocks to create a collision
- let w_t denote the input word at step t. it equals

$$\begin{aligned} m_t & \quad \text{for } 0 \leq t \leq 15 \\ m_{(1+5t) \bmod 16} & \quad \text{for } 16 \leq t \leq 31 \\ m_{(15+3t) \bmod 16} & \quad \text{for } 32 \leq t \leq 47 \\ m_{(7t) \bmod 16} & \quad \text{for } 48 \leq t \leq 63 \end{aligned}$$

that, is, we get the following index values for m:

0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,

1,6,11,0,5,10,15,4,9,13,2,7,12,1,6,11,

5,8,11,14,1,4,7,10,13,0,3,6,9,12,15,2,

0,7,14,5,12,3,10,1,8,15,6,13,4,11,2,9

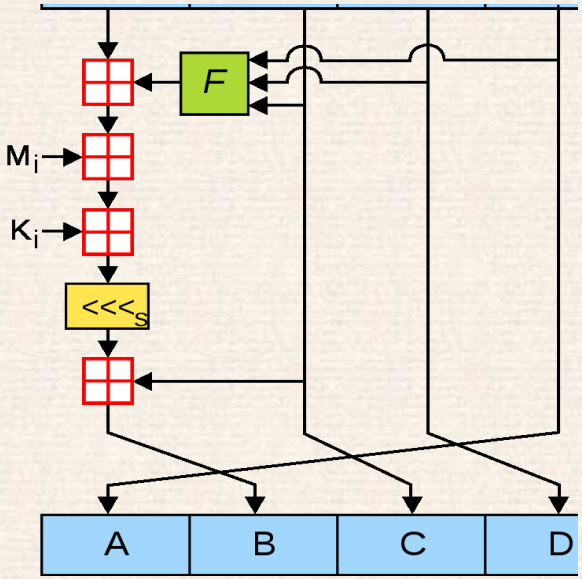
4 consecutive steps

$$a := b + ((a + f_i(b, c, d) + w_i + t_i) \lll s_i)$$

$$d := a + ((d + f_i(a, b, c) + w_{i+1} + t_{i+1}) \lll s_{i+1})$$

$$c := b + ((c + f_i(d, a, b) + w_{i+2} + t_{i+2}) \lll s_{i+2})$$

$$b := c + ((b + f_i(c, d, a) + w_{i+3} + t_{i+3}) \lll s_{i+3})$$



By Surachit - self-made SVG, based on [1] by User:Matt Crypto, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2652649

Differential analysis

- consider X and $X' = X + \Delta$ (X not fixed (random) but Δ to be fixed
- differences in m_i and m'_i influence the computation: typically one differences creates many differences and there is an avelanche...
- differential analysis: in some cases the differences do not grow and even sometimes cancel out
- write conditions that guarantee such cancelling (conditions on bits of a_i, b_i, c_i, d_i for $i = ...$)
- hoping that: for a random X each condition satisfied with pbb $\approx 2^{-k}$ where k information bits fixed by the condition)

  (violating this property would lead to easier attacks!)
- characteristic: list of differences for each step plus approx. probabilities of validity

Differences used

- as numbers: for example $a - a' \bmod 2^{32}$
- bitwise xor: $a \oplus a'$

Relations: if $a - a' = 2^6 \bmod 2^{32}$ then there are a few possibilities for $a \oplus a'$:

i. $a'[7] = 1$ and $a[7] = 0$

ii. $a'[8] = 1, a'[7] = 0$ and $a[8] = 0, a[7] = 1$ (one carry)

iii. ... (two carry bits)

Attack scenario (... but there are many options by follow up work)

only two blocks of 512 bits: M_0, M_1 and M_0', M_1'

initial difference: ΔH_0

after processing 512 bits: ΔH_1

finally: $\Delta H = 0$

$$M_0' - M_0 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 2^{15}, 0, 0, 2^{31}, 0)$$

$$M_1' - M_1 = (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, -2^{15}, 0, 0, 2^{31}, 0)$$

the goal:

$$\Delta H_1 = (2^{31}, 2^{31} + 2^{25}, 2^{31} + 2^{25}, 2^{31} + 2^{25})$$

Notation

$$a_i' = a_i[7, 9, -22]$$

means that a_i' is the same as a_i except for

$a_i[7] = 0, a_i'[7] = 1$

$a_i[9] = 0, a_i'[9] = 1$

$a_i[22] = 1, a_i'[22] = 0$

Characteristic

Step	The output in i-th step for M_0	w_i	s_i	Δw_i	The output difference in i-th step	The output in i-th step for M_0'
4	b_1	m_3	22			
5	a_2	m_4	7	2^{31}	-2^6	$a_2[7, \ldots, 22, -23]$
6	d_2	m_5	12		$-2^6 + 2^{23} + 2^{31}$	$d_2[-7, 24, 32]$
7	c_2	m_6	17		$-1 - 2^6 + 2^{23} - 2^{27}$	$c_2[7, 8, 9, 10, 11, -12, -24, -25, -26, 27, 28, 29, 30, 31, 32, 1, 2, 3, 4, 5, -6]$
8	b_2	m_7	22		$1 - 2^{15} - 2^{17} - 2^{23}$	$b_2[1, 16, -17, 18, 19, 20, -21, -24]$,
9	a_3	m_8	7		$1 - 2^6 + 2^{31}$	$a_3[-1, 2, 7, 8, -9, -32]$
10	d_3	m_9	12		$2^{12} + 2^{31}$	$d_3[-13, 14, 32]$
11	c_3	m_{10}	17		$2^{30} + 2^{31}$	$c_3[31, 32]$

Conditions

c_1	$c_{1,7}=0,\ c_{1,12}=0,\ c_{1,20}=0$
b_1	$b_{1,7}=0,\ b_{1,8}=c_{1,8},\ b_{1,9}=c_{1,9},\ b_{1,10}=c_{1,10},\ b_{1,11}=c_{1,11},\ b_{1,12}=1,\ b_{1,13}=c_{1,13},$ $b_{1,14}=c_{1,14},\ b_{1,15}=c_{1,15},\ b_{1,16}=c_{1,16},\ b_{1,17}=c_{1,17},\ b_{1,18}=c_{1,18},b_{1,19}=c_{1,19},$ $b_{1,20}=1,\ b_{1,21}=c_{1,21},\ b_{1,22}=c_{1,22},\ b_{1,23}=c_{1,23},\ b_{1,24}=0,\ b_{1,32}=1$
a_2	$a_{2,1}=1,\ a_{2,3}=1,\ a_{2,6}=1,\ a_{2,7}=0,\ a_{2,8}=0,\ a_{2,9}=0,\ a_{2,10}=0,\ a_{2,11}=0,$ $a_{2,12}=0,\ a_{2,13}=0,\ a_{2,14}=0,\ a_{2,15}=0,\ a_{2,16}=0,\ a_{2,17}=0,\ a_{2,18}=0,\ a_{2,19}=0,$ $a_{2,20}=0,\ a_{2,21}=0,\ a_{2,22}=0,\ a_{2,23}=1,\ a_{2,24}=0,\ a_{2,26}=0,\ a_{2,28}=1,\ a_{2,32}=1$
d_2	$d_{2,1}=1,\ d_{2,2}=a_{2,2},\ d_{2,3}=0,\ d_{2,4}=a_{2,4},\ d_{2,5}=a_{2,5},\ d_{2,6}=0,\ d_{2,7}=1,\ d_{2,8}=0,$ $d_{2,9}=0,\ d_{2,10}=0,\ d_{2,11}=1,\ d_{2,12}=1,\ d_{2,13}=1,\ d_{2,14}=1,\ d_{2,15}=0,\ d_{2,16}=1,$ $d_{2,17}=1,\ d_{2,18}=1,\ d_{2,19}=1,\ d_{2,20}=1,\ d_{2,21}=1,\ d_{2,22}=1,\ d_{2,23}=1,\ d_{2,24}=0,$ $d_{2,25}=a_{2,25},\ d_{2,26}=1,\ d_{2,27}=a_{2,27},\ d_{2,28}=0,\ d_{2,29}=a_{2,29},\ d_{2,30}=a_{2,30},$ $d_{2,31}=a_{2,31},\ d_{2,32}=0$
c_2	$c_{2,1}=0,\ c_{2,2}=0,\ c_{2,3}=0,\ c_{2,4}=0,\ c_{2,5}=0,\ c_{2,6}=1,\ c_{2,7}=0,\ c_{2,8}=0,\ c_{2,9}=0,$ $c_{2,10}=0,\ c_{2,11}=0,\ c_{2,12}=1,\ c_{2,13}=1,\ c_{2,14}=1,\ c_{2,15}=1,\ c_{2,16}=1,\ c_{2,17}=0,$ $c_{2,18}=1,\ c_{2,19}=1,\ c_{2,20}=1,\ c_{2,21}=1,\ c_{2,22}=1,\ c_{2,23}=1,\ c_{2,24}=1,\ c_{2,25}=1,$ $c_{2,26}=1,\ c_{2,27}=0,\ c_{2,28}=0,\ c_{2,29}=0,\ c_{2,30}=0,\ c_{2,31}=0,\ c_{2,32}=0$
b_2	$b_{2,1}=0,\ b_{2,2}=0,\ b_{2,3}=0,\ b_{2,4}=0,\ b_{2,5}=0,\ b_{2,6}=0,\ b_{2,7}=1,\ b_{2,8}=0,\ b_{2,9}=1,$ $b_{2,10}=0,\ b_{2,11}=1,\ b_{2,12}=0,\ b_{2,14}=0,\ b_{2,16}=0,\ b_{2,17}=1,\ b_{2,18}=0,\ b_{2,19}=0,$ $b_{2,20}=0,\ b_{2,21}=1,\ b_{2,24}=1,\ b_{2,25}=1,\ b_{2,26}=0,\ b_{2,27}=0,\ b_{2,28}=0,\ b_{2,29}=0,$ $b_{2,30}=0,\ b_{2,31}=0,\ b_{2,32}=0$

Attacks complexity and consequences

- functions like MD5 broken (2nd preimage broken in practice)
- SHA-1 conflict freeness broken, but 2nd preimage still not
- SHA-1: chosen prefix attack: one can take $D \neq D'$ and find Z, Z' such that

  SHA-1$(D||Z) =$ SHA-1$(D'||Z')$ (the cost is still very high)

- until 2004 the people belived that SHA-1 is secure, authorities issued recommendations
- ... but prof. Xiaoyun Wang and her team did not believe/knew about it and broke MD-4 and consequently MD-5 and SHA-1

ROGUE Certificates and MD5

- target: create a certificate (webserver, client) that has not been issued by CA
- not forging a signature contained in the certificate but:
  i. find two messages that $\mathrm{Hash}(M_0) = \mathrm{Hash}(M_1)$ and M_0 as well as M_1 have some common prefix that you expect in a certificate (e.g. the CA name)
  ii. submit a request corresponding to M_0, get a certificate with the signature over $\mathrm{Hash}(M_0)$
  iii. copy the signature from the certificate concerning M_0 to a certificate based on M_1
- problems: some data in M_0 are to be guessed: sequential number, validity period,

  some other are known in advance: distinguished name, ...

legitimate website certificate		rogue CA certificate
serial number		serial number
issuing CA		issuing CA
validity period		validity period
domain name	chosen prefixes	rogue CA name
		1024 bit RSA public key
		extensions
		"CA=true"
	.....	tumor
2048 RSA public key	collision bits	
	.....	
extension "CA=false"	identical suffix	

Table.

- finding M_0 and M_1 must be fast (otherwise guessing the serial number will fail)

- attack on MD5, general picture:

message A		message B
prefix P		prefix P'
padding S_r		padding S'_r
birthday blocks S_b		birthday blocks S'_b
near-collision block $S_{c,1}$		near-collision block $S'_{c,1}$
near-collision block $S_{c,2}$		near-collision block $S'_{c,2}$
...		...
near-collision block $S_{c,r}$	$\leftarrow$collision$\rightarrow$	near-collision block $S'_{c,r}$
suffix		suffix

Table.

prefix, birthday bits, near collision blocks:

- birthday bits: 96, end at the block boundary, they are RSA bits – in the genuine certificate, "tumor" (ignored part by almost all software- marked as a comment extension) – in the rogue certificate

  birthday bits make the difference of intermediate hash values computed for both certificates fall into a *good class*

  birthday paradox makes it possible: we may try many possibilities for tumor

- then apply 3 near-collision blocks of 512-bits. website: we have "consumed" 208 + 96 + $3 \cdot 512 = 1840$ bits of the RSA modulus. Rogue certificate: all bits concerned are in the "tumor"

- after collision bits: 2048-1840 = 208 bits needed to complete the RSA public key – how to generate an RSA number with the prefix of 1840 bits already fixed?

  – continue to get a product of two primes:

  → B denotes the fixed 1840-bit part of the RSA modulus followed by 208 ones

  → select at random 224-bit integer q until $B \bmod q < 2^{208}$, continue until both q and $p = \lfloor B/q \rfloor$ are prime. Then

  – $p \cdot q$ is an RSA number

  – $p \cdot q < B$, $B - p \cdot q = B - q \cdot \lfloor B/q \rfloor < 2^{208}$. Hence $p \cdot q$ has the same 1840 most significant bits as B

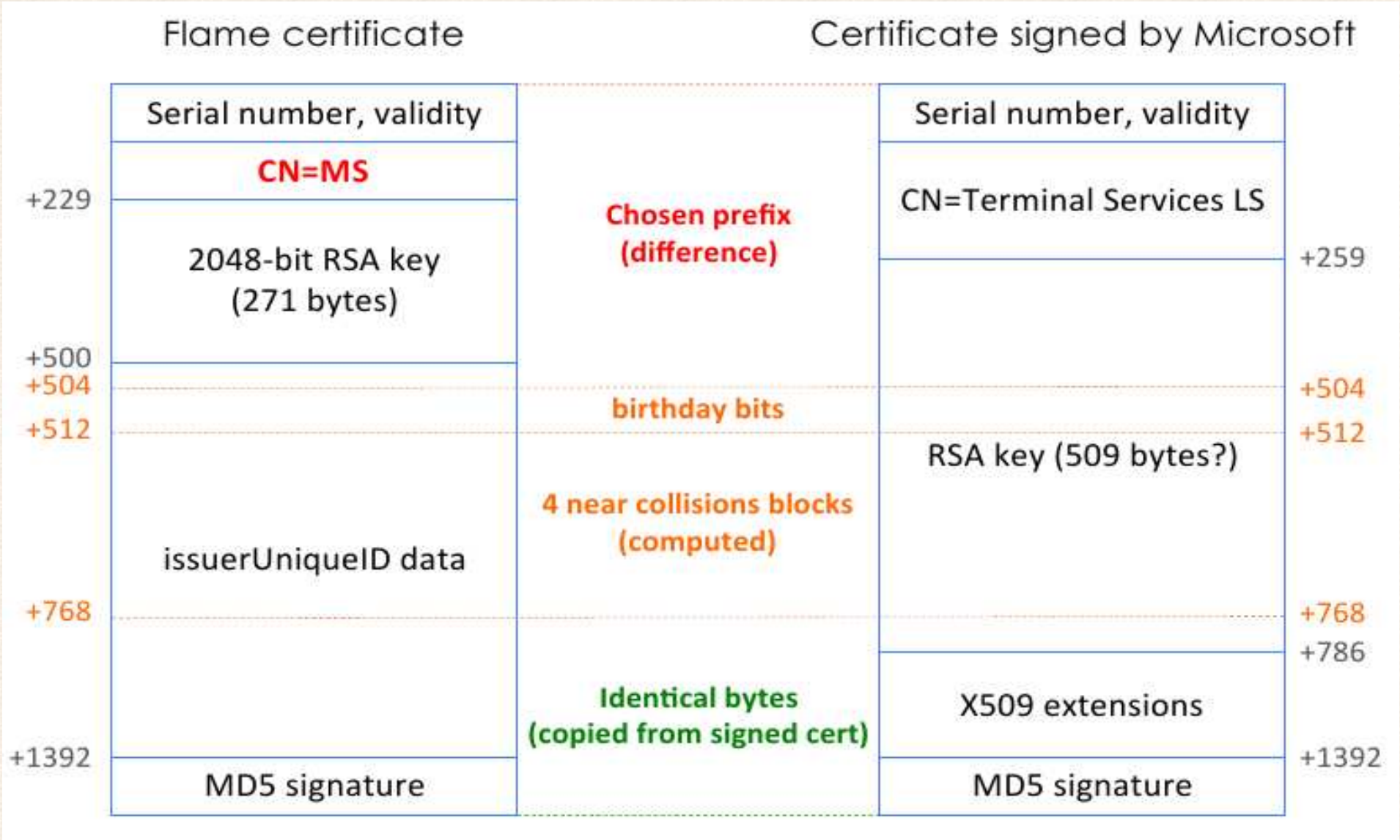
  → not a good RSA number but CA has no possibility to check it

  → ... the attacker can create RSA signature for the certificate request

- attack complexity (number of hash block evaluations) for a chosen prefix MD5: 2^{49} at 2007, 2^{39} in 2009, not much motivation for more work - remove MD5 certificates! (For a collision: 2^{16})
- **ethical disclosure**:
  - $\rightarrow$ attack found
  - $\rightarrow$ real collision computed as a proof-of-concept
  - $\rightarrow$ CA informed and given time to update
  - $\rightarrow$ publication
  - $\rightarrow$ code available

FLAME

- malware discovered 2012, 20MB, sophisticated code, mainly in Middle East,
- draft of the attack:
  - client attempts to resolve a computer name on the network, in particular makes WPAD (Web Proxy Auto-Discovery Protocol) requests
  - Flame claims to be WPAD server, provides wpad.dat configuration file
  - victim that gets wpad.dat sets its proxy server to a Flame computer (later no sniffing necessary!)
  - Windows updates provided by FLAME computer. The updates must be properly signed to be installed!
  - signatures obtained for terminal Services (not for Windows updates!), certificates issued by Microsoft.
  - till 2012 still signatures with MD5 hash
  - MD5 collision necessary to cheat


Flame certificate
Certificate signed by Microsoft
Serial number, validity
CN=MS
+229
2048-bit RSA key
(271 bytes)
+500
+504
+512
issuerUniqueID data
+768
MD5 signature
+1392
Chosen prefix
(difference)
birthday bits
4 near collisions blocks
(computed)
Identical bytes
(copied from signed cert)
Serial number, validity
CN=Terminal Services LS
+259
+504
+512
RSA key (509 bytes?)
+768
+786
X509 extensions
+1392
MD5 signature

HMAC

keyed message authentication code:

i. Alice and Bob share a secret K

ii. Alice sends a message M to Bob

iii. Alice computes $H := \mathrm{HMAC}(K; M)$ and sends H to Bob

iv. Bob computes $H' := \mathrm{HMAC}(K; M)$ and accepts M iff $H' = H$

Requirement:

given M and H it is infeasible to create M' and H' such that $H' = \mathrm{HMAC}(K, M')$

(without K of course)

Standard RFC 2104

RFC="Request For Comments" but in fact RFC are semi-formal standards

no RFC for $X \Rightarrow$ nobody will ever consider to use X

generic construction for HMAC with a hash function:

$$\mathrm{HMAC}(K, M) = \mathrm{Hash}(K \oplus \mathrm{opad} \| \mathrm{Hash}(K \oplus \mathrm{ipad} \| M))$$

where $\mathrm{opad} \neq \mathrm{ipad}$ are constants

SHA-3

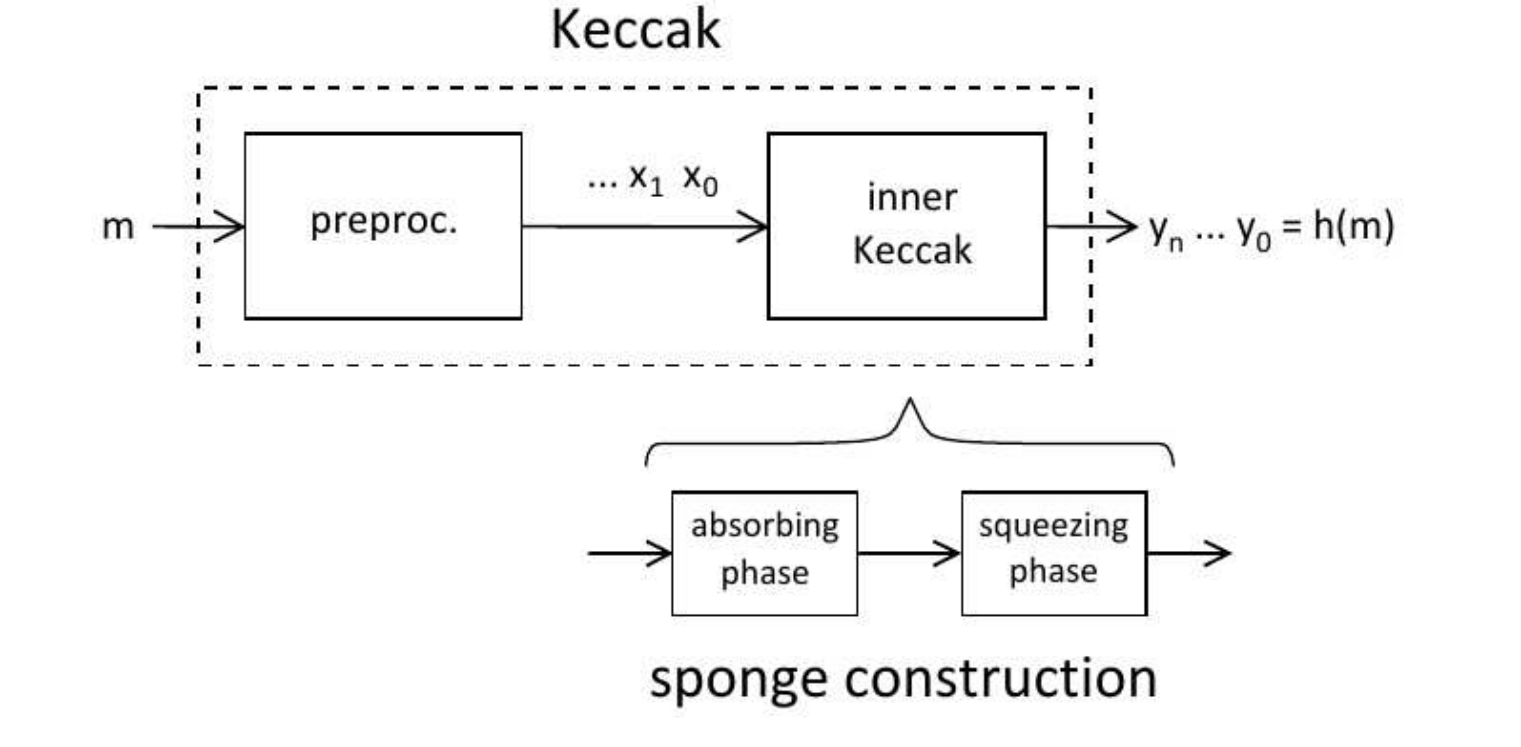
reasons:

- the attacks against MD5 applicable to the whole family - also SHA-1, SHA-2
- for SHA-2 no real threat so far but ... maybe a good idea to have a hash based on a different concept
- more flexibility: SHA-2 provides a fixed length output, some applications (stream ciphers) would prefer other output length
- increase the speed of hardware implementation

SHA-2 family and SHA-3 coexist as standards of NIST

KECCAK and Sponge construction

- **former** (Merkle-Damgard): output is the value after absorbing all message blocks
- **sponge:** first absorbe, but then "squize" the sponge outputting small parts but still changing the internal state

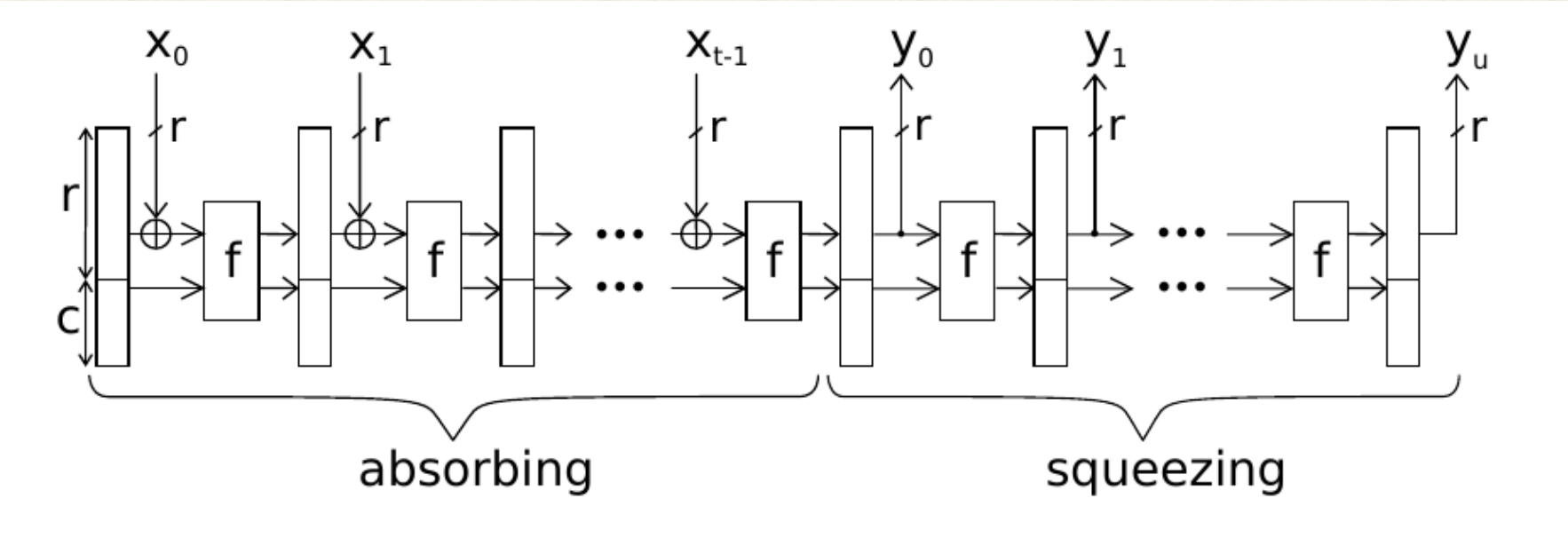


(all pictures on Keccak by: Christof Paar, Jan Pelzl)

Keccak-f function

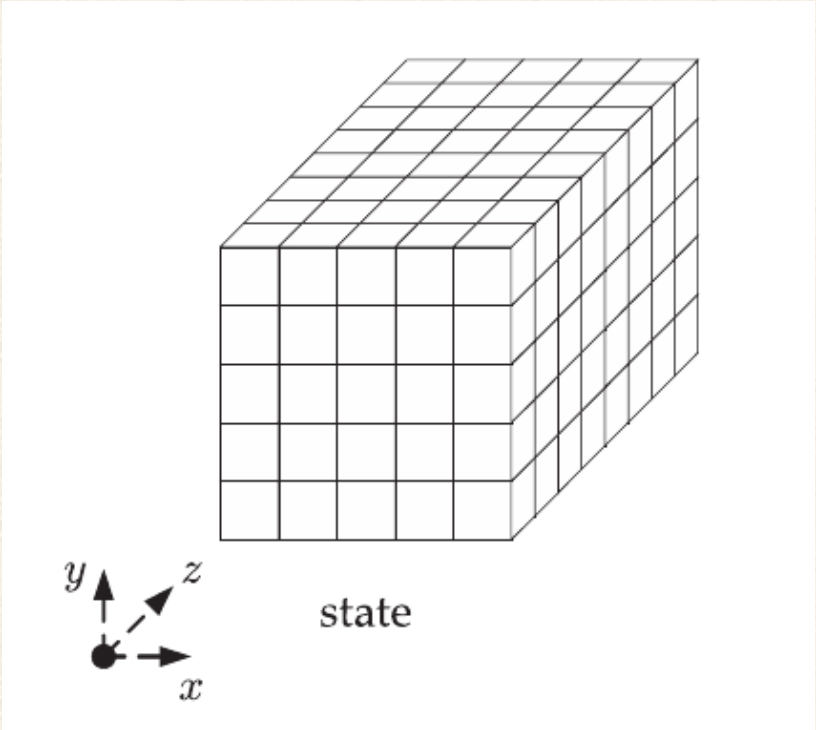
c and r are parameters, f is a function on $25 \cdot 2^l$ bits, where l is again a parameter

there are tables for combinations of parameters



Dimensions:

- **previously:** working on (say) four 32-bit blocks with Round Robin fashion
- **Keccak:** 3-dimensional structure of size 5x5xw, operations in all 3 dimensions to mix



Round

operations

→ θ **(theta)** important for diffusion

→ ρ **(rho)** dispersion within each slice

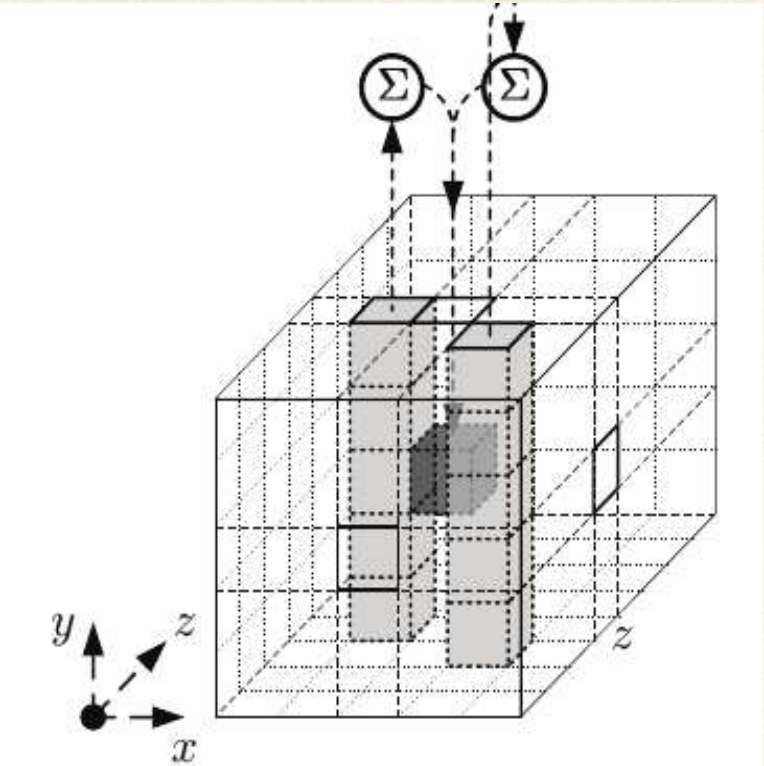
→ π **(pi)** rearranging the positions of lanes

→ χ **(chi)** simple non-linear opeartion

→ ι **(iota)** one lane xor-ed with a constant

θ (theta)

$$a[i,j][k] \leftarrow a[i,j][k] \oplus \text{parity}(a[0...4][j-1][k]) \oplus \text{parity}(a[0...4][j+1][k-1])$$



ρ (rho) and π (pi)

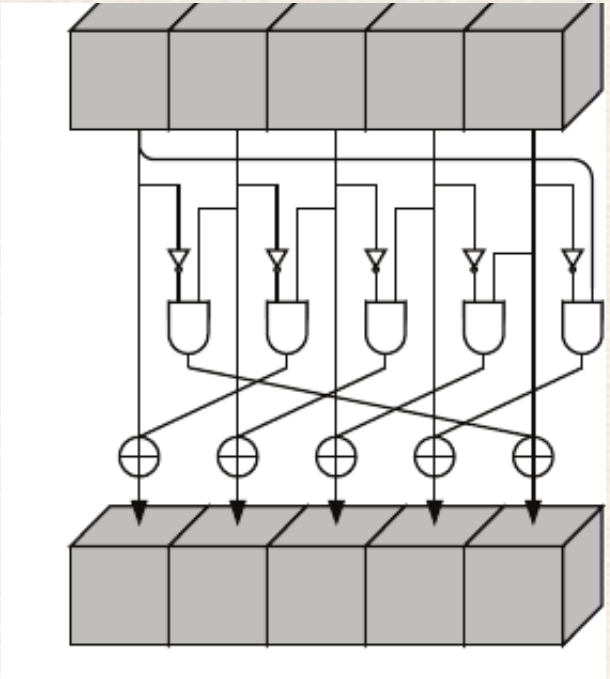
$b[y, 2x + 3y] = \mathrm{rot}\,(a[x, y], r[x, y])$, for $x, y = 0, ..., 4$ (operations mod 5)

rotation matrix r depends on the round number

χ (chi)

$$a[x,y] = b[x,y] \oplus ((b[x+1,y]) \wedge b[x+2,y]),$$

$$x, y = 0, 1, 2, 3, 4$$



ι (iota)

Exclusive-or a round constant into $a[0,0]$

HMAC

message authentication code based on Hash function and a secret key

HMAC computation for message M and key K:

$$h := \mathrm{HMAC}_K(M) = \mathrm{Hash}((K \oplus \mathrm{opad}) \| \mathrm{Hash}((K \oplus \mathrm{ipad}) \| M))$$

upon receiving M and h, the HMAC of M is recomputed and compared with h

Complexity Issues

- Sha-3 (Keccak) is optimized but still more costly than encryption while counting the number of gates in hardware implementation
- IoT may require simpler solutions – "lightweight" even at the price of the security properties

IoT example: broadcast authenticationTesla

Hash chain: $K_i = \mathrm{Hash}(K_{i+1})$ ($K_0, \ldots, K_N$ must be generated in advance)

message i contains: payload M_i, MAC $\mathrm{HMAC}(F(K_i), M_i)$ and K_{i-d},
for a one-way function F

verification: message $i+d$ reveals K_i, then check HMAC

conclusion: message $i+d$ sent by the same person (as for the message i)