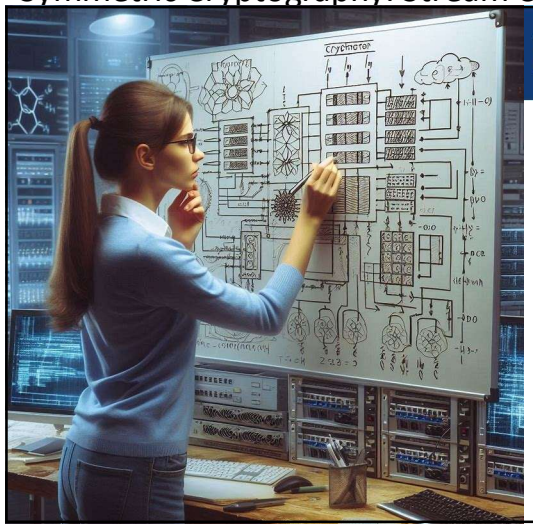


KU LEUVEN

COSIC

Symmetric Cryptography: Stream Ciphers and Hash functions

Bart Preneel
COSIC-KU Leuven
firstname.lastname(AT)esat.kuleuven.be
@bpreneel1 preneel@infosec.exchange
Dubrovnik, June 2025

KU LEUVEN

1

KU LEUVEN

COSIC

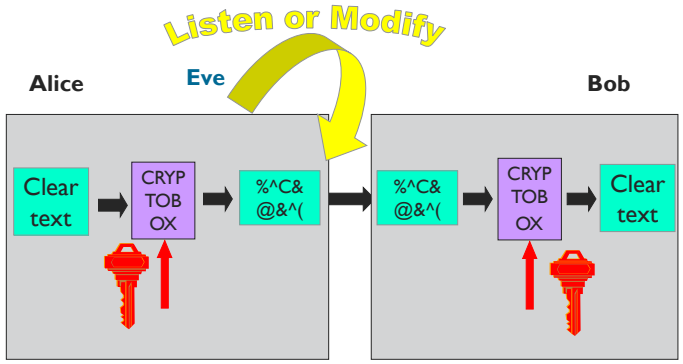
Symmetric cryptography: Introduction

Part I

KU LEUVEN

2

Symmetric cryptology: principle

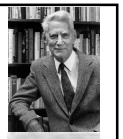


3

KU LEUVEN

3

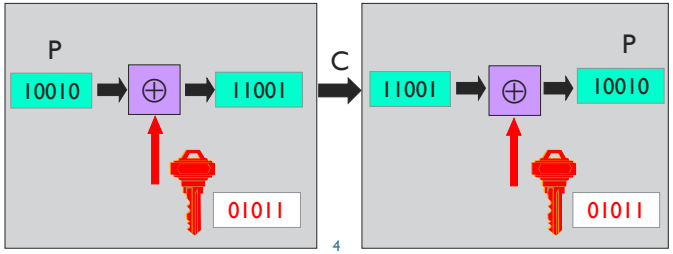
One time pad

Vernam scheme (1917) **Shannon (1948)**

Mauborgne: one time pad (1917+x) F. Miller (1882)

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



4

KU LEUVEN

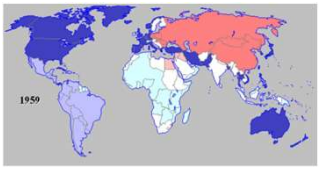
4

One time pad: properties

- perfect secrecy: ciphertext gives opponent no additional information on the plaintext or H(P|C)=H(P)
- impractical: key is as long as the plaintext
- but this is optimal: for perfect secrecy one has always H(K) ≥ H(P)

One time pad: Venona Project (1942-1948)

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

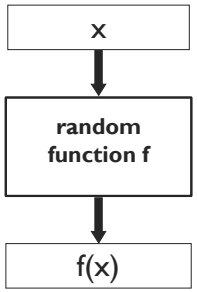
reuse of key material is also known as "transmission in depth"
https://en.wikipedia.org/wiki/Venona_project

Example: c1 V c2 (not +)

Cryptographic Building Blocks

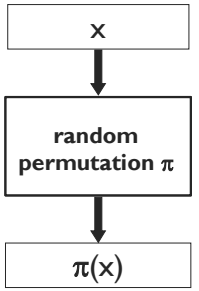
- Random Function
- Random Permutation
- PseudoRandom Function (PRF)
- PseudoRandom Permutation (PRP)

Random Function - Random Permutation



$f: \{0,1\}^n \rightarrow \{0,1\}^n : x \rightarrow f(x)$

injection



$\pi: \{0,1\}^n \rightarrow \{0,1\}^n : x \rightarrow \pi(x)$

bijection
(π^{-1} exists)

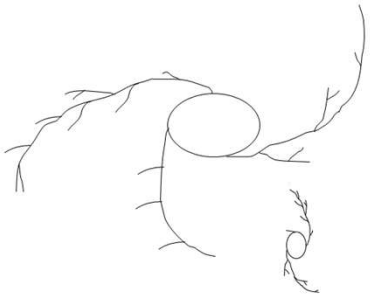
Functional Graph: Random Function - Permutation

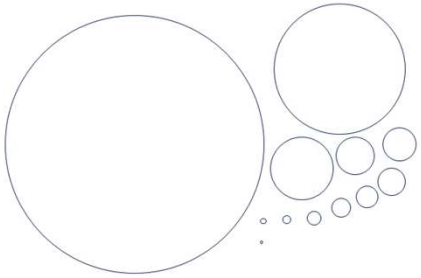
$x \to f(x) \to f(f(x)) \to f(f(f(x))) \to \ldots$

- average cycle length and distance to cycle $\approx 0.63 \cdot 2^{n/2}$

$x \to \pi(x) \to \pi(\pi(x)) \to \pi(\pi(\pi(x))) \to \ldots$

- random permutation – expected length of largest cycle $\approx 0.62 \cdot 2^{n}$

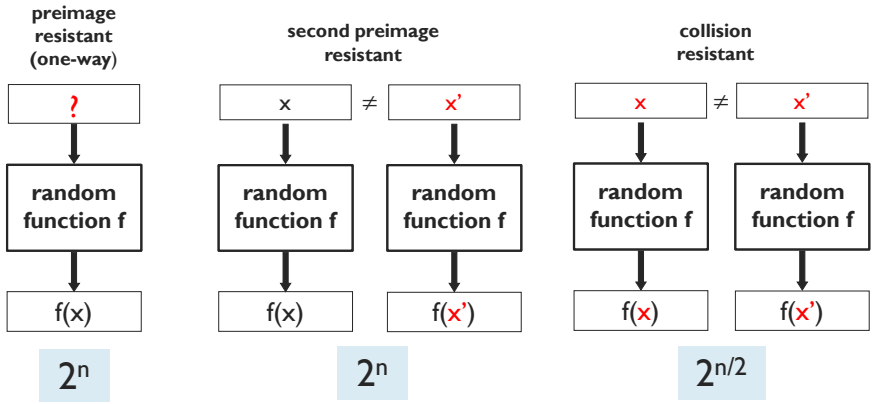




9

9

Security Requirements for Random Function



10

10

Security Requirements for Random Permutation

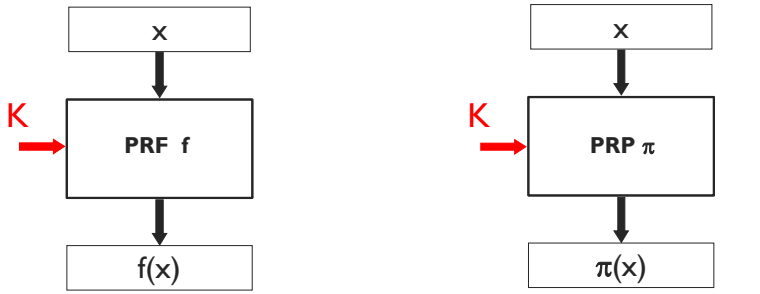
What is different?

- Bijection hence second preimage resistance and collision resistance not meaningful
- Preimage resistance is meaningful but we don't know how to achieve it for efficient primitives (RSA is an example)

11

11

PseudoRandom Function (PRF) - PseudoRandom Permutation (PRP) Family



$f: \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n : x \to f_K(x)$

Fixed input length MAC algorithm

$\pi: \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n : x \to \pi_K(x)$

Block cipher

MAC algorithms only need to be unpredictable but in practice they are all PRFs

12

12

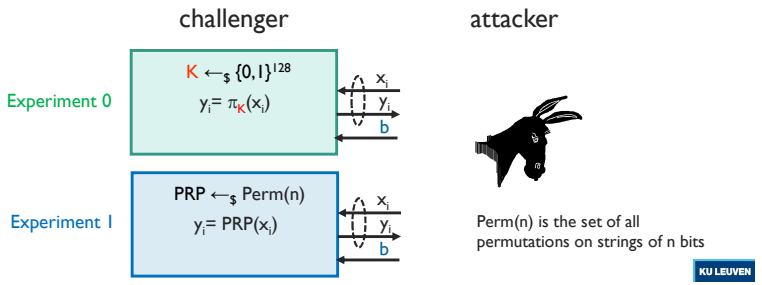
Security of a PRF/PRP

- Many applications need secret function or permutation
- Solution: secret key K selects a random function or permutation: (approximate a complex object with an easy to realize object):
  - the number of functions from $\{0,1\}^n$ to $\{0,1\}^n$: $(2^n)^{2^n} >> 2^k$ (# keys)
  - the number of permutations of $\{0,1\}^n$: $2^n! >> 2^k$ (# keys)
    - For AES-128: n = 128 $2^{128}! \approx 2^{2^{135}} >> 2^{128}$
- Computational indistinguishability
  - implies unpredictability of the output
  - implies security against key recovery
- Note that the distinguisher can always try all 2^k keys hence k should be large

Security of PRP: attack game (PRF is similar)

- It is computationally infeasible to distinguish the PRP π from a random permutation
- Advantage of a distinguisher should be "small"

$Adv_{\pi/PRP}$ = | Pr (W_0) - Pr (W_1) | with W_b probability that attacker outputs 1 in Experiment b

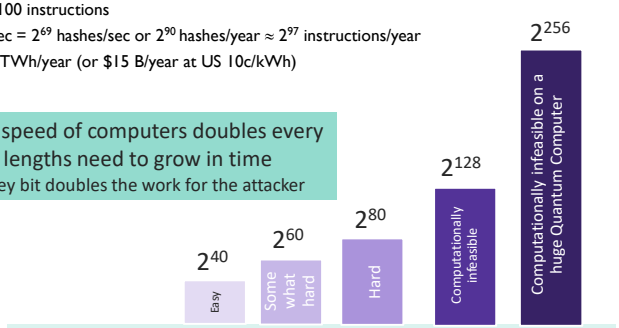


Recommendations: Key Lengths

2025: 1 million machines with 32 cores @ 5 GHz can execute 2^{57} instructions/sec or 2^{81} instructions/year
- trying 1 key ≈ 100 instructions

Bitcoin: 600 Exahashes/sec = 2^{69} hashes/sec or 2^{90} hashes/year ≈ 2^{97} instructions/year
- Electricity: 150 TWh/year (or $15 B/year at US 10c/kWh)

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



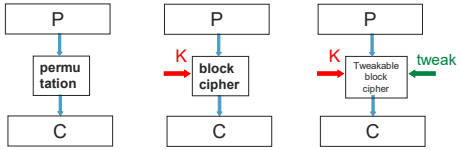
Extensions: Variable Input/Output Length

Fixed Input and Output Length	Variable Input Length	Variable Output Length	Variable Input and Output Length
Random function	Hash function		Xtensible Output Function (XOF)
PRF	MAC algorithm	PseudoRandom Bit Generator (PRBG)	
Random permutation	-	-	
PRP	-	-	Accordion function [NIST]

Extensions: Tweak

Adding an extra public parameter: tweak, salt, spice, ("key")

- Hash function: Universal One-Way Hash Function (UOWHF)
- Stream cipher: tweakable PRBG (tPRBG)
- Block cipher: tweakable block cipher



17

KU LEUVEN

17

Cryptographic Building Blocks

- MAC algorithms
- Authenticated encryption
- Block ciphers
- Stream ciphers

18

KU LEUVEN

18

Data authentication: MAC algorithms (VIL PRF)

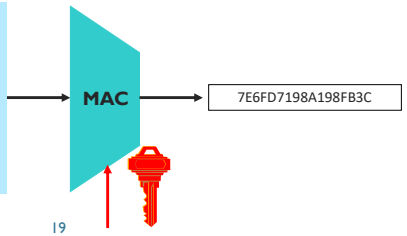
- Replace protection of authenticity of (long) message by protection of secrecy of (short) key
- Append MAC to the plaintext

CBC-MAC (CMAC/LMAC)

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.

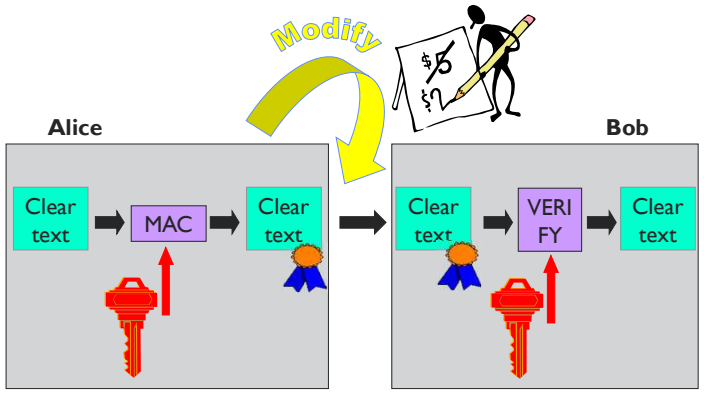


19

KU LEUVEN

19

Data Authentication: MAC algorithm

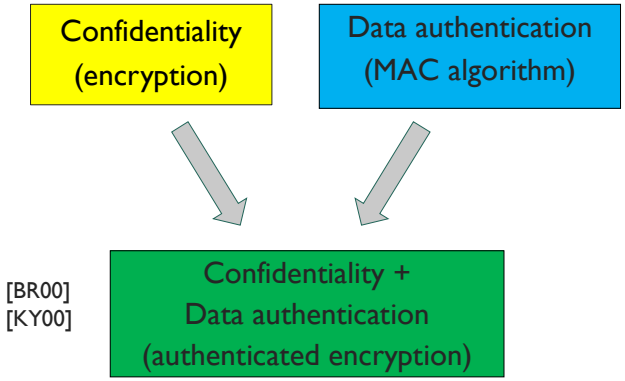


20

KU LEUVEN

20

Authenticated Encryption



[BR00]
[KY00]

21

21

Authenticated Encryption

Generic composition [BN'00][NRS'14]

- Encrypt-then-MAC with 2 independent keys
  - IPsec, TLS 1.2, 1.3
- MAC-then-Encrypt with 2 independent keys
  - TLS 1.1 and older, 802.11i WiFi
- MAC-and-Encrypt with 2 independent keys

Design "from scratch"

- Integrated authenticated encryption schemes: combined operation with a single key: GCM, GCM-SIV, CCM, OCB3, duplex, AEGIS,…

22

22

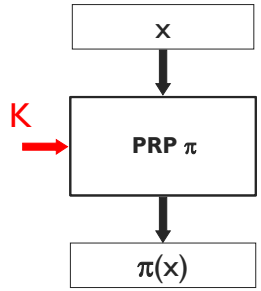
Authenticated Encryption

- Modern encryption: always be authenticated encryption (with associated data)
- Data authentication without encryption is ok but not the other way
- Limitations
  - Typically does not hide the **leng**th of the plaintext (unless randomized padding but even then…)
  - Ciphertext becomes random string: "normal" crypto does not encrypt a credit card number into a (valid) credit card number
  - Does **not** hide existence of plaintext (requires steganography)
  - Does **not** hide that Alice is talking to Bob (e.g. Tor, Nym)
  - Does **not** hide traffic volume (requires dummy traffic)

23

23

Block cipher: PRP family



permutation of plaintext x of n bits
under control of key K of k bits with

$\pi: \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n : x \rightarrow \pi_K(x)$

Block cipher

24

24

Some well-known block ciphers

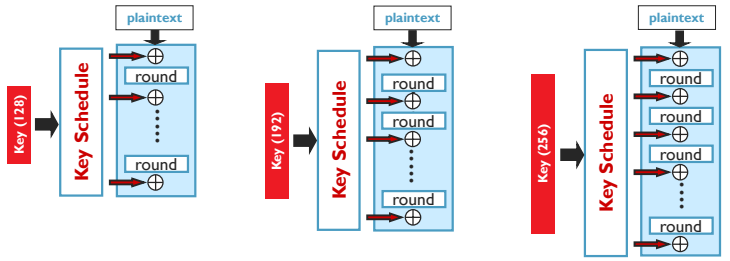
	Year	Block length n	Key length k	
DES	1977	64	56	History
Two key 3-DES	1978	64	112	Legacy
IDEA	1991	64	128	Legacy
AES-128	1997	128	128	Recommended
AES-256	1997	128	256	Recommended
KASUMI	2000	64	64/128	Lightweight
Prince	2012	64	128	Lightweight
Rijndael-256	1997	256	256	

AES block cipher (2001)

AES-128
10 rounds
Sensitive/classified (SECRET)

AES-192
12 rounds
Classified (TOP SECRET)

AES-256
14 rounds
Classified (TOP SECRET)



KU LEUVEN

COSIC

Stream ciphers

Part II

Stream ciphers: outline

- Definitions
- Generic attacks
- Constructions
- Conclusions

Stream ciphers and block ciphers: history

- 1882 one-time pad
- 1930s Hagelin
- 1930s Enigma
- 1960s LFSRs

- 1987 RC4
- 1988 A5/1, A5/2
- 1992 SEAL
- 199? CRYPTO-1
- 1998 GEA-1, GEA-2, Panama
- 1999 E0, SNOW
- 2001 MUGI
- 2004 Trivium
- 2006 SNOW 3G
- 2008 Chacha20
- 2010 ZUC
- 2019 SNOW V (5G)

2004 2008 **estream**

Block ciphers

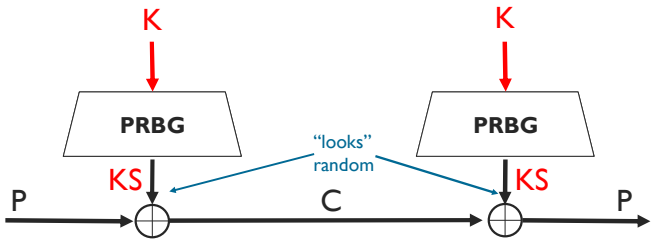
- 1977 **DES**
- 1978 Triple-DES
- 1987 FEAL
- 1990 Khufu/Khafre/IDEA
- 1999 KASUMI
- 2000 **AES**

1997 2000 **AES competition**

- 2007 Present
- 2012 Prince
- 2013 Simon, Speck

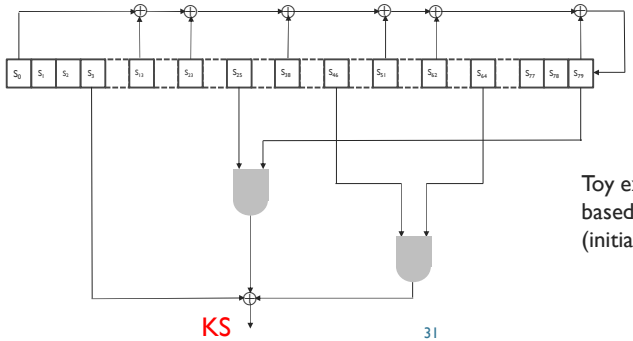
Synchronous Stream Cipher (SSC): replacing the one-time pad by a PRBG

PseudoRandom Bit Generator stretches a short key K (≥ 128 bits) to a long keystream KS that is computationally indistinguishable from random



PseudoRandom Bit Generator (PRBG): Toy Example

turn a short key K (here 80 bits) into a large key stream: KS = f(K)



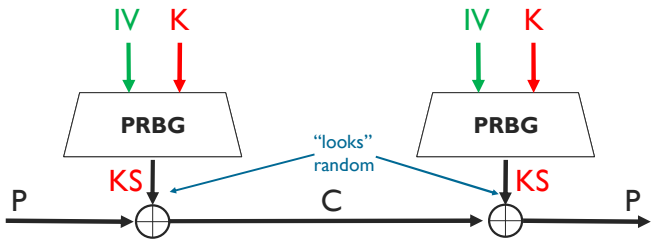
Toy example: filter generator based on LFSR
(initialize register S with K)

PRBG to Tweakable PRBG (tPRBG)

- If the same key is used for multiple files: transmission in depth and the XOR of two plaintext strings leaks
- Solution: add an extra public parameter called initialization value (IV) hence KS = f(K, IV)
- IV shall never repeat for a key K
  - counter value (stateful encryption)
  - counter derived from application: frame counter, packet counter,…
  - random value: be careful for birthday paradox
    - exercise: if the length of IV is v bits and the probability that the IV repeats should be at most α, how many messages can one encrypt with one key?

Synchronous Stream Cipher (SSC): replacing the one-time pad by a tPRBG

Tweakable PRBG stretches a short key to a long keystream that looks random; every tweak (IV) generates an independent sequence



33

KU LEUVEN

33

Stream ciphers

- Recipient needs to be synchronized with sender: synchronous stream cipher
- 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
- Typically better performance than block ciphers: simpler operations on shorter chunks

34

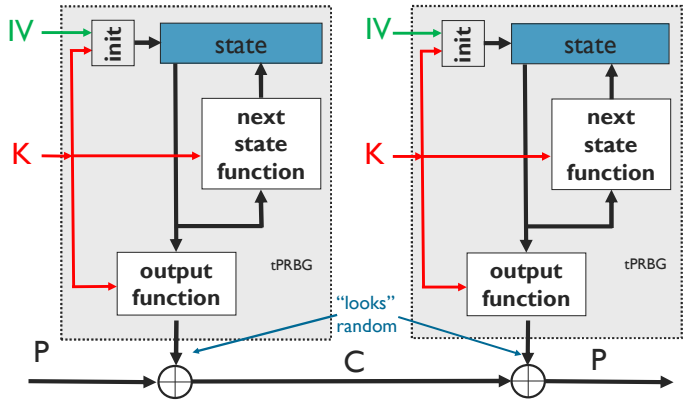
KU LEUVEN

34

Generic attacks

35

Synchronous Stream Cipher (SSC): finite state machine



36

KU LEUVEN

36

Generic attacks

- Distinguishing attacks: cycle structure
- Key recovery attacks

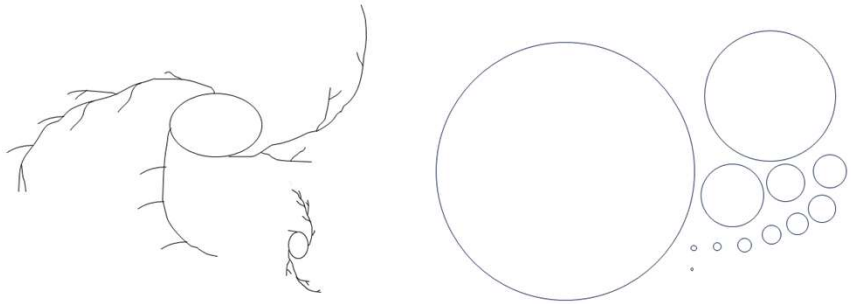
- State recovery attacks

37 KU LEUVEN

37

Next state function random function vs. permutation

- State size of m bits
- Left: random function – average cycle length and distance to cycle ≈ $0.63.2^{m/2}$
- Right: random permutation – expect length of largest cycle ≈ $0.62.2^{m}$



38

38

Generic attacks

- Distinguishing attacks: cycle structure
- Key recovery attacks
  - targeted key recovery (one particular key): $T = 2^{k-1}$
  - existential key recovery (one of μ keys): $T = 2^{k-1}/(\mu+1)$
  - universal key recovery (μ out of μ keys): time-memory tradeoffs (precomputation $P = 2^k$, memory $M = 2^{2k/3}$, time per key $T = 2^{2k/3}$)
- State recovery attacks: if key K is only used during initialization
  - $P = M = 2^{2m/3}$ and $T = D = 2^{m/3}$
  - if this attack applies, m = 2k (m=80 would not be sufficient)

Choosing parameters is tricky

39 KU LEUVEN

39

Stream cipher constructions

40

Some well-known stream ciphers

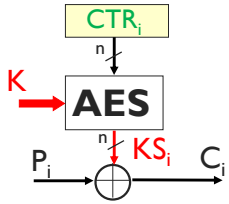
	Year	Key length k	IV length v	State length m	Status
RC4 (WEP,TLS)	1987	40-128	**No IV**	2048	History
A5/1 (2G)	1989	64	64	64	Legacy
E0 (Bluetooth)	1991	128	74	132	Legacy
Trivium	2004	80	80	288	Legacy
Kreyvium	2018	128	128	288	
Grain-128	2004	128	128	256	
HC-256	2004	256	256	65536	
ChaCha20 (TLS)	2013	256	96	(512)	

41 KU LEUVEN

41

CounTer Mode (CTR): complex output function

$C_i = P_i \oplus E_K(CTR_i), CTR_i ++$



Initialization function: state = IV || 0^{32}

Next state function: state = state++

Output function AES_K(state)

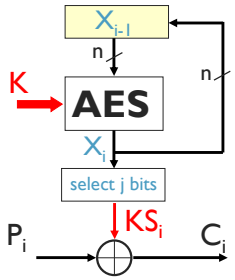
state initialized with random IV, or CTR_0 = IV, typically 32 rightmost bits of IV equal to 0 (32-bit CTR)

42 KU LEUVEN

42

Output Feedback Mode (OFB): complex nxt state fnctn

$X_i = E_K(X_{i-1}), C_i = P_i \oplus$ leftmost j bits of (X_i)



Initialization function: state = IV

Next state function: state = AES_K(state)

Output function: state (or j bits from state)

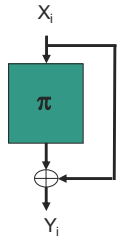
state initialized with random IV, or X_0 = IV, $j \le n$ (typically j = n)

43 KU LEUVEN

43

ChaCha20 (1/2)

- PRG based on 512-bit permutation π that stretches a 256-bit seed (key) to a very long output
  - $X_i = constant_{256} \| seed_{256} \| ctr_{64} \| nonce_{64}$
  - Y_i = key stream added to plaintext
  - $\oplus$ is not XOR but wordwise addition mod 2^{32}
- Parallel by construction
- Allows for random access
- Used for Authenticated Encryption in TLS 1.3 with Poly1305



44 KU LEUVEN

44

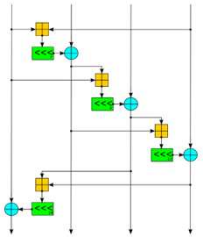
ChaCha20 (2/2)

π consists of 10 times 8 quarter rounds

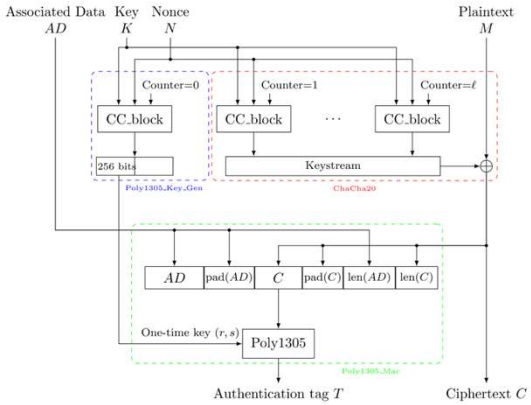
each quarter round consists of 12 operations on 32-bit words

instructions are efficient on processors; state in registers

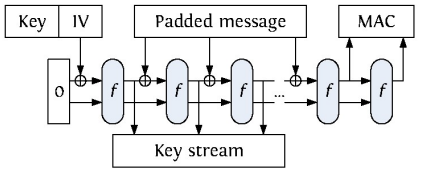
Quarter round of π



By Sissssou - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=5805693

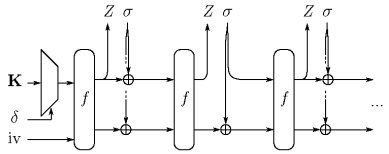


SpongeWrap design for (nonce-based) AEAD



These are not synchronous stream ciphers: plaintext is injected into the state

Keyed Duplex: more efficient



Source: G. Van Assche

Stream cipher designs

- Many stream cipher designs have a relatively simple next state function and output function (better performance than a block cipher or a large random permutation)
- This comes at the cost of a more complex initialization function (needs to be a PseudoRandom function)
- Designs
  - Based on a block cipher: CTR, OFB
  - Based on a large permutation: ChaCha20
  - Software: shuffles: RC-4, HC-128
  - Hardware: LFSR + nonlinear output or clock: A5/1, E0
  - Hardware: NLFSR: SNOW-3G, SNOW-5G, ZUC

A simple cipher: RC4 (1987)

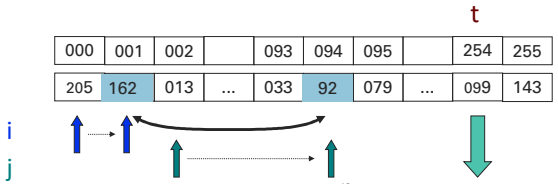
- designed by Ron Rivest (MIT)
- leaked in 1994
- S[0..255]: secret table derived from user key K

```
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)

Next state and output function

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



49

KU LEUVEN

49

RC4: weaknesses

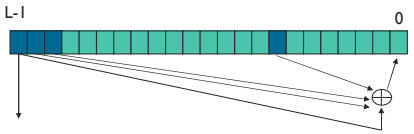
- was often used with 40-bit key (US export restrictions until Q4/2000)
- best known general shortcut attack: 2^{241} [Maximov-Khovratovich'09]
- weak keys and key setup (shuffle theory)
- large statistical deviations
  - bias of output bytes (sometimes very large)
  - can recover 220 out of 256 bytes of plaintexts after sending the same message 1 billion times (WPA/TLS) [Isobe-Ohigashi-Watanabe-Morii '13] [AlFardan-Bernstein-Paterson-Poettering-Schuldt'13][Vanhoef-Piessens'15]
- problem with resynchronization modes (WEP)

50

KU LEUVEN

50

Linear Feedback Shift Register (LFSR)



+ good randomness properties
+ mathematical theory
+ maximum period (2^L-1)
+ compact in hardware
- too linear: easy to predict

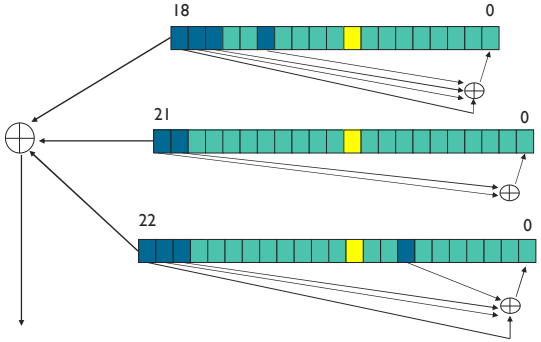
- Destroy linearity of LFSRs
  - with irregular clocking
    - e.g. A5/1 (GSM)
  - with additional memory with nonlinear update
    - e.g. E0 (Bluetooth), SNOW 3G, ZUC, SNOW 5G

51

KU LEUVEN

51

A5/1 stream cipher (GSM)



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

52

KU LEUVEN

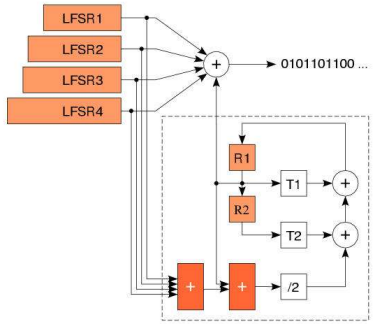
52

A5/1 stream cipher (GSM)

A5/1 attacks

- exhaustive key search: 2^{64} (or rather 2^{54})
- search 2 smallest registers: 2^{41} values – a few steps to verify a guess
- [BB05]: 10 minutes on a PC
  - 3-4 minutes of ciphertext only
- [Nohl-Paget'09]: "rainbow tables" (time-memory tradeoff)
  - seconds with a few frames of ciphertext only

Bluetooth stream cipher (E0)

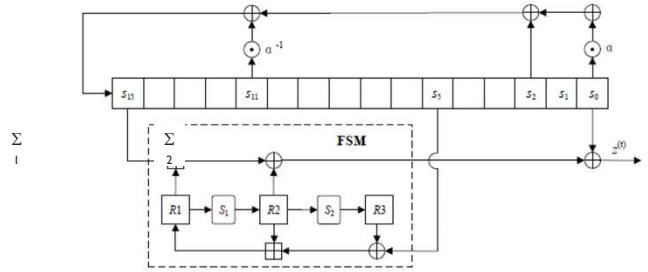


brute force: 2^{128} steps

[Lu+05] 24 known bits of 2^{24} frames, 2^{38} computations, 2^{33} memory

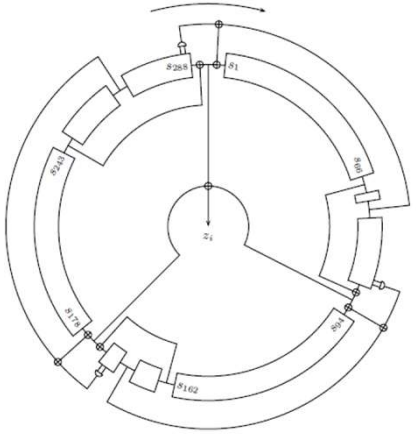
Snow 3G

608-bit state (19 32-bit words)
128-bit key and 128-bit IV
Nonlinear FSM
Parallelization possible



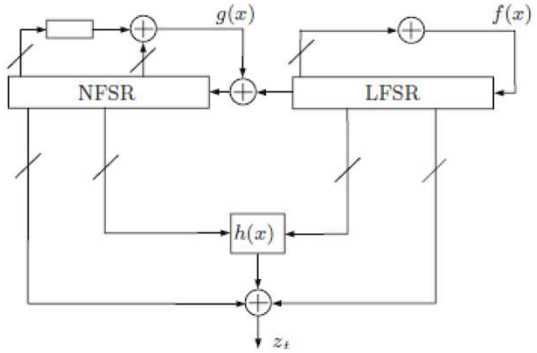
Trivium

288-bit state
80-bit key and IV
Only 3 AND gates
Parallelization possible
128-bit variant Kreyvium



Grain-128

256-bit state (2 128-bit registers)
128-bit key and 96-bit IV
Nonlinear Shift Register
Parallelization possible



57 KU LEUVEN

57

Conclusion: stream ciphers

- Need to be used in authenticated encryption mode (e.g. 3G/4G/5G with a GMAC variant or ChaCha20 with Poly1305)
- No Swiss army knife and fewer open standards (advantage of block ciphers and sponges)
- Beneficial in some areas:
  - high speed (authenticated) encryption in software or hardware
  - computing on encrypted data: low AND depth and few AND gates
- Interesting target for cryptanalysis (still less understood)

58 KU LEUVEN

58

KU LEUVEN

COSIC

Cryptographic Hash Functions

Part III

KU LEUVEN

59

Hash function:

"random" function with variable input length and fixed output length

X.509 Annex D
MDC-2
MD2, MD4, MD5
SHA-1

RIPEMD-160
SHA-256
SHA-512

SHA-3

This is an input to a crypto-graphic 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).

h

1A3FD4128A198FB3CA345932

24/07/2025 KU LEUVEN

60

Applications

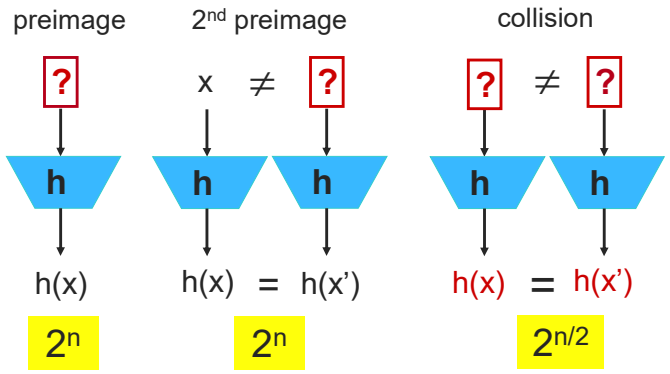
- short unique identifier to a string
  - digital signatures
  - data authentication
- one-way function of a string
  - protection of passwords
  - micro-payments
- confirmation of knowledge/commitment
- pseudo-random string generation/key derivation
- entropy extraction
- construction of MAC algorithms, stream ciphers, block ciphers, digital signatures schemes (Sphincs+, LMS, XMSS…),….

2005: 800 uses of MD5 in Microsoft Windows

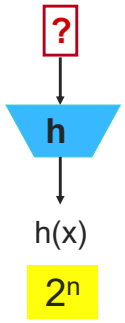
Hash functions: outline

- Definitions
- Iterations (modes)
- Compression functions
- Constructions
- Conclusions

Security requirements (n-bit result)

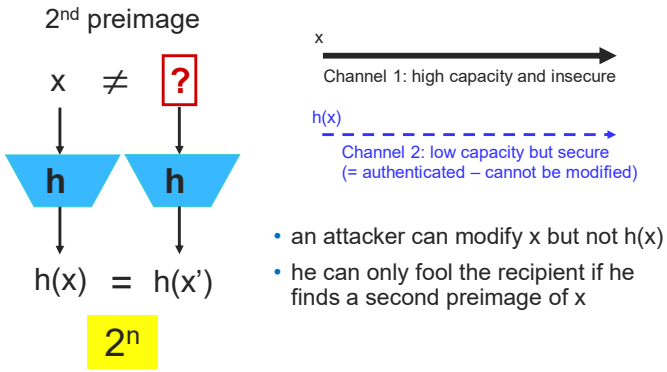


Preimage resistance



- in a password file, one does not store
  - (username, password)
- but
  - (username,hash(password))
- this is sufficient to verify a password
- an attacker with access to the password file has to find a preimage

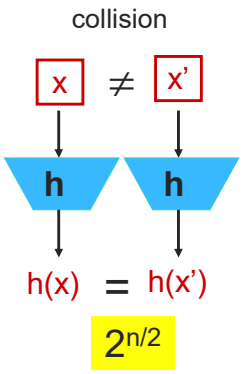
Second preimage resistance



- an attacker can modify x but not h(x)
- he can only fool the recipient if he finds a second preimage of x

Collision resistance

- hacker Alice prepares two versions of a software driver for the O/S company Bob
  - x is correct code
  - x' contains a backdoor that gives Alice access to the machine
- Alice submits x for inspection to Bob
- if Bob is satisfied, he digitally signs h(x) with his private key
- Alice now distributes x' to users of the O/S; these users verify the signature with Bob's public key
- this signature works for x and for x', since h(x) = h(x')



Brute force (2nd) preimage

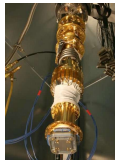
- **multiple target second preimage (1 out of many):**
  - if one can attack 2^t simultaneous targets, the effort to find a single preimage is 2^{n-t}
- **multiple target second preimage (many out of many):**
  - time-memory trade-off with $\Theta(2^n)$ precomputation and storage $\Theta(2^{2n/3})$ time per (2nd) preimage: $\Theta(2^{2n/3})$ [Hellman'80]
- answer: randomize hash function with a parameter S (salt, tweak, spice, key,…)

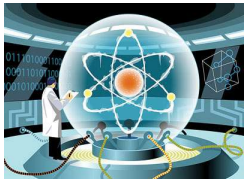
Brute force attacks in practice (2025)

- (2nd) preimage search
  - n = 128: 10 B$ for 5 billion years
  - n = 128: 10 M$ for 4 years if one can attack 2^{40} targets in parallel (success probability grows linearly with the number of targets)
- parallel collision search
  - n = 128: 10 M$ for 10 seconds (or 1 year on 10 GPUs)
  - n = 160: 10 M$ for 7 days
  - need 256-bit result for long term security (25 years or more)

Quantum computers

- in principle exponential parallelism
- inverting a one-way function: 2^n reduced to $2^{n/2}$ - but not parallizable and huge hardware requirements [Grover'96]
- collision search: can we do better than $2^{n/2}$?
  - $2^{n/3}$ computation + hardware [Brassard-Hoyer-Tapp'98] = $2^{2n/3}$
  - [Bernstein'09] classical collision search requires $2^{n/4}$ computation and hardware (= standard cost of $2^{n/2}$)





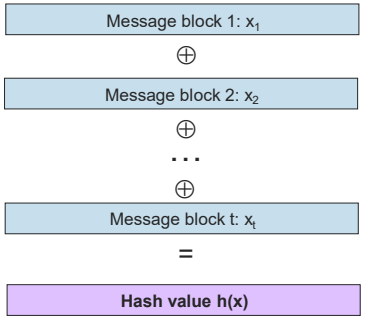
Properties in practice

- collision resistance is not always necessary
- other properties are needed:
  - PRF: pseudo-randomness if keyed (with secret key)
  - PRO: pseudo-random oracle property (indifferentiability)
  - near-collision resistance
  - partial preimage resistance (most of input known)
  - multiplication freeness
- how to formalize these requirements and the relation between them?

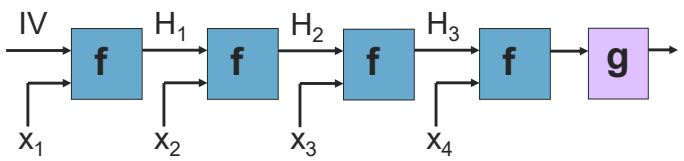
Iteration
(mode of compression function)

How not to construct a hash function

- Divide the message into t blocks x_i of n bits each



Hash function: iterated structure

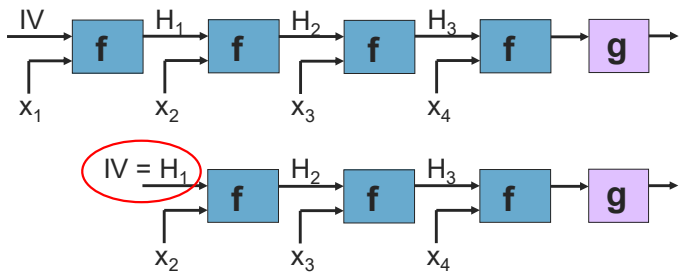


- split messages into blocks of fixed length and hash them block by block with a compression function f
- need padding at the end
- efficient and elegant…. but …

Security relation between f and h

iterating f can degrade its security

›› trivial example: 2nd preimage



Security relation between f and h (2)

› solution: Merkle-Damgård (MD) strengthening

›› fix IV, use unambiguous padding and insert length at the end

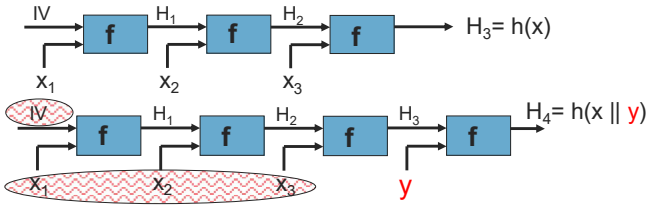
› f is collision resistant ~~⇒~~ ⇍ h is collision resistant [Merkle'89-Damgård'89]

› f is ideally 2nd preimage resistant ⇔ h is ideally 2nd preimage resistant [Lai-Massey'92]

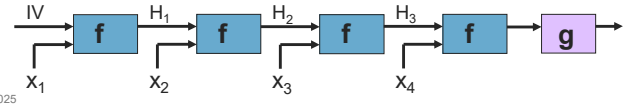
› many other results

Security relation between f and h (3)

length extension: if one knows h(x), easy to compute h(x || y) without knowing x or IV



solution: output transformation

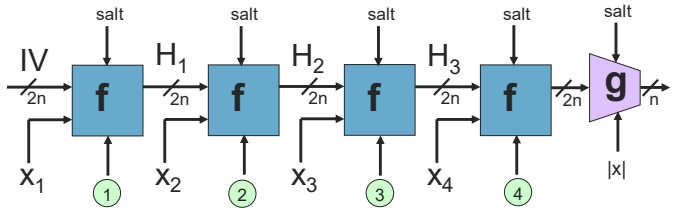


Attacks on MD-type iterations

- **long message 2nd preimage attack** [Dean-Felten-Hu'99], [Kelsey-Schneier'05]
  - Sec security degrades lineary with number 2^t of message **blocks** hashed: $2^{n-t+1} + t\,2^{n/2+1}$
  - appending the length does not help here!
- **multi-collision attack and impact on concatenation** [Joux'04]
- **herding attack** [Kelsey-Kohno'06]
  - reduces security of commitment using a hash function from 2^n
  - on-line 2^{n-t} + precomputation $2.2^{(n+t)/2}$ + storage 2^t

Improving MD iteration

salt + output transformation + counter + wide pipe



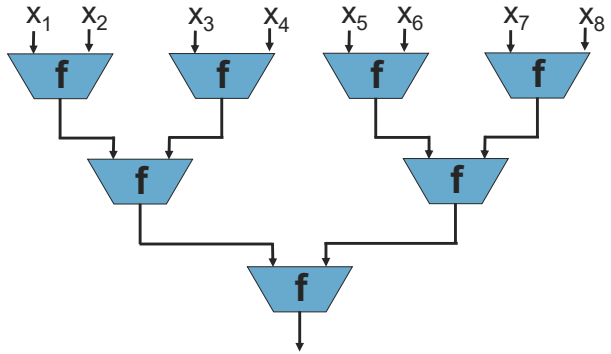
security reductions well understood
many more results on property preservation
impact of theory limited

Improving MD iteration

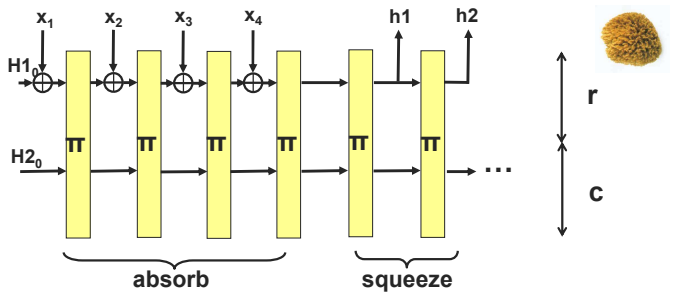
- degradation with use: salting (family of functions, randomization)
  - or should a salt be part of the input?
- PRO: strong output transformation g
  - also solves length extension
- long message 2nd preimage: preclude fix points
  - counter f → f_i [Biham-Dunkelman'07]
- multi-collisions, herding: avoid breakdown at $2^{n/2}$ with larger internal memory: known as wide pipe
  - e.g., extended MD4, RIPEMD, [Lucks'05]

Tree structure: parallelism
NIST Special Publication (SP) 800-185

[Damgård'89], [Pal-Sarkar'03], [Keccak team'13]



Permutation (π) based: sponge



if result has n bits, H1 has r bits (rate), H2 has c bits (capacity) and permutation π is "ideal"
collisions min ($2^{c/2}$, $2^{n/2}$)
2nd preimage min ($2^{c/2}$, 2^{n})
preimage min (2^{c} , 2^{n})

81

Modes: summary

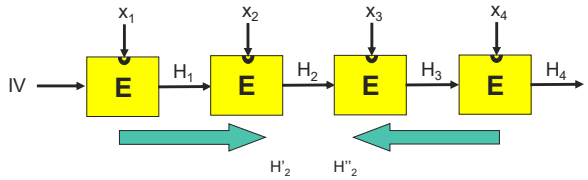
- growing theory to reduce security properties of hash function to that of compression function (MD) or permutation (sponge)
  - preservation of large range of properties
  - relation between properties
  - generic analysis: Sound hashing modes of arbitrary functions, permutations, and block ciphers [Daemen-Mennink-Van Assche'18] [Gunsing-Daemen-Mennink'20]
- MD versus sponge:
  - sponge is simpler
  - sponge easier to extend to authenticated encryption, MAC,..
  - should x_i and H_{i-1} be treated differently?
  - it is very nice to assume multiple properties of the compression function f, but unfortunately it is very hard to verify these

82

Compression functions

83

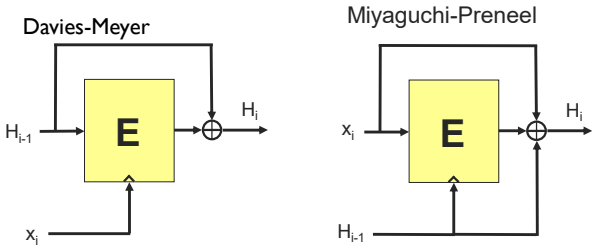
Single block length [Rabin'78]



- Merkle's meet-in-the-middle: (2^{nd}) preimage in time $2^{n/2}$
  - select $2^{n/2}$ values for (x_1,x_2) and compute forward H'_2
  - select $2^{n/2}$ values for (x_3,x_4) and compute backward H''_2
  - by the birthday paradox expect a match and thus a (2^{nd}) preimage

84

Block cipher (E_K) based: single block length

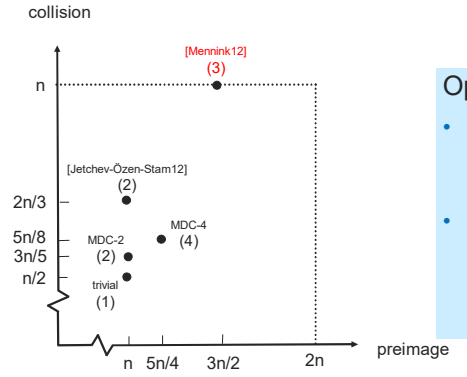


- output length = block length m; rate 1; 1 key schedule per encryption
- 12 secure compression functions (in ideal cipher model)
  - lower bounds: collision $2^{m/2}$, (2nd) preimage 2^m

[Preneel+'93], [Black-Rogaway-Shrimpton'02], [Duo-Li'06], [Stam'09],…

85

Block cipher (E_K) based: double block length (3n to 2n compression)



Open problems:

- what is the best collision/preimage security for 2 block cipher calls?
- For optimal collision security: what is the best preimage security for s block cipher calls? (upper bounds are known)

86

Iteration modes and compression functions

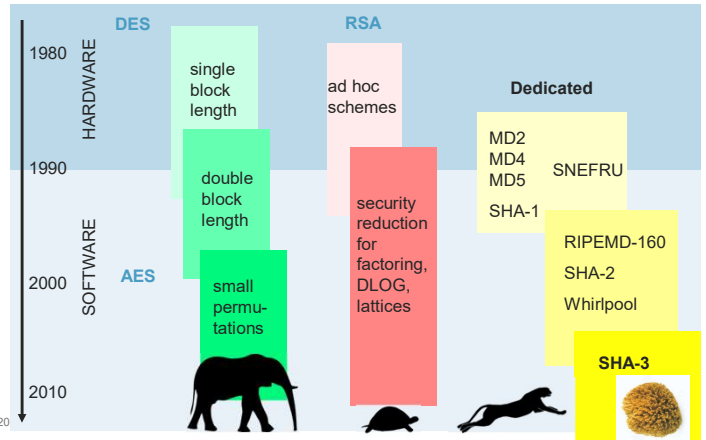
- compression functions are still made from permutations or keyed permutations (e.g. by dropping some bits)
- security of simple schemes well understood
- powerful tools available
- analysis of slightly more complex schemes very difficult

87

Hash function constructions

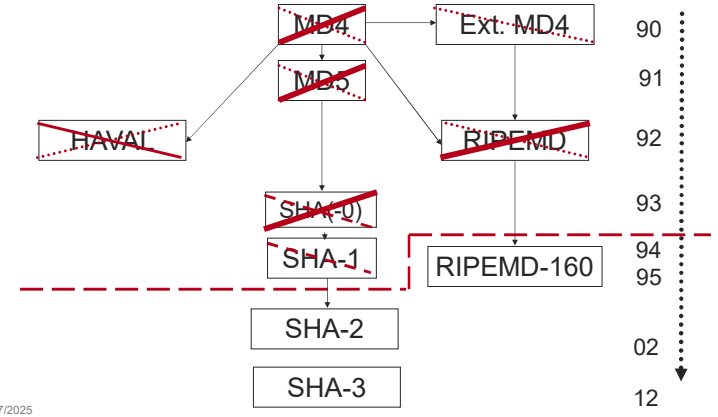
88

Hash function history 101



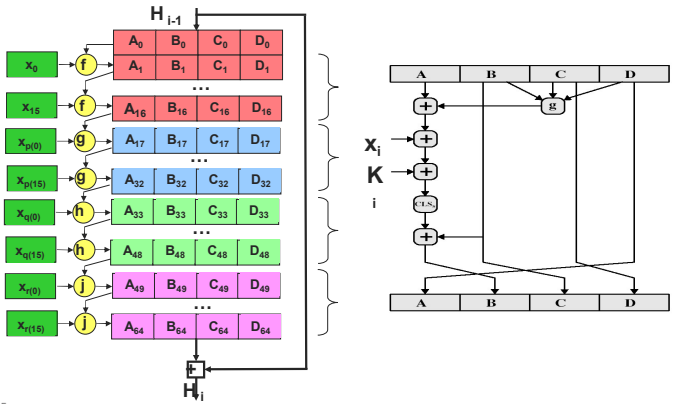
89

MDx-type hash function history



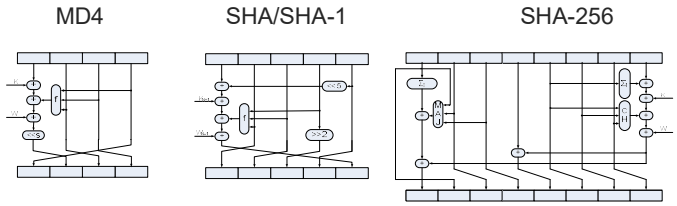
90

MD5: 4 rounds of 16 steps [Rivest'91]



91

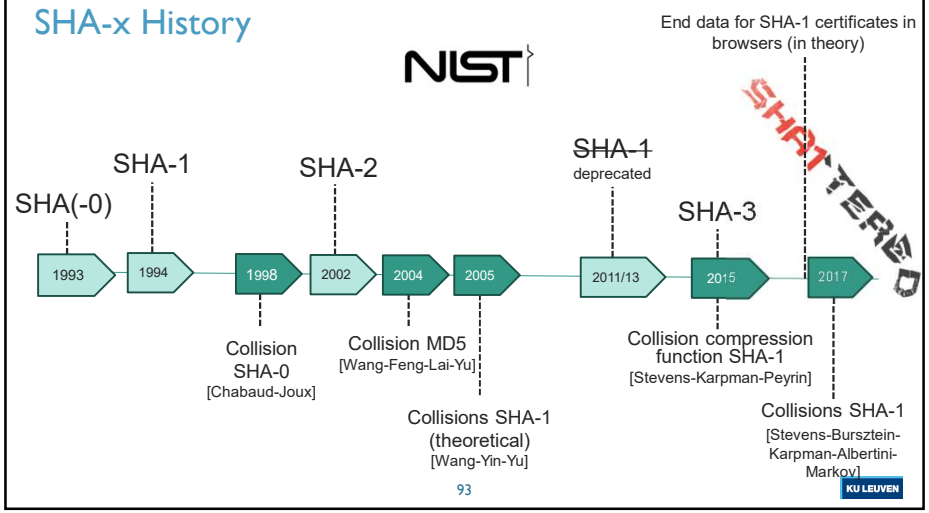
State updates in the MD4 family



Design principles copied in MD5, RIPEMD, HAVAL, SHA, SHA-1, SHA-2, ...
- All hash functions in use today except SHA-3

92

SHA-x History



93

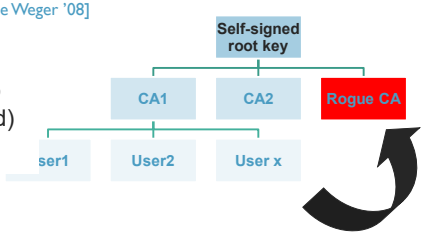
93

Rogue CA attack for MD5

[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



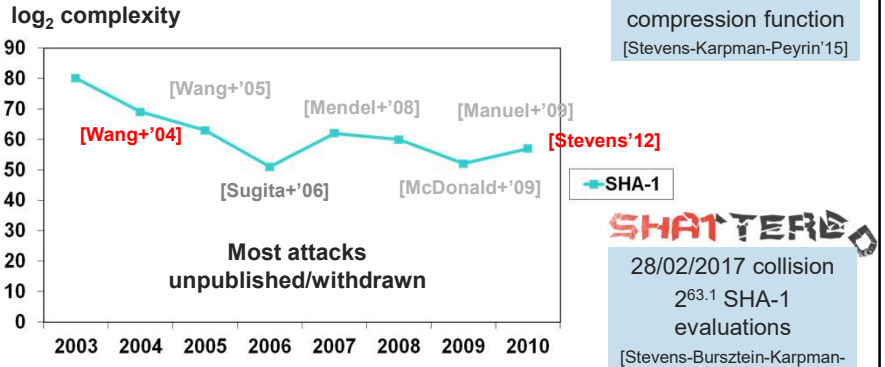
6 CAs have issued certificates signed with MD5 in 2008:
Rapid SSL, Free SSL (free trial certificates offered by RapidSSL), TC TrustCenter AG, RSA Data Security, Verisign.co.jp

TLS 1.2 (2008 but limited deployment until 2013): still allowed for MD5 certificates resulting in Sloth attack in 2015

24/07/2025

94

94



collision for compression function
[Stevens-Karpman-Peyrin'15]

SHATTERED

28/02/2017 collision
$2^{63.1}$ SHA-1 evaluations
[Stevens-Bursztein-Karpman-Albertini-Markov]

Prediction (June'16): collision for SHA-1 in the next 12 months

24/07/2025

95

95

SHA-2 : FIPS 180

designed by NSA, published in 2002

SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256

- non-linear message expansion
- SHA-384 and SHA-512: 64-bit architectures

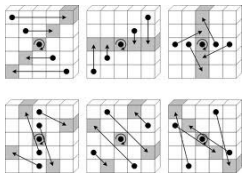
	Rounds	Collisions	Preimage	Free-start collision	Non-randomness
SHA-256	64	31 $2^{65.5}$	42 $2^{248.4}$	52 $2^{128-\varepsilon}$	47 easy
SHA-512	80	24 $2^{22.5}$	42 $2^{494.6}$	27 easy 46 $2^{254.5}$	

- implementations today faster than anticipated
  - 18 cycles/byte on Core 2 (2008) → 7.8 cycles/byte on Haswell (2013) → 7.8 cycles/byte (2023)
- adoption accelerated by other attacks on TLS
  - since 2013 deployment in TLS 1.2

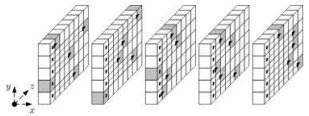
96

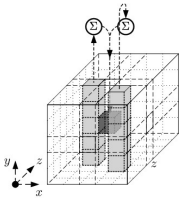
96

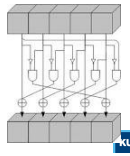
Keccak/SHA-3/FIPS 202



$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta, \text{with}$$







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

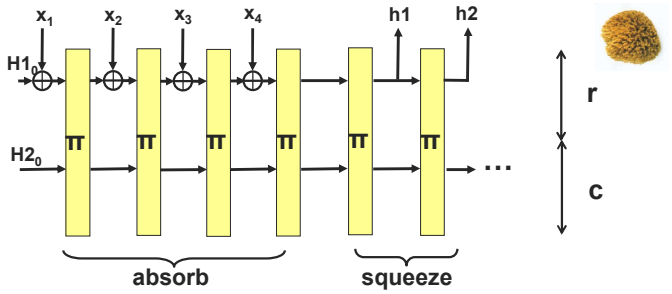
nominal version:

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

KU LEUVEN

97

Permutation (π) based: sponge



if result has n bits, H1 has r bits (rate), H2 has c bits (capacity) and
permutation π is "ideal" collisions min $(2^{c/2}, 2^{n/2})$
2nd preimage min $(2^{c/2}, 2^{n})$
preimage min $(2^{c}, 2^{n})$

24/07/2025

KU LEUVEN

98

Keccak/SHA-3/FIPS 202 (published: 5 August 2015)

- append 2 extra bits for domain separation to allow
  - flexible output length (XOFs or eXtendable Output Functions)
  - tree structure (Sakura) allowed by additional encoding
- 6 versions
  - SHA3-224: n=224; c = 448; r = 1152 (72%)
  - SHA3-256: n=256; c = 512; r = 1088 (68%)
  - SHA3-384: n=384; c = 768; r = 832 (52%)
  - SHA3-512: n=512; c = 1024; r = 576 (36%)

  } pad 01

  - SHAKE128: n=x; c = 256; r = 1344 (84%)
  - SHAKE256: n=x; c = 512; r = 1088 (68%)

  } pad 11 for XOF

if result has n bits, H1 has r bits (rate), H2 has c bits (capacity) and
permutation π is "ideal" collisions min $(2^{c/2}, 2^{n/2})$
2nd preimage min $(2^{c/2}, 2^{n})$
preimage min $(2^{c}, 2^{n})$

If c = 2n then collisions $2^{n/2}$ and (2^{nd}) preimage 2^{n}

KU LEUVEN

99

Keccak/SHA-3/FIPS 202: Very large security margin

	Rounds	Collisions	Preimage	Non-randomness
permutation	24			Zero-sum distinguisher for 18 rounds
SHA-3-256	24	5 easy 6 practical	4 2^{233} 5 2^{250}	
SHA-3-512	24	4 2^{237}	4 2^{467}	

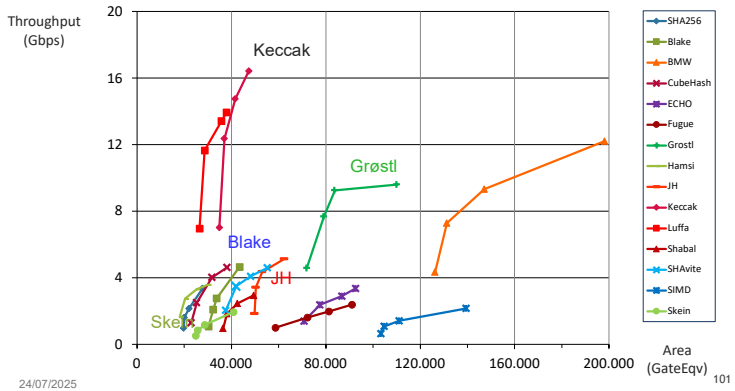
100

KU LEUVEN

100

Hardware: post-place & route results ASIC 130nm

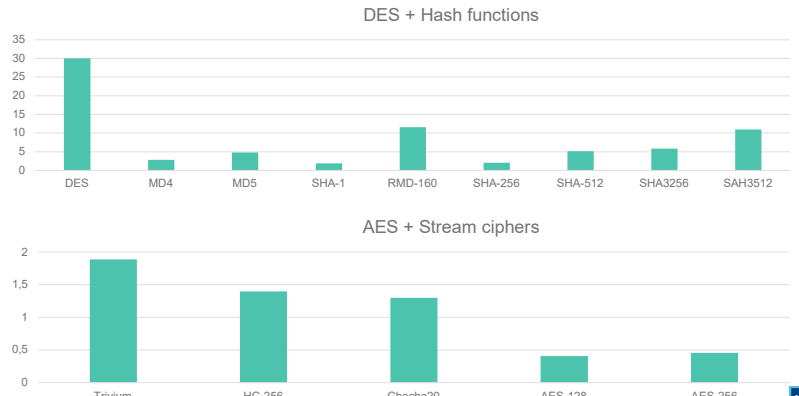
[Guo-Huang-Nazhandali-Schaumont'10]



101

Performance of hash functions, stream ciphers and block ciphers

[Bernstein-Lange] (cycles/byte) 2023 Intel Core i7-13700H, P cores; 6 x 4800MHz



102

Hash functions: conclusions

- SHA-1 would have needed 128-160 steps instead of 80
  - Even then migration by 2016 would have been needed
- 2004-2009 attacks: cryptographic meltdown but not dramatic for most applications
- Everyone uses SHA-2; hardware support will result in shift to SHA-03

- theory is developing for more robust iteration modes and extra features; still early for building blocks
- Nirwana: efficient hash functions with security reduction

103

The end

Thank you for your attention

104

Selected books on cryptology and applications

- 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 but outdated– not suited as a first text book. http://www.cacr.math.uwaterloo.ca/hac
- D. Boneh, V. Shoup, A Graduate Course in Applied Cryptography, https://toc.cryptobook.us/ Draft. Very advanced course with interesting applications.
- N. Smart, *Cryptography Made Simple*, Springer, 2015. Solid and up to date but on the mathematical side.
- D. Stinson, M. Peterson, *Cryptography: Theory and Practice*, CRC Press, 4th Ed., 2018. Solid introduction, but only for the mathematically inclined.
- Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography,* Chapman & Hall, 2014. Rigorous and theoretical approach.
- M. Rosulek, The Joy of Cryptography, https://web.engr.oregonstate.edu/~rosulekm/crypto/
- A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*, Princeton, 2016. Excellent introduction to the field.
- B. Schneier, *Applied Cryptography*, Wiley, 1996. Widely popular but no longer up to date– make sure you get the errata, online.
- P.C. van Oorschot, Computer Security and the Internet: Tools and Jewels, Springer, 2019. Brief chapters on cryptography, https://link.springer.com/book/10.1007/978-3-030-33649-3
- R. Anderson, Security Engineering, Wiley, 3rd Ed., 2020. Insightful. A must read for every information security practitioner. 2nd edition is available for free at http://www.cl.cam.ac.uk/~rja14/book.html
- W. Stallings, Cryptography and Network Security: Principles and Practice, Pearson, 8th Ed., 2022. Solid background on network security. Explains basic concepts of cryptography.

KU LEUVEN