

Криптографски хеш-функции (хеш алгоритми)

Доц. Стефка Буюклиева

Дефиниция

Хеш-функция е изчислително ефективна функция, изобразяваща двоична последователност с променлива дължина в двоична последователност с фиксирана дължина, наречена хеш-стойност (hash-value, hash-code, hash-result, hash).

Основна идея: компактно представяне на входната последователност, което наричаме още цифрова сигнатура, идентификатор, отпечатък или извлечение (message digest, digital fingerprint).

НАРЕДБА ЗА ИЗИСКВАНИЯТА КЪМ АЛГОРИТМИТЕ ЗА СЪЗДАВАНЕ И ПРОВЕРКА НА КВАЛИФИЦИРАН ЕЛЕКТРОНЕН ПОДПИС

Допълнителни разпоредби:

§ 1. По смисъла на наредбата:

1. "**Хеш алгоритъм**" е математически алгоритъм, чрез който от електронно изявление с произволна дължина необратимо се създава хеш идентификатор на изявлението.
2. "**Хеш идентификатор**" е число с фиксирана дължина, получено вследствие на прилагане на хеш алгоритъм спрямо определено електронно изявление.
3. "**Сигурен източник на случайни числа**" е метод за генериране на последователност от числа, в която всяко следващо число не може да бъде изчислено от предишните.
4. "**Метод за допълване (padding)**" е процес, при който се преобразува хеш идентификаторът на електронното изявление преди прилагане на съответния алгоритъм за създаване на квалифициран електронен подпис.

Приложение

- Криптиране
- Електронен подпис
- Съхраняване на пароли
- Контрол на целостта на данните
- Търсене на еднакви файлове

Някои бележки

Криптографска хеш-функция се счита за несигурна, ако е изчислително лесно (computationally feasible):

- Да се намери съобщението, което съответства на дадената хеш-стойност.
- Да се намерят колизии (collisions), т.е. различни съобщения с еднаква хеш-стойност.

Опонент, който може да се справи с едната от тези две възможности, може да я използва например за подмяна на съобщението.

В идеалния случай функцията би трябвало:

- да е свободна от колизии;
- да не може да се получи важна информация за съобщението по неговата хеш-стойност.

Свойства

- *Preimage resistant* (еднопосочна): по дадена хеш-стойност h да е трудно да се намери съобщение m , такава че $h = \text{hash}(m)$.
- *Second preimage resistant*: по даден вход $m1$ да е трудно да се намери друг вход $m2 \neq m1$, такъв че $\text{hash}(m1) = \text{hash}(m2)$. Това свойство се нарича още weak collision resistance.
- *Collision-resistant*: да е трудно да се намерят две различни съобщения $m1$ и $m2$, такива че $\text{hash}(m1) = \text{hash}(m2)$. Това свойство се нарича още strong collision resistance.

Примери

- 1) Контролна сума по модул 32 (32-битова сума на всички 32-битови думи в стринга) – не е еднопосочна;
- 2) $g(x)=x^2 \pmod{n}$, $n=prq$, p и q – прости числа – еднопосочна, но няма компресиращи свойства и не е устойчива на колизии.

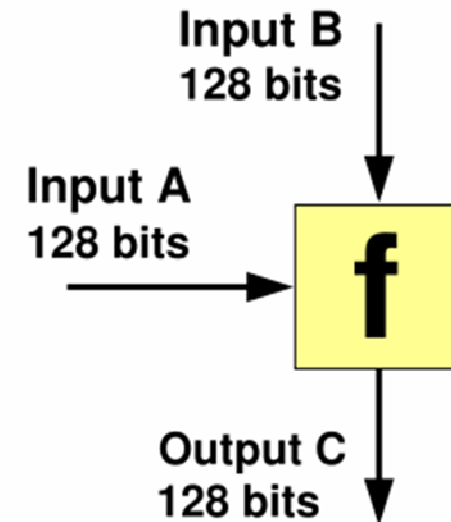
Еднопосочни функции

Дефиниция: *Еднопосочна (one-way function)* наричаме функция f , такава че за всяко x от дефиниционната ѝ област, $f(x)$ се пресмята лесно, но за почти всяко y от областта на стойностите на f е изчислително трудно да се намери x , за което $y = f(x)$.

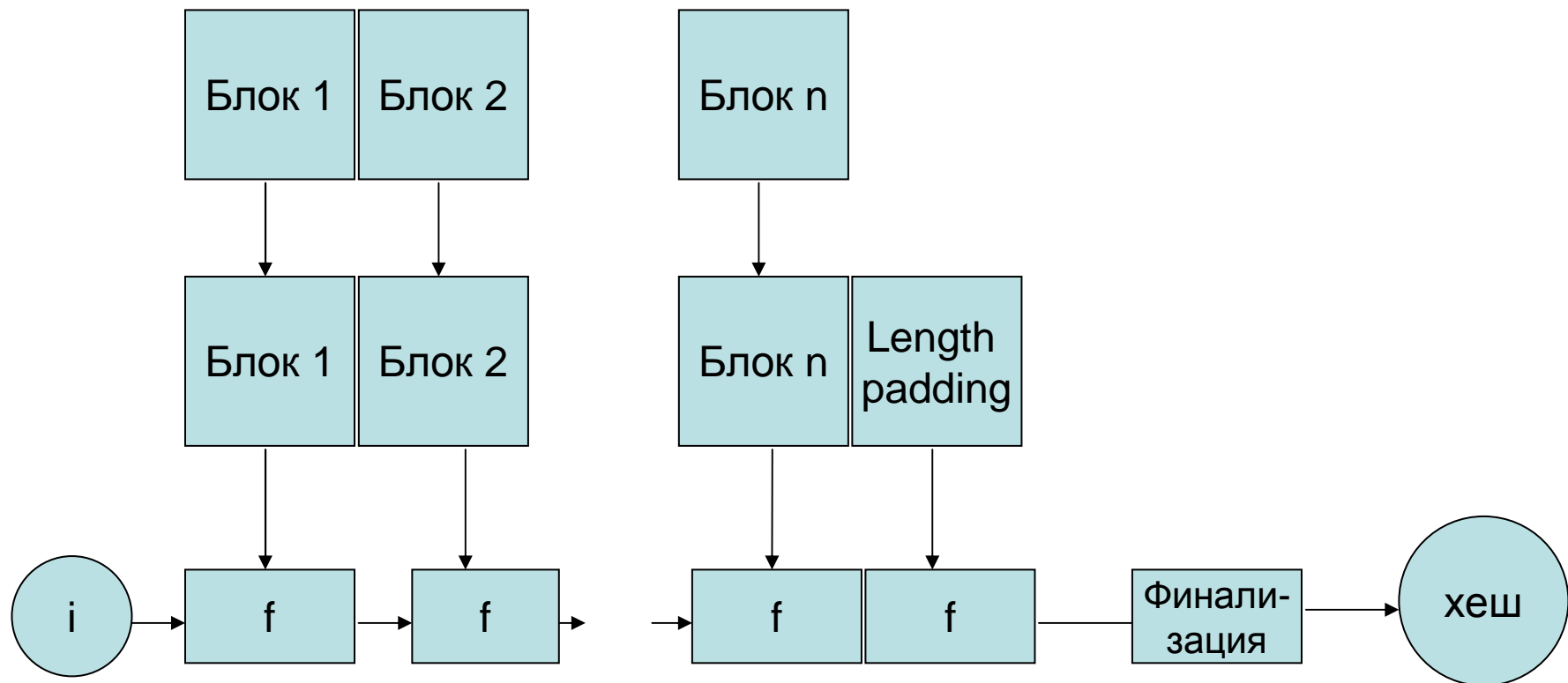
- Всъщност няма доказано еднопосочни функции (без никакви предварителни предположения).
- Въпреки познатите хеш-функции, които са доказано толкова сигурни, колкото **NP**-пълните (NP-complete) задачи, няма гаранция, че тези задачи няма да се окажат лесни в даден момент.
- Всички примери за “еднопосочни функции” могат да се квалифицират по-скоро като “предполагаемо” еднопосочни.
- Доказателство за съществуването на наистина еднопосочни функции би довело до доказване на $P \neq NP$, докато обратното (доказателство за несъществуване) би имало опустошителни последици за криптографията, но не води директно до извода, че $P = NP$.

Компресираща функция F

Компресиращата функция трансформира думи от два или повече входа с фиксирана дължина във една дума с дължина като на единия вход.



Merkle-Damgård construction



f – однопосочна компресираща хеш-функция

i – начален вектор

Итеративна конструкция

$\{x_1, x_2, \dots, x_r\}$ – блокове с фиксирана дължина

H_0 = initial value,

$H_i = f(H_{i-1}, x_i), 1 \leq i \leq r,$

$h(x) = g(H_r)$

Видове хеш-функции

- *MDC (modification detection code)* – не използват секретен ключ и се прилагат само за проверка на целостта на данните
- *MAC (message authentication code)* – използват секретен ключ и се използват за автентикация на източника на информацията

MDC

- MD4, MD5
- SHA-1 – дава като резултат 160 бита хеш-стойност
- RIPEMD-160
- DES in MDC mode – ключът не е таен

MAC

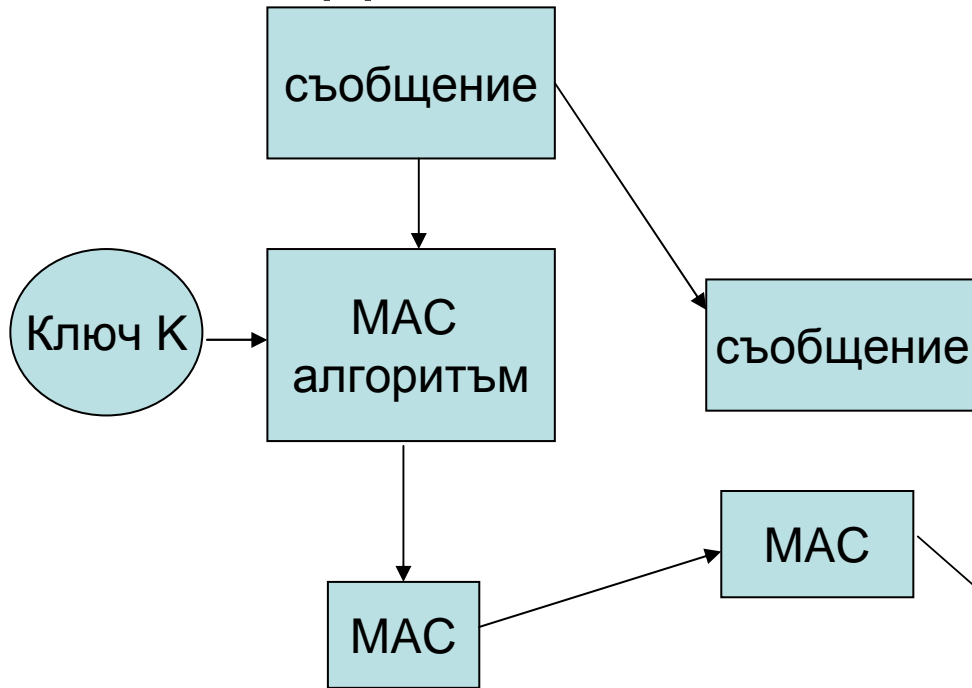
- DES in CBC mode
- Message Authenticator Algorithm (MAA) – 64 битов ключ
- MD5-MAC – ключ от 128 бита, от ключа и фиксирани константи се получават входните параметри за MD5

DES као MAC

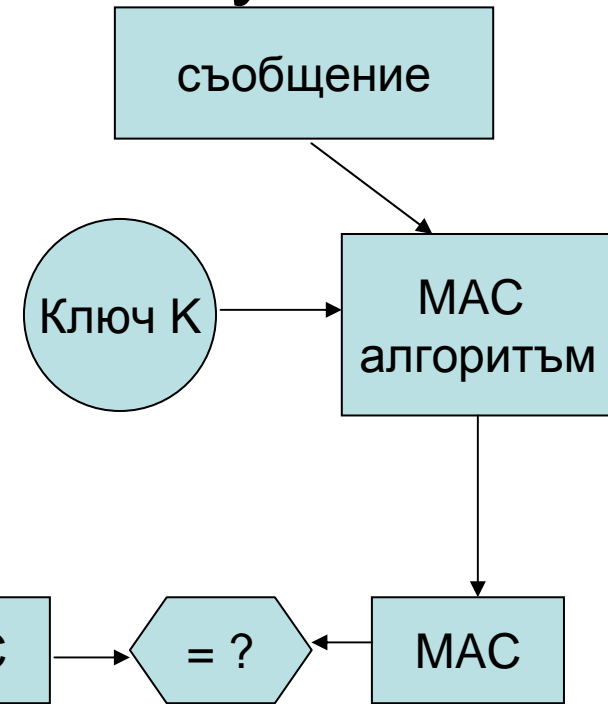
- **input:** binary string $\{a_1, a_2, \dots, a_L\}$, $L=64q$
- **initialize:** $h=(00\dots0)$ – 64 bits
- **for** $i=0$ to $q-1$ **do**
 - $h=\text{DES}(h \oplus (a_{64i+1}, a_{64i+2}, \dots, a_{64i+64}))$
- **Output:** hash value h

MAC

подател:



получател:



Хеш функции MD2, MD4, MD5

Създадени от Ronald Rivest:

- MD2 – 1989 година
- MD4 – 1990 година
- MD5 – 1991 година

Атаки:

Март 2005: Lenstra и Wang демонстрират два цифрови сертификата X.509 с различни публични ключове и еднакви MD5 извлечения

18 Март 2006: Владимир Клима демонстрира работещ алгоритъм за атака върху MD5

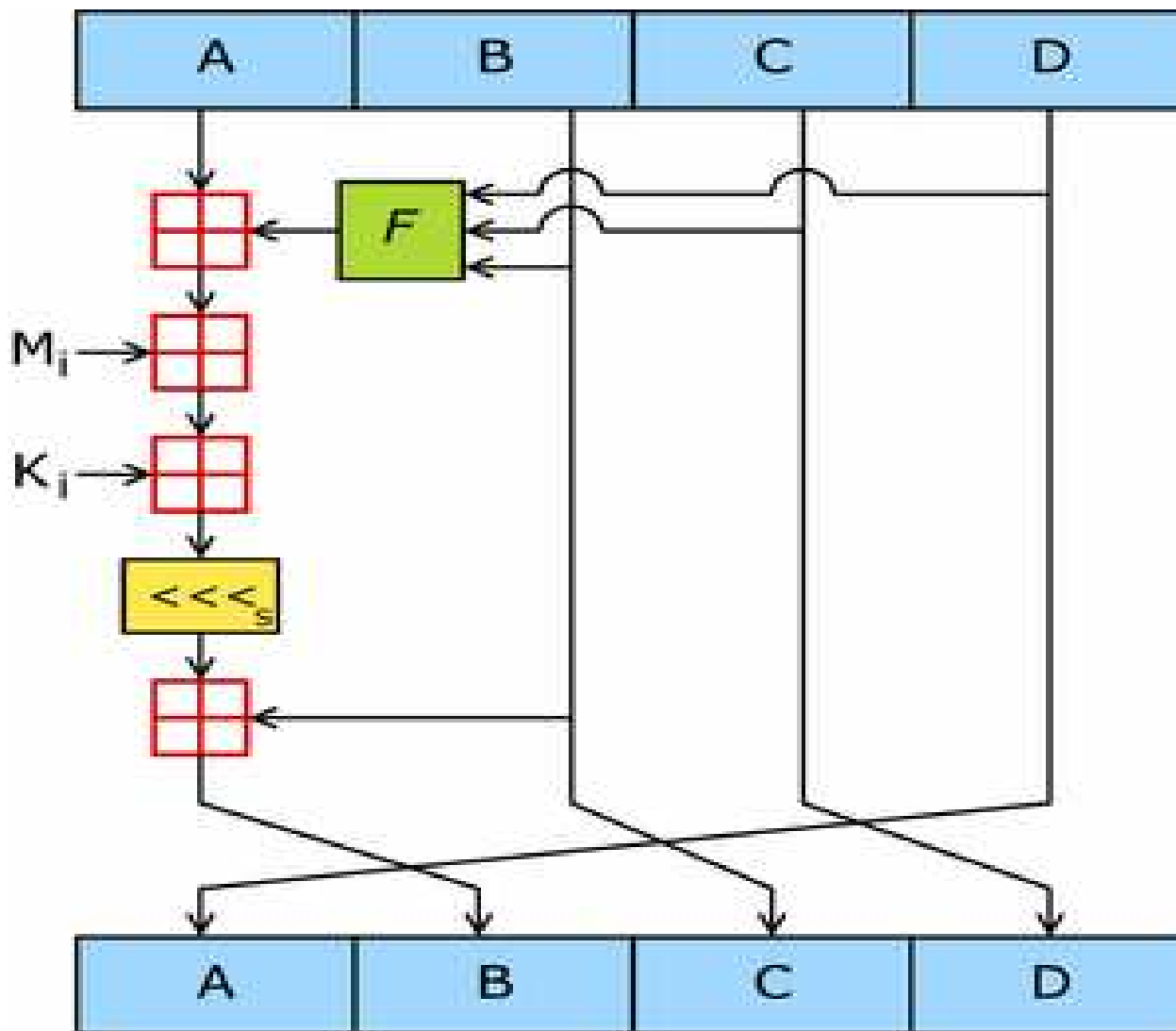
Приложения

- Трансфер на файлове
- При Unix дистрибуции и Windows приложения
- За съхраняване на пароли

АЛГОРИТЪМ

- Работи със 128 битова дума
- Допълване до 512 бита
- Четири 32 битови регистъра A, B, C, D
- Работи с операциите XOR, OR, AND и NOT

Алгоритъм



MD5

- **General Designers:** Ronald Rivest;
- **First published:** April 1992;
- **Series:** MD2, MD4, MD5, MD6;
- **Digest sizes:** 128 bits;
- **Structure:** Merkle–Damgård construction;
- **Rounds:** 4;
- **Best public cryptanalysis:** A 2009 attack by Tao Xie and Dengguo Feng breaks MD5 collision resistance using just 220.96 time. This attack runs in a few seconds on a regular computer.

НАРЕДБА ЗА ИЗИСКВАНИЯТА КЪМ АЛГОРИТМИТЕ ЗА СЪЗДАВАНЕ И ПРОВЕРКА НА КВАЛИФИЦИРАН ЕЛЕКТРОНЕН ПОДПИС

Изисквания за използваните алгоритми за квалифициран електронни подписи

1. Хеш-алгоритми:
 - 1.1. **SHA-1** (Secure Hash Algorithm);
 - 1.2. **SHA-2** (224, 256, 384, 512 бита);
 - 1.3. **RIPEMD-160** (Race Integrity Primitives Evaluation Message Digest) (256, 320 бита).
2. Не се допуска използването на следните хеш-алгоритми MD 4 и MD 5.
3. Допуска се използване и на други, различни от посочените в т. 1, хеш-алгоритми, които са с най-малко същото ниво на сигурност.
4. Дължината на идентификатора на електронното изявление следва да е не по-малка от 160 бита.

SHA (Secure Hash Algorithm)

SHA са 5 криптографски хеш-функции, конструирани от Националната агенция за сигурност на САЩ (NSA) и публикувани от Националния институт за стандарти и технологии (National Institute of Standards and Technology) като U.S. Federal Information Processing Standard (FIPS PUB 180). Тези алгоритми пресмятат хеш-стойност (редица от битове с фиксирана дължина, *message digest*) на входна редица (съобщение, *message*) с произволна дължина.

SHA-1, SHA-224, SHA-256, SHA-384, SHA-512

SHA-0 и SHA-1

- Оригиналните спецификации на алгоритъма са публикувани през 1993 като **Secure Hash Standard**, FIPS PUB 180, от NIST. Тази версия е известна като **SHA-0**. Скоро след това е изтеглена от NSA и заменена от ревизираната версия, публикувана през 1995, FIPS PUB 180-1, като **SHA-1**. **SHA-1** се различава от **SHA-0** само в един елемент на компресиращата функция. Скоро след това са забелязани слабости и в **SHA-0**, и в **SHA-1**. **SHA-1**.
- **SHA-1** (както и **SHA-0**) се базира на принципите, които са в основата и на функциите MD4 и MD5.

SHA

- SHA-1 дава “message digest” с дължина 160 бита.
- SHA-1 е внедрен в няколко широко разпространени приложения и протоколи, като TLS и SSL, PGP, SSH, S/MIME, IPsec.
- SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 са хеш-функциите, които се изискват по закон в някои приложения на правителството на САЩ, използващи криптографски алгоритми и протоколи за защита на некласифицирана информация. FIPS PUB 180-1 поощрява и използването на SHA-1 от частни и търговски организации.
- Основен мотив за публикуването на SHA е Digital Signature Standard, където е вложен.
- Хеш-функциите SHA са в основата на блоковите шифри SHACAL.

SHA-1

- **SHA-1 General Designers:** National Security Agency;
- **First published** 1993 (SHA-0), 1995 (SHA-1)
- **Series** (SHA-0), SHA-1, SHA-2, SHA-3
- **Digest sizes:** 160 bits;
- **Structure:** Merkle–Damgård construction;
- **Rounds:** 80;
- **Best public cryptanalysis:** A 2008 attack by Stéphane Manuel breaks the hash function, as it can produce hash collisions with a complexity of 2^{51} operations.

SHA 1 - пример

"The quick brown fox jumps over the lazy
dog"

= 2fd4e1c6 7a2d28fc ed849ee1 bb76e739
1b93eb12

"The quick brown fox jumps over the lazy
cog"

= de9f2c7f d25e1b3a fad3e85a 0bd17d9b
100db4b3

SHA-1 algorithm

Initialize variables:

h0 := 0x67452301

h1 := 0xEFCDAB89

h2 := 0x98BADCFE

h3 := 0x10325476

h4 := 0xC3D2E1F0

Pre-processing:

- Добавяне на бит '1' към съобщението
- Добавяне на k бита '0', където k е най-малкото естествено число такова че полученото съобщение има дължина (в битове) $\equiv 448 \pmod{512}$
- Представяне на дължината на съобщението (в битове) като 64-битово цяло число

- Съобщението се разглежда като редица от 512-битови блокове.
- Всеки блок се разбива на 16 на брой 32-битови думи $w[i]$, $0 \leq i \leq 15$.
- Тези 16 32-битови думи се разширяват до 80 32-битови думи:

for i **from** 16 **to** 79 $w[i] := (w[i-3] \text{ xor } w[i-8] \text{ xor } w[i-14] \text{ xor } w[i-16]) \text{ leftrotate } 1$

Инициализация:

$a := h0$

$b := h1$

$c := h2$

$d := h3$

$e := h4$

Основен цикъл:

for i from 0 to 79

if $0 \leq i \leq 19$ then

f := (b and c) or ((not b) and d)

k := 0x5A827999

else if $20 \leq i \leq 39$

f := b xor c xor d

k := 0x6ED9EBA1

else if $40 \leq i \leq 59$

f := (b and c) or (b and d) or (c and d)

k := 0x8F1BBCDC

else if $60 \leq i \leq 79$

f := b xor c xor d

k := 0xCA62C1D6

temp := (a **leftrotate** 5) + f + e + k + w[i]

e := d

d := c

c := b **leftrotate** 30

b := a

a := temp

Добавяне на текущия блок:

h0 := h0 + a

h1 := h1 + b

h2 := h2 + c

h3 := h3 + d

h4 := h4 + e

Final hash value:

digest = hash = h0 **append** h1 **append** h2 **append** h3 **append** h4

SHA sizes

Алгоритъм	Output size (bits)	State size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	collisions
SHA-0	160	160	512	$2^{64} - 1$	32	80	yes
SHA-1	160	160	512	$2^{64} - 1$	32	80	2^{63} attack
SHA-256/224	256	256	512	$2^{64} - 1$	32	64	None yet
SHA-512/384	512	512	1024	$2^{128} - 1$	64	80	None yet

Характеристики на някои хеш-функции

Име	Извлечение (в битове)	Цикли x брой стъпки за цикъл	Сравнителна скорост
MD2	128	3x16	1.00
MD4	128	4x16	0.68
RIPEMD- 128	128	4 x 16	0.39
SHA-1	160	4 x 20	0.28
RIPEMD- 160	160	5 x 16	0.24

RIPEMD-160

- Автори: Hans Dobbertin, Bart Preneel, COSIC research group, Katholieke Universiteit Leuven.
- Публикувана: 1996.
- Предложени са и **RIPEMD-128**, **RIPEMD-256**, **RIPEMD-320**.
- Няма патенти и може да се използва свободно.

SHA-1, 2, 3

Сигурността на SHA-1 е в известен смисъл компрометирана от криптографите. Въпреки че все още не са предложени добри атаки срещу вариантите на SHA-2, те са алгоритмично идентични с SHA-1. Затова се работи за разработването на по-добри хеширащи алгоритми. В момента върви конкурс за нова SHA-3 функция, който е записан във *Federal Register* на 2 ноември 2007.

DEPARTMENT OF COMMERCE

National Institute of Standards and Technology

[Docket No.: 070911510–7512–01]

Announcing Request for Candidate

Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA–3) Family

AGENCY: National Institute of Standards and Technology, Commerce.

ACTION: Notice and request for nominations for candidate hash algorithms.

Submissions are due October 31, 2008 and the proclamation of a winner and publication of the new standard are scheduled to take place in 2012.

NIST hash function competition

31 октомври 2008 – краен срок за изпращане на предложения;

9 декември 2008 – публикуван е официалният списък на приетите кандидати за първи тур;

февруари 2009 - NIST организира конференция, на която участниците са докладвали детайлите на техните алгоритми, а представители на NIST са обсъдили критериите за отсяване на кандидатите за втори тур;

24 юли 2009 - публикуван е официалният списък на приетите кандидати за втори тур 14 проекта;

август 2010 – проведена е втора конференция в University of California, Santa Barbara, където са дискутирани тези 14 проекта.

В края на 2010 са обявени 5 финалиста, а победителят ще бъде анонсиран през 2012 година.

SHA-3 финалисти

- **BLAKE** (Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael Phan)
- **Grøstl** (Praveen Gauravaram, Lars Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen)
- **JH** (Hongjun Wu)
- **Keccak** (Keccak team: Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche)
- **Skein** (Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas and Jesse Walker)

SHA-3 финалисти

- Performance: "A couple of algorithms were wounded or eliminated by very large [hardware gate] area requirement – it seemed that the area they required precluded their use in too much of the potential application space."
- Security: "We preferred to be conservative about security, and in some cases did not select algorithms with exceptional performance, largely because something about them made us 'nervous,' even though we knew of no clear attack against the full algorithm."
- Analysis: "NIST eliminated several algorithms because of the extent of their second-round tweaks or because of a relative lack of reported cryptanalysis – either tended to create the suspicion that the design might not yet be fully tested and mature."
- Diversity: The finalists included hashes based on different modes of operation, including the HAIFA and sponge hash constructions, and with different internal structures, including ones based on AES, bitslicing, and alternating XOR with addition.