Сравнителен анализ на съвременните криптографски HASH функции

П. Стоянов, Г. Бебров, М. Иванов, Н. Димитров

Comparative Analysis of the Advanced Cryptographic Hash Functions

P. Stoianov, G. Bebrov, M. Ivanov, N. Dimitrov

Department of Communication Engineering and Technologes, Technical University of Varna, 1 Studenska St., 9010 Varna, Bulgaria, pl_stoianov@tu-varna.bg, g.bebrov@tu-varna.bg, martinIvanov@tu-varna.bg, nikolay.dimitrov@tu-varna.bg

Key Words: *Cryptographic; hash-function; comparative analysis; MDC MAC; collision resistance; Message Digest.*

Abstract.*. The paper presents the fundamental role of hash functions in modern cryptography and Network security. To control the integrity of stored or exchanged data, cryptographic hash functions, also called OWHF (OWHF – One Way Hash Functions), are most often used. These functions are not used to protect the data, but to verify its identity.Hash Functions compresses a message of arbitrary length to a message of fixed length called hashcode or Message Digest (MD). According to the purpose they serve, there are two main types of hash functions - MDC and MAC. Keyed hash functions use secret key for computing the digest and these are also known as MAC (Message Authentication Code). The purpose of MAC is to authenticate the data source. MDC (Message Detection Codes) are un-keyed functions and serve to authenticate the data integrity. The three distinctive properties of hash functions are discussed: 1. Preimage resistance – for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output. 2. 2nd-preimage resistance – it is computationally infeasible to find any second input which has the same output as any specified input. 3. Collision resistance – it is computationally infeasible to find any two distanct inputs whish result to the same output. Defined the fields of application of the hash functions: 1. Data integrity check when exchanging data (file, string, database, etc.). 2. Storing access passwords on sites. 3. Digital signature. 4. Search for identical files and match of data. This paper presents a comparative analysys of hash functions that are curentlu used. The most frequently used functions (MD, SHA, WHIRLPOOL, RIPEMD, MASH-2 etc.) are presented in tabular form. A comparison was made according to some basic characteristics: input message size, output value, conversion method, logical operations used, operations used, etc. On this basis, can be made a decision to use a particular hash function for one purpose or another, according to the specific requirements for input message dimensionality, speed, and degree of security.*

1. Въведение

Думата криптография има гръцки произход и означава тайно писане (κρυπτός, *криптос* – **скрит**, и γράφω, *графо* – **пиша**). Задача на криптографията е преобразуване на открит текст (plaintext) в шифриран текст (ciphertext) чрез използване на криптографски алгоритъм и еднозначно възстановяване на оригиналният текст. Процесът на преобразуване в шифриран текст се нарича шифриране или криптиране (encryption), докато обратният процес на възстановяване е дешифриране или декриптиране (decryption). За контролиране на целостта на съхранявани или обменени данни най-често се използват криптографски hash функции, наричани още еднопосочни hash функции (OWHF – One Way Hash Functions). Тези функции не се използват за защита на данните, а за проверка на тяхната идентичност. От входно съобщение с определен размер функцията изчислява изходно съобщение с фиксиран размер, наречено hash стойност или цифрова сигнатура/извлечение (MD – Message Digest). Процесът на извличане на това съобщение се нарича **хеширане**.

Основна идея на хеширането е представянето на една голяма по обем последователност от входни данни с една много по-малка по размер стойност, която я представлява. Целта на този процес е по-малкия обем на хешираните стойности при обмен и съхранение и по-бързото сравнение на тези стойности. Hash функциите са от съществено значение за гарантиране на сигурността на цифровите подписи и проверките за целостта на данните. Те предоставят средства за проверка на автентичността и целостта на цифрови документи или съобщения, както и за откриване на всякакви неразрешени промени.

Според целта, която изпълняват, има два основни вида hash функции – MDC и MAC:

- **MDC (Message Detection Codes)** е за удостоверяване на целостта на данните. Когато се предават данни (примерно някакъв документ) се изчислява hash стойността и се предава заедно с данните. Получателят изчислява самостоятелно hash стойността на получените данни. Ако тези две стойности (получена и изчислена) съвпадат, може да се твърди с много голяма доза на сигурност, че

получените данни не са били променени. Ако тези стойности не съвпадат, със сигурност данните са били променени след предаването. Използването на съвременни алгоритми за хеширане гарантира, че най-малки промени в изходните данни (смяна на местата на два символа, смяна на главна и малка буква, вмъкване на пауза и други) води до много голяма промяна в изчислената hash стойност. Изчислението на hash стойности и тяхното сравнение гарантират липсата на неоторизирано подправяне на данните и тяхната надежност.

- **MAC (Message Autentication Codes)** е за удостоверяване на източника на данните. За разлика от MDC при MAC в създаването и проверката на hash стойността на данните участват личният и публичният ключ на предаващия (при използване на симетрични алгоритми е необходим само един ключ). Възможни са два варианта. Първият вариант е при изчисляване на hash стойността да се използва и личният ключ. По-често се използва вторият вариант (както е при цифров подпис DS – Digital Signature), където за входните данни (документ) се изчислява hash стойността чрез известен алгоритъм и се криптира с личния ключ. Получателят изчислява hash стойността чрез същата hash функция и дешифрира получената hash стойност с публичния ключ на предаващия. Ако тези две стойности съвпадат, се гарантира, че данните не са променяни след изчисляване на hash стойността и се потвърждава кой е източникът на данните.

Съществуват три отличителни свойства на hash функциите [1,2]:

1. **Preimage resistance (устойчивост/защита на входните данни).** При известна изчислена hash стойност y да е изчислително невъзможно да се намери някаква входна последователност *x*, т.е. *hash(x)=y*. Това защитава входните данни. При известна изчислена hash стойност да не може да се получи информация за входните данни, които са неизвестни. Също така да гарантира, че не може да се намери някаква входна последователност, която генерира известната hash стойност. В литературата се използва терминът one-way (еднопосочно действие).
2. **2nd-preimage resistance.** При известна изчислена hash стойност *y* и известни входни данни *x* изчислително невъзможно е да се намери друга входна последователност $x' (x \neq x')$, така че $hash(x) = hash(x') = y$. Различните входни данни с една и съща изчислена hash стойност се наричат колизии (collisions). Това свойство гарантира, че и най-малката промяна на входните данни (примерно ако е текст: замяна на малка и главна буква, вмъкване на интервал, смяна на цифра и други) ще доведе до различна изчислена hash стойност. В литературата се използва определението weak collision resistance (слаба защита от колизии).
3. **Collision resistance (устойчивост на колизии/различия).** Изчислително невъзможно е да се намерят две входни последователности *x* и *x'*, които да имат една и съща изчислена hash стойност, т.е. $hash(x) = hash(x')$. За разлика от (2) двете последователности *x* и *x'* са произволно избрани. Това свойство е известно като strong collision resistance (издръжлива/твърда защита от колизии).

Области на приложение на hash функциите

1. **Проверка на целостта на данните при предаване.** При обмен на данни (файл, стринг, база от данни и други) предаващата страна изчислява hash стойността на данните по определен алгоритъм и го предава заедно с данните. Приемната страна изчислява самостоятелно hash стойността на получените данни по известния алгоритъм и сравнява изчислената стойност с приетата. Ако тези стойности съвпадат, с много голяма вероятност може да се твърди, че данните не са били променени и тяхната надежност е гарантирана. Ако тези стойности са различни, е сигурно, че данните след предаването са променени и е извършена неоторизирана модификация. Няма значение дали е злоумишлено подправяне на данните (примерно файлът е инфилтриран с вирус) или е грешка при физическото предаване на данните.
2. **Съхранение на пароли за достъп в сайтове.** Вместо да се съхраняват паролите на потребителите, системите обикновено съхраняват изчислената hash стойност на паролата, а не самата парола. При заявка на потребителя за достъп системата изчислява hash стойността на въведената парола (в реално време) и сравнява изчислената стойност със съхранената. Ако двете стойности съвпадат, се дава разрешение за достъп. Ако някой неправомерно получи достъп до съхранените hash стойности на паролите, той не може да определи входните данни (паролата) и да получи достъп от този потребител. Използва се свойството на hash стойността, че е еднопосочна и необратима.
3. **Цифров подпис.** Криптирането на голям по обем файл (документ) е бавно и продължително. Предпочита се по-бързото хеширане на документа и криптиране на изчисленат hash стойност с личния ключ, което удостоверява целостта на документа. Оригиналният документ и криптираната hash стойност са общодостъпни. От оригиналния документ получателят изчислява hash стойността чрез използваната hash функция и декриптира тази стойност, из-

ползвайки публичния ключ на предаващия данните. При съвпадение на тези две стойности (предадена и изчислена декриптирана) се гарантира, че данните не са променени след предаването и е сигурно кой е източникът на данни. Това съвпадение има и реална правна стойност при оспорване.

4. **Търсене на еднакви файлове и съвпадение на данни в големи масиви.** При съхранение на файлове се изчислява hash стойността и тя се съхранява заедно с първоначалните данни. При необходимост от проверка дали този файл съществува в паметта, се изчислява hash стойността на търсения файл и се сравнява с вече записаните. При съвпадение на тези две стойности се предполага с много висока степен на вероятност, че този файл вече съществува. В този случай е необходима цялостна проверка на входните данни за наличие на съвпадение. По същия начин данните в големи масиви могат да се обединят по определен критерий и да се изчисли тяхната hash стойност. Проверката за съвпадение на hash стойности с определен размер е много по-бърза от проверката на входни данни с голям обем данни и различен размер.

Основни свойства на hash функциите

1. Лесно и бързо изчисляване на hash стойността hash(m) = h.
2. При известна изчислена hash стойност не може да се извлекат данни за входната последователност.
3. Изчислително невъзможно е създаването на съобщение по известна hash стойност.
4. Изчислително невъзможно е намирането на различни входни данни, които създават една и съща hash стойност.
5. Много малки промени на входните данни водят до значителни промени в изчислената hash стойност.

Целта на статията е да се направи основен преглед на съвременните hash функции, които се използват в момента. Разглеждат се двата основни вида – MDC и MAC, свойства на hash функциите и области на приложение. Извършен е сравнителен анализ на използваните hash функции: според използваните логически операции, според броя операции/стъпки, използвани константи, размерност на изчислената hash стойност и други.

2. Съвременни hash функции

Според стандарта на ISO (International Organization for Standardization) и IEC (International Electrotechnical Commission) [3] основните използвани hash функции са SHA (различни модификации), RIPEMD, WHIRLPOOL, STREEBOG и SM3. За да се получи представа за използваните hash функции, трябва да се разгледат и други видове като MD, BLAKE, MASH и други.

2.1. Фамилия Hash функции MD (Message Digest)

Има различни модификации: MD2, MD4, MD5, MD6, разработени от Ronald Rivest. Първите създадени (без MD6) са основани на преобразуване Merkle-Damgard [2], предложено от Ralph Merkle и Ivan Damgard. При тази схема входното съобщение се разделя на блокове с определена дължина и се допълва със символи (padding) до кратен размер на блоковете. На всеки етап се извършва компресиране на данните от два входа (или повече) и получаване на изходни данни с размерност на входните. Преобразуването се извършва на стъпки колкото е броят на входните блокове. Използва се начална стойност (входен вектор) и последователно се обработват входните блокове от данни. След финалната обработка се получава изчислената hash стойност на входните данни с определена размерност. MD2 преобразува произволна входна последователност в 128 битова изходна hash стойност. Входната последователност се разделя на блокове от 16 байта и към тях се добавя контролна сума. Тази сума зависи от входните данни и предварително зададена 256 битова таблица за замяна (S-box). Получената последователност се обработва на стъпки чрез сумиране по mod2 със съответната стойност на S-box за получаване на изходната hash стойност.

MD4 обработва 512-битов блок и използва три функции, реализирани чрез логическите операции и, или, mod2 и ротация на 32-битови данни в три етапа. Изчислената hash стойност е също 128 бита както при MD2, но липсват S-box. MD5 има малки разлики от MD4. Броят на етапите е увеличен на четири и са добавени допълнителни константи. Има промяна в логическата обработка на функциите и е добавена четвърта функция. Последната предложена версия MD6 [4] преобразува входна последователност до 26^4 бита в изходна стойност до 512 бита. MD6 дава възможност и за директно използване като MAC, тъй като е възможно използването на до 512 битов ключ. Броят на отделните етапи на обработка е до 168 в зависимост от размера на изчисляваната hash стойност. MD6 използва дървовидна (разклонена) структура (Merkle tree), която предоставя възможност за паралелни изчисления с повишено бързодействие. MD6 притежава висока степен на сигурност и е предложена за новия стандарт SHA-3, но е изтеглена преди финалния етап поради по-ниското бързодействие спрямо останалите предложения.

2.2. Hash функция RIPEMD

Създадена е по проекта на EU RIPE (**R**ACE **I**ntegrity **P**rimitives **E**valuation) през 1992 г. и към абревиатурата се добавя **MD** (Message Digest). Предложена е фамилия от четири функции: RIPEMD -128, -160, -256 и -320. Разли-

чават се по размерността на изчислената hash-стойност съответно 16, 20, 32 и 40 байта. Също така разликите са в броя на използваните цикли (round) при преобразуването, различните пермутации, брой на преместванията (shift), използваните константи и логическите функции при обработка на отделните блокове от входното съобщение. Създадени са на основата на MD4 и MD5 и използват схемата на Merkle-Damgard [2]. Най-често използвана е RIPEMD -160 [5] примерно стандарти при Bitcoin. Предимство на RIPEMD е, че не е лицензирана и може да се използва свободно в различни приложения. Не се препоръчва нейното използване (както и на MD4, MD5, SHA-1) поради публикувани възможности за атака [6], при които се получават различни входни съобщения с еднаква изчислена hash стойност (collision resistance). Поради посочените предимства RIPEMD все още може да се използва, като същевременно се отчита необходимото време за атака (получаване на входно съобщение при известна изчислена hash стойност) според публикуваните разработки и анализи.

2.3. Hash функция WHIRLPOOL

Името (преведено като водовъртеж) е вдъхновено от астрономията с първата открита галактика със спираловидна структура (M51a или NGC 5194). Тази hash функция не е патентована и може да се използва свободно за всяко приложение [7]. Използва структура, подобна на симетричен криптографски алгоритъм Rijndael, разработен от белгийски криптографи (Vincent Rijmen участва при създаването на WHIRLPOOL и Rijndael), стандартизиран като AES (Advanced Encryption Standart) чрез FIPS PUB 197. Входно съобщение с максимална размерност 2^{256} бита се преобразува в 512 битова изходна hash стойност. Входните данни се разделят на блокове по 512 бита и се разглеждат като матрица 8x8 байта (8x8x8 = 512 бита). Във всеки блок се извършват аритметични действия с полиноми в поле на Галоа GF (2^8) (GF-Galois Field). Използват се основни логически Булеви функции и изчисления в GF за получаване на новата стойност. Изчисленията на полиноми в GF е бавно, поради което в практиката се използват предварително изчислени стойности и записани в таблици за замяна S-box (Substitution box). Записът на предварително изчислени константи в ROM паметта води до увеличаване на използваната памет, но се увеличава бързодействието при преобразуване. Отделните блокове от входното съобщение се преобразуват последователно, като се използва изчислената стойност от предишния блок. След последното преобразуване се получава крайната изчислена hash стойност на входното съобщение. Предложената hash функция WHIRLPOOL е удобна за реализиране на процесори с различна размерност от 8 до 64 битови [7]. До този момент няма данни за успешна атака, освен при намален брой цикли при изчисление на отделните блокове (в оригиналния вариант циклите са 10 за всеки блок).

2.4. Hash функция STREEBOG

Тази функция е руски стандарт, разгледан в ISO/IEC [3]. Името (Стрибог) произлиза от славянската митология като бог или божество на вятъра, който доставя богатство и благоденствие. Тази hash функция заменя по-стария стандарт GOST R 34.11-94 с нов стандарт GOST R 34.11-2012, за да отговори на новите изисквания, заложени при SHA-3. Организацията е подобна на hash функция WHIRLPOOL. Входното съобщение се разделя на блокове по 512 бита, където се преобразуват данните в GF както при симетричен криптографски алгоритъм AES. Разликата е, че във всеки блок се извършват 12 цикъла на преобразуване (при WHIRLPOOL са 10) и има два варианта: 256 и 512 битова hash стойност. Двата варианта са идентични при преобразуването, но имат различно първоначално състояние и различна по размерност изходна hash стойност. Както и при WHIRLPOOL няма данни за успешна атака, освен при намален брой цикли на изчисления в блоковете на входното съобщение [8].

2.5. Фамилия Hash функции SHA

SHA (Secure Hash Algorithms) е фамилия от криптографски функции, стандартизирани от NIST (National Institute of Standards and Technology) и публикувани чрез FIPS PUB 180 (Federal Information Processing Standarts PUBlication) като правителствен стандарт за обработка на информация [9,10]. Стандартизирани са четири различни алгоритъма: SHA-0, SHA-1, SHA-2 и SHA-3.

2.5.1. Hash функция SHA-0

Оригиналната версия е публикувана през 1993 г. под името SHA, но малко след това е оттеглена поради открити недостатъци. Разработена е на основата на MD4 с изходна изчислена hash стойност 160 бита. Поради доказани открити колизии не се препоръчва нейното използване.

2.5.2. Hash функция SHA-1

SHA-1 е с малки разлики от SHA-0. Проектирана е за национална агенция за сигурност NSA (National Security Agency) като част от алгоритъм за цифров подпис DSA (Digital Signature Algorithm) и е стандартизирана чрез FIPS PUB 180 -2 [9] . Използвана е в различни протоколи и приложения за хеширане на съобщения. Разработена е на основата на MD4 и MD5. Входното съобщение с максимална размерност 264-1 бита се разделя на блокове от 512 бита. Последният блок се допълва със символи (padding) до кратен размер на блоковете, като последните 64 бита се запазват за размерността на входното съобщение. Блоковете се обработват последователно, като се използват данните от преобразуването на предишния блок до получаване на изходната hash стойност от 160 бита. До този момент има предложени няколко разработки за получаване на колизии с нама-

лен брой цикли при изчисление на отделните блокове (достигнати са 73 при оигинални 80). Публикувани са теоретични разработки за получаване на колизии при пълен брой цикли, но те изискват много време и голям изчислителен ресурс. Поради възникналите съмнения организациите, използващи SHA-1, препоръчват преминаване към SHA-2 и SHA-3.

2.5.3. Hash функция SHA-2

SHA-2 има малки алгоритмични разлики от SHA-1, но се различава по размерността на изходната изчислена стойност 224 до 512 бита (при SHA-1 е 160 бита). При двата варианта се обработва различен по размерност входен блок: 512 или 1024 бита. При първия вариант може да се получи 224 или 256-битова изходна изчислена hash стойност, а при втория вариант (обработка на блок от 1024 бита) – 224, 256, 384 или 512 бита [10].

2.5.4. Hash функция SHA-3

През 2014 г. след продължителни обсъждания NIST избира алгоритъм Keccak за SHA-3 [11]. Този алгоритъм има *spong* (гъба, сюнгер) структура с два етапа и е принципно различен от разгледаните MD варианти. На първия етап се преобразуват входните данни, а на втория се получава hash стойността. Данните се разделят на 25 клетки, разположени като матрица от 5 реда и 5 стълба и може да се разглежда триизмерно (x,y,z) заедно с размерността на клетките. Всяка клетка може да съдържа 1, 2, 4, 8, 16, 32 или 64 бита. Между клетките се извършват логическите операции или, и, инверсия, сума по mod 2 и ротация, но без използване на S-box. Данните се обработват на няколко еднакви етапа, чиито брой зависи от размера на клетката. В края на всеки етап резултатът се сумира с константа, различна за всеки етап. Изходната стойност може да е различна (128 до 512 бита), както и размерността на обработвания блок (576 до 1600 бита).

2.6. Hash функция MASH

Съществуват hash функции основани на модулна аритметика. Приложението им е по-ограничено, но са интересни поради начина за обработка на входната информацията. Пример за такава е MASH-1 (Modular Arithmetic Secure Hash, algorithm 1) стандартизиран от ISO/IEC10118-4 [12]. Входната последователност P се разделя на t блока с размерност n бита, след което изходната hash стойност – H се изчислява чрез:

$H_i = H_{i-1} \oplus \text{Right}(n)(((H_{i-1} \oplus p_i) \vee A)^e \bmod M$, където A е предварително зададена константа-0xF0..0, e .= 2, v е логическо "или", Right(n) отделя десните n бита от резултата и стъпките са i = 1. ÷ t+1. Поради недостатъчна сигурност на алгоритъма [13] в следващата версия MASH-2 степента е се променя на $e = 2^8 + 1$.

3. Сравнителен анализ и препоръки

Табл. 1. Характеристики на използвани hash функции

Hash функции / Характеристики	MD5	MD6	SHA-2	SHA-3	RIPEMD-160	WHIRLPOOL	MASH-2
Входно съобщение max (bit)	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$ или $2^{128}-1$	Няма ограничение	$2^{64}-1$	$2^{256}-1$	Няма ограничение
Обработван блок (bit)	512	1024	512 или 1024	До 1600	512	512	M-1, M е избран модул
Брой на циклите в блок	64	до 168	64 или 80	24	80	10	Входен блок/n
Изходна hash (bit)	128	до 512	от 224 до 512	224 до 512	160	512	n < M – 1
Основни операции	⊕ , и, или, не, ротация mod 2^{32}	⊕ , и, или, не, ротация	⊕ , и, или, не, ротация, mod 2^n	⊕, и, не, ротация	⊕ , и, или, не, ротация mod 2^{32}	⊕, ротация S-box	⊕ , или, степен, mod M
Възможност за MAC	Не	Да	Не	Не	Не	Да	Не
Метод на преобразуване	Merkle-Damgard	Merkle tree	Merkle-Damgard	Spong	Merkle-Damgard	Galois Field	Модулна редукция
Константи	64x32 (bit)	1 (64 bit)	64x32 или 80x64 (bit)	24	8	S-box 16x16 byte	1

В *табл. 1* са разгледани едни от най-често използваните hash функции. Представен е сравнителен анализ според няколко основни характеристики. При повечето функции има ограничение на размерността на входното съобщение, тъй като в алгоритъма са предвидени определен брой битове за размера на това съобщение преди преобразуването му. На практика това няма голямо значение, защото дори и най-малкото предвидено входно съобщение при hash функциите $2^{64}-1$ е по-голямо от възможното съобщение за хеширане в момента (2^{40} bit = 1Tbit). При голяма част от използваните hash функции изходната изчислена стойност е с точно оп-

ределена разредност до 512 бита. За промяна на разредността е необходима и промяна в алгоритъма. При MD6 и SHA-3 промяната на разредността на изходната стойност е заложена в алгоритъма: при MD6 е през 1 бит, при SHA-3 – през 25. Засега 512 битова изчислена hash стойност е достатъчна за осигуряване на сигурност срещу атаки, но при необходимост от по-висока разредност тези два алгоритъма предоставят лесна възможност за промяна съобразно новите изисквания. Логическите операции и, или, не, сума по mod2 се използват при всички алгоритми, като времето им за изпълнение е съизмеримо. По-бавно е изпълнението на операции преместване и ротация особено при изпълнение на процесори с по-ниска разредност от разредността на обработвания блок, тъй като се изпълняват отделни операции с отчитане на преноса. Изчислението на mod 2^n е бързо, тъй като се отделят последните n бита на резултата. Изчислението mod M (както при MASH) е много по-бавно, защото M е произволно число (използва се идеята на асиметричен криптографски алгоритъм RSA). Теоретично всяка hash функция може да се използва в режим MAC, като при допълване на входното съобщение (padding) се вмъкне и личният ключ на предаващия. При малко от функциите, както е при MD6 и WHIRLPOOL, е предвидена алгоритмична възможност за използване на 512-битов ключ.

Таблиците за замяна (S-box) се използват като нелинейни функции при обработка на данни. Аритметичните действия при изчисления на матрици и полиноми изискват много време за обработка. За повишаване на бързодействието изчисленията се извършват предварително и се записват в таблици. Така входната стойност се заменя директно с вече предварително изчислената стойност. Това повишава използваната ROM памет за съхранение на данни, но увеличава бързодействието.

Като константи се използват определени числа или изрази, които трудно могат да се изчислят (дробната част на *e*, π, √2, √3, изчисления на sin(i) и други). В практиката те предварително се изчисляват и се записват като константи с цел по-бързо преобразуване. Използването на известни числа и проверими изрази, а не произволни стойности, гарантира, че при проектирането на алгоритъма не са заложени възможности за нерегламентирано предимство при преобразуването (така наречените *вратички* в алгоритъма).

4. Заключение

В статията е извършен сравнителен анализ на hash функциите, които се използват в момента. Разгледани са двата основни вида – MDC и MAC, и трите основни свойства на hash функциите: preimage resistance, 2nd-preimage resistance и Collision resistance. Описани са областите на приложение на hash функциите при проверка на целостта на данните при обмен, търсене за еднаквост на файлове в големи масиви, съхранение на пароли и в системите за цифров подпис. В табличен вид са представени най-често използваните функции (MD, SHA, WHIRLPOOL, MASH-2 и RIPEMD). Извършено е сравнение според някои основни характеристики: входен размер на съобщението, изходна стойност, метод на преобразуване, използвани логически операции, използвани операции и други. На тази основа може да се вземе решение за използване на определена hash функция за една или друга цел според специфичните изисквания за размерност на входното съобщение, бързодействие и степен на сигурност. Като препоръка за следващи изследвания е включване и на други съвременни функции, например STREEBOG, основана на GF, и криптографски алгоритъм AES, за които няма публикувани успешни атаки. Особен интерес предизвикват новите предложения (извън метод на Merkle-Damgart) като MD6 и SHA-3. Те предлагат иновативни алгоритми с големи възможности за промени и висока гъвкавост. SHA-3 може да се изпълнява ефективно на 8 до 64-битови платформи. При MD6 изходната стойност може да се променя през 1 бит, но са необходими допълнителни решения за повишаване на бързодействието при преобразуване.

Литература

1. Rogaway, P., T. Shrimpton. Crypttographic Hash Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second Resistance, Second-Preimage Resistance, and Collision Resistance. Fast Software Encryption. Lecture Notes in Computer Science, 3017, Springer-Verlag, Berlin, 2004, 371-388.

2. Katz, J., Y. Lindell. Introduction to Modern Cryptography. CRC Press Taylor & Francis Group, 2015.

3. ISO/IEC 10118-3:2018, IT Security Techniques, Hash-functions Part 3: Dedicated Hash-Functions. https://www.iso.org/standard/67116.html.

4. Rivest, R. The MD6 Hash Function A Proposal to NIST for SHA-3, 2010. https://citeseerx.ist.psu.edu.

5. Dobbertin, H., A. Bosselaers, B. Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. Fast Software Encryption. Third International Workshop, Cambridge, UK, 1996, 71-82. https://link.springer.com/chapter/10.1007/3-540-60865-6_44.

6. Yinghin, L., L. Fukang, G. Gaoli. Automating Collision Attacks on RIPEMD-160. – *IACR Transactions on Symmetric Cryptology,* 2023, 4, 112-142. https://tosc.iacr.org/index.php/ToSC/article/view/11282.

7. Barreto, P., V. Rijmen. The Whirlpool Hashing Function. Submitted to NESSIE, May 2003.

https://www.researchgate.net/publication/228610491_The_Whirlpool_hashing_function.

8. Ma, B., B. Li, R. Hao, X. Li. Improved Cryptanalysis on Reduced-Round GOST and Whirlpool Hash Function. Applied

Cryptography and Network Security, ACNS 2014, 289-307. https://link.springer.com/chapter/10.1007/978-3-319-07536-5_18.

9. FIPS PUB 180-2. Secure Hash Standart, Federal Information Processing Standart (FIPS), NIST, US Department of Commerce

http://csrc.nist.gov/publications/.

10. FIPS PUB 180-4. Secure Hash Standart, Federal Information Processing Standart (FIPS), NIST, US Department of Commerce,

2015. https://csrc.nist.gov/publications/fips.

11. Bertoni, G., J. Daemen, M. Peeters, G. Assche, R. Keer. The Keccak Implementation Overview, Version 3.2, 2012.

12. SO/IEC 10118-4:1998/Amd 1:2014 Information Technology Security Techniques Hash-functions Part 4: Hash-functions Using Modular Arithmetic. https://www.iso.org/standard/62543.html.

13. Antipkin, V. Smashing MASH-1. IACR Cryptology ePrint Arhive 2013:589, 2013.

За контакти:
Гл. ас. д-р инж. **Пламен Стоянов**
pl_stoianov@tu-varna.bg
Гл. ас. д-р инж. **Георги Бебров**
g.bebrov@tu-varna.bg
Гл. ас. д-р инж. **Мартин Иванов**
martinIvanov@tu-varna.bg
Ас. инж. **Николай Димитров** – докторант
nikolay.dimitrov@tu-varna.bg
Катедра "Комуникационна техника и технологии"
Технически университет – Варна