Crypto Analysis of Newly Compression Function

A.Arul Lawrence Selvakumar
Assistant Professor/CSE
The Oxford College of Engineering
Bangalore, INDIA

C.Suresh Gnanadhas
Professor/CSE
Veltech College of Engineering
Chennai, INDIA

Abstract
Cryptographic hash functions are a useful building block for several cryptographic applications. The most important are certainly the protection of information authentication and digital signatures. This overview paper will discuss the definitions, describe some attacks on hash functions, and will give an overview of the existing practical constructions

1. Introduction

Hash functions are functions that map an input of arbitrary length to a string of fixed length, the hashcode. If these mappings satisfy some additional cryptographic conditions, they can be used to protect the integrity of information. Other cryptographic applications where hash functions are useful are the optimization of digital signature schemes, the protection of passphrases and the commitment to a string without revealing it.
Hash functions appeared in cryptographic literature when it was realized that encryption of information is not sufficient to protect its authenticity. The simplest example is the encryption with a block cipher in Electronic Code Book (ECB) mode, where every block is encrypted independently. It is clear that an active attacker can easily modify the order of the ciphertext blocks and hence of the corresponding plaintext blocks. It will be shown that cryptographic hash functions allow for efficient constructions to protect authenticity with or without secrecy.
In a first part of this overview paper, the definitions of several types of hash functions will be given, and the basic attacks on hash functions will be discussed. Then it will be explained briefly how they can be used to protect the integrity of information. The protection of software integrity will be treated as an example. Subsequently a general model is described and an extensive overview is given of the proposed schemes. Here a distinction will be made between hash functions with an without a secret key. Finally the conclusions will be presented.

2. Definitions

In this section definitions will be given for hash functions that do not use a secret key (Manipulation Detection Code orMDC) and for hash functions that use a secret key (Message Authentication Code or MAC).According to their properties, the class of MDC's will be further divided into one-way hash functions (OWHF) and collision resistant hash functions (CRHF).

A brief discussion of the existing terminology can avoid the confusion that is present in the literature. The term *hash functions* originate historically from computer science, where it denotes a function that compresses a string of arbitrary input to a string of fixed length. Hash functions are used to allocate as uniformly as possible storage for the records of a file.

The name hash functions has also been widely adopted for cryptographic hash functions or cryptographically strong compression functions, but the result of the hash function has been given a wide variety of names in cryptographic literature:

hashcode, hash total, hash result, imprint, (cryptographic) checksum, compression, compressed encoding, seal, authenticator, authentication tag, fingerprint, test key, condensation, Message Integrity Code (MIC), message digest, etc.

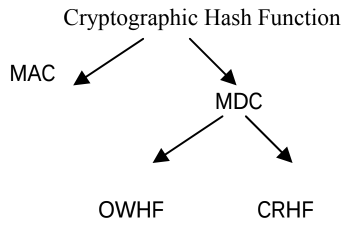

Figure 1: A taxonomy for cryptographic Hash Function

In the following the hash function will be denoted with h, and its argument, i.e., the information to be protected with X. The image of X under the hash function h will be denoted with (X) and the secret key with K.

The general requirements are that the computation of the hashcode is "easy" if all arguments are known. Moreover it is assumed that Kerckhoff's principle[2] is valid, which means that in the case of an MDC the description is public, and in the case of a MAC the only secret information lies in the key.

Manuscript received December 5, 2008
Manuscript revised December 20, 2008

2.1 One-way hash function (OWHF)

The first informal definition of a OWHF was apparently given by R. Merkle [88, 89] and M.O. Rabin [111].

***Definition1**: A **one-way hash function** is a function h satisfying the following conditions:*
1. The argument X can be of arbitrary length and the result h(X) has a fixed length of n bits (with $n \geq 64$).
***Definition 2:** The hash function must be one-way in the sense that given a Y in the image of h, it is **"hard"** to find a message X such that **h(X) = Y** and given X and h(X) it is **"hard"** to find a message $X^0 = X$ such that $h(X^0) = h(X)$.*

The first part of the second condition corresponds to the intuitive concept of one-wayness, namely that it is "hard" to find a preimage of a given value in the range. In the case of permutations or injective functions only this concept is relevant. The second part of this condition, namely that finding a second preimage should be hard, is a stronger condition that is relevant for most applications. The meaning of "hard" still has to be specified. In the case of "ideal security", introduced by X. Lai and J. Massey [79], producing a (second) preimage requires 2^noperations. However, it may be that an attack requires a number of operations that is smaller than 2^n, but is still computationally infeasible.

2.2 Collision resistant hash function (CRHF)

The first formal definition of a CRHF was apparently given by I. Damg°ard [31, 32]. An informal definition was given by R. Merkle in [89].

Definition 2: **A collision resistant hash function** is a function h satisfying the following conditions:

1. The argument X can be of arbitrary length and the result h(X) has a fixed length of n bits (with $n \geq 128$).

2. The hash function must be one-way in the sense that given a Y in the image of h, it is "hard" to find a message X such that h(X) = Y and given X and h(X) it is "hard" to find a message $X^0 = X$ such that $h(X^0) = h(X)$.

3. The hash function must be collision resistant: this means that it is "hard" to find two distinct messages that hash to the same result.

Under certain conditions one can argue that the first part of the one-way property follows from the collision resistant property. Again several options are available to specify the word "hard". In the case of "ideal security" [79], producing a (second) preimage requires 2^noperations and producing a collision requires $O(2^{n/2})$ operations. This can explain why both conditions have been stated separately.

One can however also consider the case where producing a (second) preimage and a collision requires at least $O(2^{n/2})$ operations, and finally the case where one or both attacks require less than $O(2^{n/2})$ operations, but the number of operations is still computationally infeasible (e.g., if a larger value of n is selected).

2.3 Message Authentication Code (MAC)

Message Authentication Codes have been used for a long time in the banking community and are thus older than the open research in cryptology that started in the mid seventies. However, MAC's with good cryptographic properties were only introduced after the start of open research in the field.

Definition 3: A **MAC** is a function satisfying the following conditions:
1. The argument X can be of arbitrary length and the result h (K, X) has a fixed length of n bits (with $n \geq 32 \ldots 64$).
2. Given h and X, it is "hard" to determine h(K, X) with a probability of success "significantly higher"than $1/2^n$. Even when a large number of pairs$\{X_i, h(K, X_i)\}$are known, where the $X_i$have been selected by the opponent, it is "hard" to determine the key K or to compute $h(K, X^0)$ for any $X^0 6= X_i$.This last attack is called an adaptive chosen text attack.

Note that this last property implies that the MAC should be both one-way and collision resistant for someone who does not know the secret key K. This definition leaves open the problem whether or not a MAC should be one-way or collision resistant for someone who knows K. An example where this property could be useful is the authentication of multi-destination messages [94].

3. Methods of attack

Three elementary attacks that are independent of the specific hash function will be described. For a more extensive overview of attacks on hash functions the reader is referred to [102, 106].
In case of a random attack the opponent select a random message and hopes that the modification will remain undetected. For an ideally secure hash function, his probability of success equals $1/2^n$. The feasibility of this attack depends on the action taken in case of detection of an erroneous result, on the expected value of a successful attack and on the number of attacks that can be carried out. For most applications this implies that n = 32 bits is not

sufficient. This is certainly true if an MDC is used and hence the opponent can attack several messages off-line and in parallel. In that case the hashcode should have a length of 64 bits or more.

The idea behind the **birthday attack** [126] is that for a group of 23 people the probability that at least two people have a common birthday exceeds 1/2. Intuitively one would expect that the group should be significantly larger. This can be exploited to attack a hash function in the following way: an adversary generates r_1 variations on a bogus message and $r_2$variations on a genuine message. A second possibility is that he collects a large number of messages and is able to divide them in two categories. This is the only way in the case of a MAC, where the opponent is unable to generate (message, MAC) pairs. The probability of finding a bogus message and a genuine message that hash to the same result is given by

$$1 - \exp(-(r1 \cdot r2)/2^n) ,$$

Which is about 63 % when $r = r1 = r2 = 2^{n/2}$. The involved comparison problem does not require r^2 operations: after sorting the data, which requires $O(r \log r)$ operations, comparison is easy. If the hash function can be called as a black box, the new algorithm of J.-J. Quisquater [109], requires $\sqrt{2\pi}.2^{n/2}$ operations and negligible storage. The conclusion is that if the hash function has to be collision resistant, n should be sufficiently large. In view of the fact that the speed of computers doubles every three years, 128 bits corresponding to 2^{64}operations is sufficient for the next 10 years, but it will be only marginally secure within 20 years.

In the case of a MAC, a third attack is relevant, namely an **exhaustive search** for the secret key K. It is a known plaintext attack, where an attacker knows M plaintext-MAC pairs for a given key and will try to determine the key by trying all possible keys. In order to determine the key uniquely, M has to be slightly larger than k/n. The expected number of trials is equal 2^{k-1}, with k the size of the key in bits.

4. Applications of cryptographic hash functions

This section aims to illustrate briefly how cryptographic hash functions can be used to protect the authenticity of information and to build digital signature schemes. In a second part the application to software protection will be discussed.

4.1 Information authentication and digital signatures

The protection of the authenticity of information includes two aspects:

- The protection of the originator of the information, or in ISO terminology [62] data origin authentication.
- The fact that the information has not been modified or in ISO terminology [62] the integrity of the information.

There are two basic methods for protecting the authenticity of information.

• The first approach is analogous to the approach of a asymmetric cipher, where the secrecy of large data quantities is based on the secrecy and authenticity of a short key. In this case the authentication of the information will also reply on the secrecy and authenticity of a key. To achieve this goal the information is compressed with a hash function and the resulting hashcode is appended to the information. The presence of this redundancy allows the receiver to make the distinction between authentic information and bogus information.

In order to guarantee the origin of the data, a secret key that can be associated to the origin has to intervene in the process. In the case of a MAC, the secret key is involved in the compression process, while in the case of an MDC, the information and/or the hashcode will subsequently be encrypted with a secret key.

• The second approach consist of basing the authenticity (both integrity and origin authentication) of the information on the authenticity of the hashcode.

Hash functions play also an important role in digital signature schemes. A digital signature is a cryptographic technique that produces the electronic equivalent of a manual signature. This means that a digital signature can prohibit the forging of a message by anybody else but the sender. Moreover the receiver is given guarantee of the message's authenticity, in the sense that he can subsequently prove to a third party that the message is authentic even if its originator revokes it. The concept of a digital signature was suggested in 1976 by W. Diffie and M. Hellman [45]: the sender transforms the information with a secret key, and the receiver can verify the signature by applying the corresponding public key.
The first practical scheme was the RSA cryptosystem [112]. Other efficient schemes include the scheme of ElGamal [46], the proposed NIST digital signature standard [51], and the schemes based on zero-knowledge techniques: Fiat-Shamir [47], Guillou-Quisquater [108], and Schnorr [117]. For a more detailed discussion the reader is referred to [95].

Most digital signature schemes can be optimized in the following way: the signature scheme is not applied to the message but to the hashcode of the message. Note that this corresponds to the second approach for authentication, i.e., replace the message by a short hashcode.

The advantages are that the signature has a fixed short length and that the computational requirements are minimized (most signature schemes are rather slow). Moreover the security level of the signature scheme can be increased. In other signature schemes, like the zero-knowledge based schemes and many theoretical constructions, a OWHF forms an integral part of the signature scheme.

Now the choice between a OWHF and a CRHF will be discussed. It is clear from the definition that a CRHF is stronger than a OWHF, which implies that using a CRHF is playing safe. On the other hand, it should be noted that designing a OWHF is easier, and that the storage for the hashcode can be halved (64 bits instead of 128 bits). A disadvantage of a OWHF is that the security level degrades proportional to the number of applications of *h:* an outsider who knows *s* hashcodes has increased his probability to find an $X^1$with a factor s. This limitation can be overcome through the use of a parameterized OWHF.

In order to understand when a CRHF is required, the following attack is considered. Assume that the has h function is not collision resistant: an attacker will thus be able to find a collision, i.e., two plain texts X and X^0such that h(X) = h(X^0). Subsequently he will protect the authenticity of X through h(X), and later he can substitute $X^0$for X. Therefore the use of a CRHF is recommended for any application where the attacker has complete control over the argument of the hash function, which implies he can select X, or he is an insider to the system.

However, one can certainly imagine applications where this attack is not relevant. If the attacker is an outsider, he can not choose X, and will be restricted to observe X and h(X). Subsequently he will try to come up with an X^0such that h(X) = h(X^0). Examples of applications where a OWHF would be sufficient are the following:

1. The parties involved completely trust each other, which is trivially the case if there is only one party. One could think of someone who protects the integrity of his computer files (which he only can generate) through the calculation and storage of an MDC.

2. The computation of the h(X) involves a random component, that can not be controlled by the insider [89]: X can be randomized before applying h through encryption of X with a good block cipher using a truly random key, that is added to the resulting ciphertext [88], or through the selection of a short random prefix to X [31];h itself can be randomized through randomly choosing h from a family of functions indexed by a certain parameter.

One can conclude that the choice between a CRHF and a OWHF is application dependent.

4.2 Software protection

To illustrate the use of a MAC, MDC, and a digital signature scheme, it will be explained how these three techniques can be applied to protect the integrity of software [92]. The two parties involved in the application are the software vendor (who is also supposed to be the author of the software) and the user.

The attacker will try to modify the software: this can be a computer virus [21], a competitor or even one of the parties involved. For this application there is clearly no need for secrecy. The three approaches will be discussed together with their advantages and disadvantages.

MAC: the software vendor will use his secret key to compute the MAC for the program and append the MAC to the program. The main problem here is the distribution of the secret key to the user through a channel that protects both its secrecy and its authenticity, which induces a significant overhead. This secret key has to be protected carefully by both software vendor and user: if a compromise at one place occurs, the protection is lost. Both software vendor and user can modify the program and the corresponding MAC, and thus in the case of a dispute, a third party can not make any distinction between them. The vulnerability of the secret key implies that it is mandatory that every user shares a different key with the software vendor. The advantage of this approach is that the secret storage is independent of the number of programs to be protected, but depends on the number of users (for the software vendor) and on the number of different software vendors (for the user).

MDC: the software vendor will compute the MDC for the program. The main problem here is the distribution of the MDC to the user through a channel that protects the authenticity of the MDC. This is easier than the distribution of a secret key, but for every update of the program or for every new program a new transmission of an MDC is necessary. If the authenticity of the MDC is compromised, the protection is lost: the software vendor, the user and any third party can modify the program and the corresponding MDC. If a dispute occurs, one has to show to a judge that the value of an MDC is authentic: it is generally not possible to prove to the judge who actually modified the authentic channel and the program. The main advantage is that this approach requires no secret storage. Every program needs an authentic storage both at the user's site and at the vendor's site.

Digital signature: the software vendor will append to the program a digital signature that is computed with his secret key. The main problem here is the distribution of the corresponding public key to the user through a channel that protects its authenticity. The secrecy and authenticity of the secret key have to be protected carefully by the software vendor: if it is compromised, anyone can modify programs and update the corresponding signature.

If the authenticity of the public key is compromised, the protection is also lost: anyone can replace it with the public key corresponding to his secret key. The difference with the previous approaches is the asymmetry: only the software vendor can generate a signature, but anyone can verify it. This implies that the vendor can be held liable to a third party if the program contains a virus. The only way he can escape is by claiming that his secret key was stolen. However, if he can not repeat this type of fraud, as he will loose quickly the trust of his customers. Every vendor has to store one secret key, while every user needs an authentic storage for the public key of every vendor.

The selection of a particular solution will depend on one hand on the number of users, vendors and programs, and on the other hand on the availability of authentic and/or secret storage and communication. The digital signature is clearly the only solution that can protect the users against a malicious software vendor.

A similar verification process can be executed when the program is loaded from disk to the Central Processing Unit. If the disk is not shared, non-repudiation is not required, but it is still attractive to use a digital signature scheme: the CPU has to know only the public key corresponding to the disk. An alternative is to provide for authentic storage of the MDC of a file that contains the MDC's of all programs.In [36] a scheme is described that combines a digital signature for checking new software with a MAC for verification at run-time.

5. An overview of MDC proposals

5.1 A general model

Before discussing a small fraction of the large number of proposals for MDC's, the general scheme for describing a hash function will be sketched. All known hash functions are based on a compression function with fixed size input; they process every message block in a similar way. This has been called an "iterated" hash function in [79]. The information is divided into t blocks X1 through X_t. If the total number of bits is no multiple of the block length, the information can be padded to the required length. The hash function can then be described as follows:

$$H_0 = IV$$
$$H_i=f(X_i,Hi-1) \qquad i=1,2,\dots t$$
$$h(X)=H_t$$

The result of the hash function is denoted with h(X) and IV is the abbreviation for Initial Value. The function f is called the round function. Two elements in this definition have an important influence on the security of a hash function: the choice of the padding rule and the choice of the IV.

It is recommended that the padding rule is unambiguous (i.e., there exist no two messages that can be padded to the same message), and that it appends at the end the length of the message. The IV should be considered as part of the description of the hash function. In some cases one can deviate from this rule, but this will make the hash function less secure and may lead to trivial collisions or second preimages.

The research on hash functions has been focussed on the question: what conditions should be imposed on f to guarantee that satisfies certain conditions? The two main results on this problem will be discussed. The first result is by X. Lai and J. Massey [79] and gives necessary and sufficient conditions for f in order to obtain an "ideally secure" hash function h.

Proposition 1 *Assume that the padding contains the length of the input string, and that the message X (without padding) contains at least 2 blocks. Then finding a second preimage for h with a fixed IV requires 2^n operations if and only if finding a second preimage for*
f with arbitrarily chosen H_i-1 requires 2^n operations.

A second result by I. Damg˚ard [33] and independently by R. Merkle [89] states that it is sufficient for f To be a collision resistant function.

Proposition 2 *Let f be a collision resistant function mapping l to n − l bits (with n − l > 1). If an unambiguous padding rule is used, the following construction will yield a collision resistant hash function:*

$$H_1 = f(0^{l+1} \| x_1)$$
$$H_i = f(H_{i-1} \| 1 \| x_1) \quad \text{for } i = 2, 3, \dots t.$$

5.2 Hash function based on a block cipher

Two arguments can be indicated for designers of hash functions to base their schemes on existing encryption algorithms. The first argument is the minimization of the design and implementation effort: hash functions and block ciphers that are both efficient and secure are hard to design, and many examples to support this view can be found in the literature. Moreover, existing software and hardware implementations can be reused, which will decrease the cost. The major advantage however is that the trust in existing encryption algorithms can be transferred to a hash function. It is impossible to express such an advantage in economical terms, but it certainly has an impact on the selection of a hash function. It is important to note that for the time being significantly more research has been spent on the design of secure encryption algorithms compared to the effort to design hash functions. It is also not obvious at all that the limited number of design principles for encryption algorithms are also valid for hash functions.

The main disadvantage of this approach is that dedicated hash functions are likely to be more efficient. One also has to take into account that in some countries export restrictions apply to encryption algorithms but not to hash functions. Finally note that block ciphers may exhibit some weaknesses that are only important if they are used in a hashing mode. A distinction will be made between hash functions for which the size of the hashcode equals the block length of the underlying block cipher and hash functions for which the size of the hashcode is twice this length. This is motivated by the fact that most proposed block ciphers have a block length of only 64 bits, and hence an MDC with a result twice the block length is necessary to obtain a CRHF. Other proposals are based on block ciphers with a large key and on a block cipher with a fixed key. The encryption operation E will be written as Y = E(K, X). Here X denotes the plaintext, Y the ciphertext and K the key. The corresponding decryption operation will be denoted with X = D(K, Y). The size of the plaintext and ciphertext or the block length will be denoted with r, while the key size will be denoted with k. In the case of the well known block cipher DES, r = 64 and k = 56 [48]. It will be assumed that the reader is familiar with the basic modes of operation of a block cipher as described in [49, 66, 91]. The rate of a hash function based on a block cipher is defined as the number of encryptions to process r plaintext bits.

5.2.1 Size of hashcode equals block length

From Definition 2 it follows that in this case the hash function can only be collision resistant if the block length r is at least 128 bits. Many schemes have been proposed in this class, but the first secure scheme was not proposed until 1985. Seven years later, the author has suggested a synthetic approach: we have studied all possible schemes which use exclusive ors and with an internal memory of only 1 block. As a result, it was shown that 12 secure schemes exist, but up to a linear transformation of the variables, they correspond essentially to 2 schemes: the 1985 scheme by S Matyas, C. Meyer and J. Oseas [85]:

$$f = E© (s(H_i{-}1), X_i)$$

(here s() is a mapping from the ciphertext space to the key space and E ©(K, X) = E(K, X) © X), and the variant that was proposed by the author in 1989 [102] and by Miyaguchi et al. [96] for N-hash and later for any block cipher [69] $f = E ©_(s(H_i{-}1^), X_i) © H_i{-}1$ The first scheme is currently under consideration for ISO standardization as a one-way hash function [67]. The 12 variants have slightly different properties related to weaknesses of the underlying block cipher [106]. One of those variants that is well known is the Davies-Meyer scheme [38, 125]: $f = E©_(X_i, H_i{-}1)$ It was also shown by the author that the security level of these hash functions is limited by min(k, r), even if the size of some internal variables is equal to max(k, r).

5.2.2 Size of hashcode equals twice block length

This type of functions has been proposed to construct collision resistant hash functions based on block ciphers with a block length of 64 bits like DES.
A series of proposals attempted to double the size of the hashcode by iterating a OWHF (e.g., [73, ?] and [83]); all succumbed to a ‘divide and conquer’ attack [24, 55, 95]. The scheme by Zheng, Matsumoto and Imai [128] was broken in [?]. A new scheme that was broken by the author and independently by I. Damg˚ard and L. Knudsen [35] is the Algorithmic Research hash function [70].

An interesting proposal was described by R. Merkle [89]. A security “proof” was given under the assumption that DES has sufficient random behaviour. However the rate of the most efficient proposal equals about 3.6. The proof for this proposal only showed a security level of 52.5 bits; however this has been improved to 56 bits in [106].

5.3 Hash functions based on modular arithmetic

These schemes are designed to use available hardware for modular arithmetic to produce digital signatures. Their security is partially based on the hardness of certain number theoretic problems and they are easily scalable. The disadvantage is that precautions have to be taken against attacks that exploit the mathematical structure like fixed points of modular exponentiation [10] (trivial examples are 0 and 1), multiplicative attacks and attacks with small numbers, for which no modular reduction occurs.

5.3.1 Schemes with a small modulus

A first type of schemes uses only a small modulus (about 32 bits). R. Jueneman proposed a first version in [71, 72], but several attacks resulted in several new versions [73, 74, 75]. The last scheme was broken by D. Coppersmith in [25].

5.3.2 Schemes with a large modulus

A second class of schemes uses a large modulus (the size of the modulus n is typically 512 bits or more). In this case the operands are elements of the ring corresponding to an RSA modulus, i.e., the product of two large primes, or of a Galois Field GF (p) or GF (2^n). For convenience these schemes will be denoted as schemes with large modulus, although this terminology applies strictly speaking only to RSA based schemes. The latter class can be divided into practical schemes and scheme that are provably secure. In the case of an RSA modulus, one has to solve the following practical problem: the person who has generated the modulus knows its factorization, and hence he has a potential advantage over the other users of the hash function. The solution is that the modulus can be generated by a trusted third party (in that case one can not use the modulus of the user), or one can design the hash function in such a way that the advantage is limited. The most efficient schemes are based on modular squaring.

Moreover some theoretical results suggest that inverting a modular squaring without knowledge of the factorization of the modulus is a difficult problem. Again one can study all possible schemes which use a single squaring and exclusive ors and with an internal memory of only 1 block. Several schemes of this type have been evaluated in previous papers [54, 99, 100]. It can be shown [106] that the optimal scheme is of the form:

$$f = (X_{i©}\ H_i{-}1)^2 \bmod N$$

Most existing proposals use however the well known Cipher Block Chaining (CBC) mode:

$$f = (X_i \quad H_i{-}1\ )^2 \bmod N\ .$$

In order to avoid the vulnerabilities (one can go backwards easily), additional redundancy is added to the message. The first proposal was to fix the 64 most significant bits to 0 [38]. It was however shown in [54, 76] that this is not secure. In a new proposal, that appeared in national and international standards (the informative annex D of CCITT-X.509 [18] and the French standard ETEBAC 5 [19]) the redundancy was dispersed. D. Coppersmith showed however that one can construct two messages such that their hashcode is a multiple of each other [26, 77]. If the hash function is combined with a multiplicative signature scheme like RSA [112], one can exploit this attack to forge signatures. As a consequence, new methods for adding redundancy were specified in [58, 77, 64, 68], but these methods are still under study.

Other schemes based on squaring that are insecure are the scheme by I. Damg˚ard [33] that was broken by B. den Boer [40] and the scheme by Y. Zheng, T. Matsumoto and H. Imai [128], that is vulnerable to the attack described in [54]. I. Damg˚ard [34] has identified several weaknesses in an MDC (and a MAC) based on arithmetic in GF (2^{593}) [1].

Stronger schemes have been proposed that require more operations. Examples are the use of two squaring operations [54]:

$$f = {}^3H_i{-}1 \quad (X_i)^{2}{}' \bmod N$$

and the replacement of the squaring by a higher exponent (3 or $2^{16}+ 1$) in the previous schemes. This allows simplifying the redundancy [16, 54].

One can conclude that it would be desirable to find a secure redundancy scheme for a hash function based on modular squaring and to replace the CBC mode by a more secure mode. If a slower scheme is acceptable, the exponent can be increased.

This section will be concluded with provably secure schemes. I. Damg˚ard [31, 32] has suggested constructions for which finding a a collision is provably equivalent to factoring an RSA modulus or finding a discrete logarithm modulo a large prime. The construction of J.K. Gibson [53] yields a collision resistant function based on the discrete logarithm modulo a composite. Both the factoring and the discrete logrithm problem are known to be difficult number theoretic problems. The disadvantages of these schemes are that they are very inefficient.

5.4 Schemes based on knapsacks

The knapsack problem is a well known NP-complete problem. In cryptography, a special case is considered, namely given a set of n b-bit integers $\{a_1, a_2 . . . a_n\}$, and an s-bit integer S (with $s \approx b + \log_2 n$), find a vector X with components $x_i$equal to 0 or 1 such that

The first application of this function was the 1978 Merkle-Hellman public key scheme [87]. Almost all schemes based on the knapsack problem have been broken [12, 44], which has given the knapsack a bad reputation. It is an open problem whether the knapsack problem is only hard in worst case, while the average instance is easy. If this would be true the knapsack problem would be useless for cryptography.

The attractivity of the problem lies in the fact that implementations in both hardware and software are much faster than schemes based on number theoretic problems.
In the case of additive knapsacks, several constructions have been suggested and broken (e.g., the scheme by I. Damg˚ard [33] was broken by P. Camion and J. Patarin

[15]). Other results can be found in [56, 61]. It is for the time being an open problem whether a random knapsack with n = 1024 and b = 512 is hard to solve.

The first multiplicative knapsack proposed by J. Bosset [11] was broken by P. Camion [14]. A new scheme by G. Z´emor is also based on the hardness of finding short factorizations in certain groups [127]. For the suggested parameters it can be shown that two messages with the same hashcode will differ in at least 215 bits. It remains an open problem whether it is easy to find factorizations of a 'reasonable size'.

5.5 Dedicated hash functions

In this section some dedicated hash functions will be discussed, i.e., algorithms that were especially designed for hashing operations.

MD2 [78] is a hash function that was published by R. Rivest of RSA Data Security Inc. in 1990. It makes use of a random 8-bit permutation and although it is software oriented, it is not too fast in software.The evaluation by the author [106] suggests that 16 rounds are sufficient, but only offer a marginal security. No other evaluations of MD2 have been published.

A faster algorithm by the same designer is MD4 [113, 114]. This algorithm uses standard 32-bit logic and arithmetic operations and is very efficient in software. The number of rounds, which corresponds to the number of times a single message bit is processed, is equal to 3. It was however shown by R. Merkle in an unpublished result and by B. den Boer and A. Bosselaers [40] that omitting the third respectively the first round yields a weak scheme, which suggests that the security offered by MD4 is only marginal (although the scheme is not broken for the time being).

In consequence of these attacks, R. Rivest realized that the security level of MD4 was not as large as he intended, and he proposed in 1991 a strengthened version of MD4, namely MD5 [115, 116]. An additional argument was that although MD4 was not a very conservative design, it was being implemented fast into products. An additional round was added, and several operations were modified to make the previous attacks more difficult. It was however shown by B. den Boer and A. Bosselaers [42] that with respect to certain criteria MD5 is weaker than MD4.

On January 31, 1992, NIST (National Institute for Standards and Technology, USA) published in the Federal Register a proposed Secure Hash Standard (SHS) [52] that contains the description of the Secure Hash Algorithm (SHA). This hash function is designed to work with the Digital Signature Algorithm (DSA) proposed in the Digital Signature Standard (DSS) [51]. The algorithm is clearly inspired by MD4, but it is a strengthened version. The two most remarkable changes are the increase of the size of the hashcode from 128 to 160 bits and the fact that the message words are not simply permuted but transformed with a cyclic code. Another improved version of MD4 called RIPEMD was developed in the framework of the EEC-RACE project RIPE. HAVAL was proposed by Y. Zheng, J. Pieprzyk and J. Seberry at Auscrypt'92 [129]; it is an extension of MD5. The most important properties are a variable number of rounds, a variable size of the hashcode and the use of highly nonlinear Boolean functions. This concludes the variants on MD4.

N-hash is a hash function designed by S. Miyaguchi, M. Iwata and K. Ohta [96, 98]. Serious weaknesses have been identified in this scheme by B. den Boer [41] and E. Biham and A. Shamir [6]. A new version of N-hash appeared in a Japanese contribution to ISO [69]. It was shown that this scheme contains additional weaknesses [9, 106].

FFT-Hash I and II are MDC's suggested by C.P. Schnorr [118, 119]. The first version was broken independently by J. Daemen, A. Bosselaers, R. Govaerts and J. Vandewalle [?] and by T. Baritaud, H. Gilbert and M. Girault [4]. The second version was broken three weeks after its publication by S. Vaudenay [122].

R. Merkle suggested in 1989 a software oriented one-way hash function that he called Snefru [90]. It is based on large random substitution tables (2 Kbyte per pass). E. Biham and A. Shamir showed several weaknesses of Snefru [7]. A weaker attack was found independently by J. Williams [124]. As a consequence of theses attacks, it is recommended to use 6 and preferably 8 passes, possibly combined with an increased size of the hashcode. However, these measures increase the size of the substitution tables and decrease the performance.

The scheme by I. Damg˚ard [33] based on cellular automata has been broken in [28]. J. Daemen, J. Vandewalle and R. Govaerts subsequently proposed Cellhash, a new hash function based on a cellular automaton [28] and an improved version Subhash [30]. Both schemes are hardware oriented.

6. An overview of MAC proposals

The general model for an iterated MAC is similar as the model for an MDC. The basic difference is that the round function f and in some cases the initial value IV depend on the secret key K. In contrast with the variety of MDC proposals, very few algorithms exist. This

can perhaps be explained by the fact that the existing standards are still widely accepted. the ANSI standard [3] specifies that the resulting MAC contains 32 bits. It is clear that a result of 32 bits can be sufficient if a birthday attack (cf. section 3) is not feasible and if additional protection is present against random attacks (cf. section 3), which is certainly the case in the wholesale banking environment. In other applications, this can not be guaranteed. Therefore, certain authors recommend also for a MAC a result of 128 bits [74, 75].
The most widespread method tocompute a MAC are the Cipher Block Chaining (CBC) and Cipher FeedBack (CFB) mode of DES [3, 50, 63, 65, 91]. The descriptions and standards differ because some of them select one of the two modes, suggest other padding schemes or leave open the number of output bits that is used for the MAC.

$$\mathrm{CBC} : f = E(K, H_i{-}1 \quad X_i)$$
$$\mathrm{CFB} : f = E(K, H_i{-}1^{)} \quad X_i$$

In the case of CFB it is important to encrypt the final result once more, to avoid a linear dependence of the MAC on the last plaintext block. In the case of DES, an attack based on exhaustive key search (cf. section 3) or differential attacks [8] can be thwarted by encrypting only the last block with triple DES; at the same time this can block the following chosen plaintext attack [41] (it will be described for the case` of CBC): let H and H^0be the CBC-MAC corresponding to key K and plaintext X and X^0respectively. The attacker appends a block The Message Authentication Algorithm (MAA) is a dedicated MAC.

It was published in 1983 by D. Davies and D. Clayden in response to a request of the UK Bankers Automated Clearing Services (BACS) [37, 39]. In 1987 it became a part of the ISO 8731 banking standard [63]. The algorithm is software oriented and has a 32-bit result, which makes it unsuitable for certain applications.

A new non-iterative MAC based on stream ciphers was proposed recently by X. Lai, R. Rueppel and J. Woollven [81]. Further study is necessary to assess its security.
The DSA algorithm (Decimal Shift and Add, not to be confused with the Digital Signature Algorithm) was designed in 1980 by Sievi of the German Zentralstelle f¨ur das Chiffrierwesen, and it is used as a message authenticator for banking applications in Germany [39]. Weaknesses of this algorithm have been identified in [59, 106]. The scheme by F. Cohen [22] and its improvement by Y. Huang and F. Cohen [60] proved susceptible to an adaptive chosen message attack [103]. Attacks were also developed [5, 106] on the weaker versions of this algorithm that are implemented in the ASP integrity toolkit [23]. Finally it should be noted that several MAC algorithms exist that have not been published, such as the S.W.I.F.T authenticator and Dataseal [?]. This section will be concluded with a discussion of provably secure and efficient MAC's based on universal hash functions [17, 20, 121, ?]. The advantage is that one can prove based on information theory that an attacker can do no better than guess the MAC: the security is hence independent of the computing power of the opponent.

7. Conclusions

Cryptographic hash functions are a useful tool in the protection of information integrity. Therefore the need exist for provably secure and efficient constructions. For the time being only a limited number of provably secure constructions exist, that are very inefficient. Some theoretical results are available to support practical constructions, but most of our knowledge on practical systems is originating from trial and error procedures. Therefore it is important that new proposals are evaluated thoroughly by several independent researchers and that they are not implemented too quickly. Moreover implementations should be modular such that upgrading of the algorithm is feasible. The choice between different algorithms will also depend on the available hardware.

References

[1] G.B. Agnew, R.C. Mullin and S.A. Vanstone, "Common application protocols and their security characteristics, "CALMOS CA34C168 Application Notes, U.S. Patent Number 4,745,568, August 1989

[2] S.G. Akl, "On the security of compressed encodings," Advances in Cryptology, Proceedings Crypto'83, D. Chaum, Ed., Plenum Press, New York, 1984, pp. 209–230.

[3] "American national standard for financial institution message authentication (wholesale)," X9.9-1986 (Revised), ANSI, New York.

[4] T. Baritaud, H. Gilbert and M. Girault, "FFT hashing is not collision-free," Advances in Cryptology, Proceedings Eurocrypt'92, LNCS 658, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 35–44.

[5] I. Bellefroid and K. Beyen, "Evaluatie van de cryptografische veiligheid van anti-virus paketten (Evaluation of the security of anti-virus software – in Dutch)," ESAT Laboratorium, Katholieke Universiteit Leuven, Thesis grad. eng., 1992.

[6] E. Biham and A. Shamir, "Differential cryptanalysis of Feal and N-hash," Advances in Cryptology, Proceedings Eurocrypt'91, LNCS 547, D.W. Davies, Ed., Springer-Verlag, 1991, pp. 1–16.

[7] E. Biham and A. Shamir, "Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer,"Advances in Cryptology, Proceedings Crypto'91, LNCS 576, J. Feigenbaum, Ed., SpringerVerlag, 1992, pp. 156–171.

[8] E. Biham and A. Shamir, "Differential cryptanalysis of the full 16-round DES," Technion Technical Report # 708, December 1991.

[9] E. Biham, "On the applicability of differential cryptanalysis to hash functions," E.I.S.S. Workshop on Cryptographic Hash

Functions, Oberwolfach (D), March 25-27, 1992.

[10] G.R. Blakley and I. Borosh, “Rivest-Shamir-Adleman public-key cryptosystems do not always conceal messages,” Comp. and Maths. with Applications, Vol. 5, 1979, pp. 169–178.

[11] J. Bosset, “Contre les risques d’alt´eration, un syst`eme de certification des informations,” 01 Informatique, No. 107, February 1977.

[12] E.F. Brickell and A.M. Odlyzko, “Cryptanalysis: a survey of recent results,” in “Contemporary Cryptology: The Science of Information Integrity,” G.J. Simmons, Ed., IEEE Press, 1991, pp. 501– 540.

[13] L. Brown, J. Pieprzyk and J. Seberry, “LOKI – a cryptographic primitive for authentication and secrecy applications,”Advances in Cryptology, Proceedings Auscrypt’90, LNCS 453, J. Seberry and J. Pieprzyk, Eds., Springer-Verlag, 1990, pp. 229–236.