Construction of secure and fast hash functions using nonbinary error-correcting codes

Knudsen, Lars Ramkilde; Preneel, Bart

Published in:
I E E E Transactions on Information Theory

Link to article, DOI:
10.1109/TIT.2002.801402

Publication date:
2002

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
Knudsen, L. R., & Preneel, B. (2002). Construction of secure and fast hash functions using nonbinary error-correcting codes. I E E E Transactions on Information Theory, 48(9), 2524-2539. DOI: 10.1109/TIT.2002.801402

Construction of Secure and Fast Hash Functions Using Nonbinary Error-Correcting Codes

Lars Knudsen and Bart Preneel, *Member, IEEE*

***Abstract*—This paper considers iterated hash functions. It proposes new constructions of fast and secure compression functions with nl-bit outputs for integers $n > 1$ based on error-correcting codes and secure compression functions with l-bit outputs. This leads to simple and practical hash function constructions based on block ciphers such as Data Encryption Standard (DES), where the key size is slightly smaller than the block size; IDEA, where the key size is twice the block size; Advanced Encryption Standard (AES), with a variable key size; and to MD4-like hash functions. Under reasonable assumptions about the underlying compression function and/or block cipher, it is proved that the new hash functions are collision resistant. More precisely, a lower bound is shown on the number of operations to find a collision as a function of the strength of the underlying compression function. Moreover, some new attacks are presented that essentially match the presented lower bounds. The constructions allow for a large degree of internal parallelism. The limits of this approach are studied in relation to bounds derived in coding theory.**

***Index Terms*—Birthday attacks, block ciphers, hash functions, nonbinary codes.**

I. Introduction

HASH *functions* map a string of arbitrary size to a short string of fixed length, typically $\ell = 128$ or 160 bits. They are very popular tools for cryptographic applications such as digital signatures, conventional message authentication, and password and pass-phrase protection schemes. The basic idea, dating back to the work by Diffie and Hellman [11], is that in a digital signature, one signs a short "digest" or "imprint" of the message, rather than the message itself. Similarly, when one has to protect the integrity of information between mutually trusting parties, one can protect the imprint rather than the information itself. For the protection of passwords or pass-phrases, one stores in the computer system the image under the hash function rather than the value itself.

While there are many preimages corresponding to any hash value, for cryptographic applications one requires that finding messages with identical hash values is difficult, and that it is hard to reconstruct the password or pass-phrase from the hash value. This distinguishes cryptographic hash functions from hash functions that are typically used in algorithmic applications like sorting. This can be translated to the following security properties:

> *preimage resistance*: for essentially all outputs, it is "computationally infeasible" to find any input hashing to that output;
>
> *second-preimage resistance*: it is "computationally infeasible" to find a second (distinct) input hashing to the same output as any given input;
>
> *collision resistance*: it is "computationally infeasible" to find two colliding inputs, i.e., x and $x' \neq x$ with $h(x) = h(x')$.

In this paper, a hash function that is preimage resistant and seceond-preimage resistant is called *one way*; a hash function that satisfies the three security properties is called *collision resistant*.

While the first two properties seem to be very close, one can show with some simple examples that they are distinct, and that none of them is strictly stronger than the other one (see, for example, Menezes *et al.* [30, Ch. 9]). The second and third property are also closely related, but collision resistance is strictly stronger than second-preimage resistance as explained later. A theoretical motivation for this has been provided by Simon [50]. A one-way hash function or compression function is called *ideal* if the best way known to find a preimage or a second-preimage is a brute-force search; such an attack requires on average $\Theta(2^{\ell-1})$ evaluations of the hash function. It is clear that such an attack can be parallelized efficiently. A collision-resistant hash function or compression function is called *ideal* if the best algorithm to find a collision is a brute-force collision search; such an attack requires on average $\Theta(2^{\ell/2})$ evaluations of the hash function, and a small amount of additional storage (Quisquater and Delescaille, [45]). This search is based on the so-called birthday paradox, as observed by Yuval in [52]. The basic idea is that one expects to find two colliding inputs in a set of size $\sqrt{2} \cdot 2^{\ell/2}$. Efficient parallel implementations of collision search algorithms are described by van Oorschot and Wiener in [51]. From their work, one can conclude that for a collision-resistant hash function, ℓ needs to be at least 160 bits or more; in 2001, this is sufficient to resist a well-funded opponent for 5 to 10 years. For preimage and second-preimage resistance, ℓ needs to be a least 64 bits (marginally secure); $\ell \geq 80$ is required for long-term security (again these are numbers valid in 2001).

Manuscript received October 1, 1998; revised April 6, 2001. This work was supported in part by the Fund for Scientific Research, Flanders (Belgium) and by the Concerted Research Action (GOA) Mefisto-2000/06 of the Flemish Government. This work was performed in part while visiting the University of Bergen, Norway. The material in this paper was presented in part at Asiacrypt'96, Kyungju, Korea, November 4–7, 1996 and at Crypto'97, Santa Barbara, CA, August 17–21, 1997.

L. Knudsen is with the Department of Mathematics, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (e-mail: lars@ramkilde.com).

B. Preneel is with the Department of Electrical Engineering-ESAT, Katholieke Universiteit Leuven, B-3001 Leuven-Heverlee, Belgium (e-mail: bart.preneel @esat.kuleuven.ac.be).

Communicated by D. Stinson, Associate Editor for Complexity and Cryptography.

Publisher Item Identifier 10.1109/TIT.2002.801402.

0018-9448/02$17.00 © 2002 IEEE

Note: The probability to find at least one collision after $\alpha\cdot 2^{\ell/2}$ hash function evaluations is equal to $1 - \exp\left(-\alpha^2/2\right)$. For $\alpha = \sqrt{2}$ as above, the success probability is equal to $1 - 1/e \approx 0.63$. In order to simplify the results, we will choose in the remainder of this paper $\alpha = 1$, corresponding to a success probability of $1 - 1/\sqrt{e} \approx 0.39$.

Extensive research has been performed on the design of hash functions that take a bit string of arbitrary length and produce an l-bit output from a compression function that takes a bit string of some fixed length ($>l$) and produces an l-bit output. A new method is proposed for constructing hash functions that take a bit string of arbitrary length and produce an output of length that is any multiple of l given a compression function with an l-bit output.

One particular interesting application of the results is for constructions based on block ciphers which typically have a small output size. More precisely, m-bit compression functions with $(t+1)m$-bit inputs $(t \geq 1)$ are considered using linear codes over GF (2^s) $(s \geq 2)$ resulting in fast and secure um-bit hash functions, where $u > 1$. Using block ciphers such as Data Encryption Standard (DES) [15], IDEA [26], and Advanced Encryption Standard (AES) [7], [18] as the underlying compression function, these constructions result in hash functions that are both faster and more secure than those known in the literature. Tables IV–IX in Section IX provide some concrete examples which allow to compare the security and efficiency of the schemes proposed in this paper to existing schemes.

In Section II, general construction methods for hash functions are summarized. Section III presents an overview of existing constructions for hash functions based on block ciphers and explains why these constructions are not satisfactory. In Section IV, a simple model for the new construction is proposed. The new construction is described in Section V, and is further developed in Section VI. A generic attack on all constructions is given in Section VII and Section VIII provides additional detail on the error-correcting codes used in the constructions. Some practical examples are given in Section IX, and the conclusions are presented in Section X.

II. General Constructions for Hash Functions

Almost all cryptographic hash functions are *iterated hash functions* based on a *compression function* $h(\cdot, \cdot)$ from two binary sequences of respective lengths l and l' to a binary sequence of length l. The message M is split into blocks M_i of l bits, $M = (M_1, M_2, \ldots, M_\tau)$. If the length of M is not a multiple of l, M is padded using an unambiguous padding rule (for example, always append a "1" bit followed by a number of "0" bits such that the length of the padded message becomes a multiple of l). The *hash result* Hash$(IV, M) = H = H_\tau$ is obtained by computing iteratively

$$H_i = h(H_{i-1}, M_i), \qquad i = 1, 2, \ldots, \tau \tag{1}$$

where $H_0 = IV$ is a specified *initial value*. Sometimes an *output transformation* $g(\cdot)$ is applied to H_τ to derive the hash result H from H_τ. The length in bits of H is denoted with ℓ. A collision/second-preimage/preimage *attack* on Hash(., .) is defined as an algorithm that tries to find a collision/second-preimage/preimage. In order to define these attacks in a formal way, one needs to formally specify a model of computation, the inputs of the algorithm, the type of algorithm, the input distributions, etc. We will skip this as formal definitions are not essential to understand the results in this paper (see, for example, [40]). Collision attacks, second-preimage attacks, and preimage attacks can be applied to both the compression function and the hash function. For the former, the attacker has full control over all $l + l'$ input bits. Lai calls this type of attacks *free-start* attacks [26], while Preneel uses the term *pseudo*collision/preimage attacks [39].

In the remainder of this paper no distinction is made between preimage and second-preimage attacks and the term "preimage attacks" is used to refer to both of them; it is always clear from the context if only one of these two is intended.

One can relate the security of Hash$(\cdot, \cdot)$ to that of $h(\cdot, \cdot)$ in several models; for collision resistance, this has been achieved independently by Damgård [8] and Merkle [32]; for preimage resistance, Lai and Massey have derived similar results in [26]. Naor and Yung did the same for a related concept, universal one-way hash functions [38]; see also Bellare and Rogaway [3] for further results on this type of hash functions. For one-way and collision-resistant hash functions, one needs to fix the IV of the hash function and append an additional block at the end of the input string containing its length, known as MD-strengthening (after Merkle [32] and Damgård [8]), leading to the following result.

Theorem 1 [8], [32]: Let Hash$(IV, \cdot)$ be an iterated hash function with MD-strengthening. Then preimage and collision attacks on Hash$(\cdot, \cdot)$ (where an attacker can choose IV freely) have roughly the same complexity as the corresponding attacks on $h(\cdot, \cdot)$.

Theorem 1 provides a lower bound on the security of Hash$(IV, \cdot)$. It indicates that a strong compression function is a sufficient but not a necessary condition for a strong hash function. Most practical hash functions do not treat the two inputs of the compression function in the same way; an example is the popular MDx-family, comprising MD4 [46], MD5 [47], SHA-1 [16], SHA-2 [17], and RIPEMD-160 [14]. Moreover, collisions for the compression function of MD5 have been presented by den Boer and Bosselaers [10] and by Dobbertin [13]; while it seems possible to extend the collisions of [13] to collisions for MD5 itself, this has yet not been achieved. MDC-2 and MDC-4 (see Section III-B) are examples of hash functions that are believed to offer a reasonable security level, but that have weak compression functions. The few hash functions that are designed according to Theorem 1 include DES based hash functions of Merkle [32] (cf. Section III-D), Snefru (another design by Merkle [33]), and the constructions proposed in this paper.

For preimage resistance, this result has been strengthened by Lai and Massey [26] as follows:

Theorem 2: Let Hash$(IV, \cdot)$ be an iterated hash function with MD-strengthening. Then Hash$(IV, \cdot)$ is ideally secure against preimage attacks if and only if $h(\cdot, \cdot)$ is ideally secure against preimage attacks.

This theorem shows that a compression function that is (ideally) resistant against preimage attacks is also a necessary condition for a hash function to be (ideally) resistant against preimage attacks.

For the remainder of this paper we shall assume that MD-strengthening is used. The main conclusion from Theorems 1 and 2 is that for an iterated hash function, the only way in which one knows to prove properties of the hash function is by starting from a strong compression function.

Note: It is also very natural to start from a collision-resistant compression function. This can be understood as follows: if one assumes that one has a collision-resistant hash function, which takes inputs of arbitrary size, one can always restrict the input to a fixed and small size. This results in a compression function, which is—by assumption—collision resistant. One can then use this compression function to define a new (but slightly slower) hash function which is based on a collision-resistant compression function.

The *hash rate* ρ of a hash function based on an m-bit block cipher with a tm-bit key is defined as the number of m-bit message blocks hashed per encryption; here one encryption is called one "operation." Similarly, the *hash rate* ρ of a hash function based on a $(t+1)m$-bit to m-bit compression function $h(\cdot, \cdot)$ is defined as the number of m-bit message blocks hashed per application of h; an application of h is also called one "operation." In summary: in order to hash τ m-bit message blocks, one needs τ/ρ applications of the block cipher, respectively, of the compression function. The *complexity* of an attack is the total number of operations required for an attacker to succeed with a high probability.

III. Hash Functions Based on Block Ciphers

Hash functions based on block ciphers have been popular in part for historical reasons, as designers tried to use the DES [15] also for hashing. This reduces the design and evaluation effort, and results in compact implementations, which is important for certain environments such as smart cards. It also allows to transfer the trust in DES (or in any other block cipher) to a hash function. This is quite important since many custom-designed hash functions have been broken. One illustration are Dobbertin's attacks [12], [13], [53] on MD4 [46] and MD5 [47]. One can expect that a similar argument will apply to AES [18]; however, further research on the use of AES in hash function constructions would be advisable.

However, this approach has some complications. The use of a block cipher in this application requires *different properties* from the block cipher. Indeed, it might be that the block cipher has certain properties that do not affect its security level for encryption, but create serious problems in hashing modes and vice versa. Examples are the (semi-)weak keys of DES [9], [36] and the quasi-weak keys and weak hash keys identified by Knudsen [21]. Also, in [23] it was shown that collisions for hash functions based on SAFER K [28] can be found faster than by using the birthday attack, but this does not seem to pose a threat to SAFER K when used for encryption. Another problem is that differential cryptanalysis can be adopted to this setting; for DES this has been explored by Rijmen and Preneel in [44]. A second element is that custom designed hash functions are likely to be more efficient. Moreover, the efficiency of these constructions is limited by the fact that every iteration typically requires a key change—this almost excludes block ciphers with a slow key setup such as RC5 [48]. One should also note that the use of a block cipher may create additional *export problems*.

The block length of a block cipher is denoted with m, while the key length is denoted with k. For convenience, it will be assumed that k is an integer multiple of m; it is possible to extend the constructions to the more general case. A block cipher defines, for each k-bit key, a random permutation on m-bit strings. In the following, $E_K(x)$ denotes the encryption of plaintext x using the key K.

In constructions using a block cipher it will be assumed that the block cipher has no weaknesses, i.e., that in attacks on the hash functions based on the block cipher, no shortcut attacks on the block cipher will help an attacker.

In the remainder of this section, constructions for hash functions based on block ciphers are reviewed. First, block ciphers are considered for which the block size m is equal to the key size k. Section III-A discusses single block length hash functions $(\ell = m)$, while Section III-B treats double block length hash functions $(\ell = 2m)$. Next, constructions are discussed for block ciphers for which the key length k is twice the block length. Finally, the proposals of Merkle are reviewed in Section III-D. They are important because they represent the first constructions with a security proof. This paper tries to extend his approach, but with different design constraints and assumptions.

Note that there are alternatives that are strictly speaking not hash functions based on block ciphers. Aiello and Venkatesan propose in [1] a construction to double the output of a random function. In order for it to be usable for hashing, one needs to define the key schedule of this larger "block cipher." The construction by Aiello, Haber, and Venkatesan [2] replaces the key schedule of DES by a function from the MDx family with the encryption; several instances are combined by choosing different (fixed) plaintexts.

A. Single Block Length Hash Functions ($\ell = m$)

For these hash functions the size of the hash result is equal to the block size of the block cipher. All these schemes have rate 1. The first secure construction for such a hash function was the scheme by Matyas, Meyer, and Oseas [29]

$$H_i = E_{H_{i-1}}(M_i) \oplus M_i.$$

This scheme has been included in the 1994 edition of ISO/IEC Std.10118-2 [20], with an additional mapping from the ciphertext space to the key space (as DES has $m = 64$ and $k = 56$). Its dual is known as the Davies–Meyer scheme after its inventors

$$f(M_i, H_{i-1}) \overset{\text{def}}{=} H_i = E_{M_i}(H_{i-1}) \oplus H_{i-1}. \tag{2}$$

As this function will be used repeatedly in this paper, a short notation, f, for it has been introduced.

A classification of all "simple" single block length hash functions has been presented by Preneel *et al.* in [42]. The main conclusion is that 12 secure variants exist, which are obtained by an affine transformation of variables applied to the Matyas, Meyer, and Oseas scheme and to this variant proposed independently by Preneel and Miyaguchi *et al.* [35]

$$H_i = E_{H_{i-1}}(M_i) \oplus M_i \oplus H_{i-1}.$$

The advantage of using the compression function f is that it is defined for block ciphers with different block and key sizes. It is conjectured that for the function f (and for the 11 variants) no shortcut attacks exist [42], which is rephrased as follows.

Assumption 1: Let $E_K(\cdot)$ be an m-bit block cipher with a tm-bit key K for an integer $t \geq 1$. Then finding collisions for f requires about $2^{m/2}$ encryptions (of an m-bit block), and finding a preimage for f requires about 2^m encryptions.

Note that there is only some empirical evidence for the security of f: after 10–15 years, no one has been able to find a better attack. For the remainder of this paper, this assumption is made if a block cipher is used as the underlying compression function.

Since most present-day block ciphers have a block length of $m = 64$ bits, collisions can be found in only 2^{32} operations. The AES [18] has only a block size of 128 bits. Therefore, hash functions with a larger hash result are needed. Note that Rijndael (the algorithm selected for AES) has also an instance with a block length of 256 bits.

B. Double Block Length Hash Functions ($\ell = 2m$)

The goal of *double block length* hash functions is to achieve a higher security level against collision attacks. Ideally, a collision attack on such a hash function should require 2^m operations, and a (second-)preimage attack 2^{2m} operations. An important class of proposals of rate 1 is of the following form:

$$\begin{aligned} H_i^1 &= E_{A_i^1}(B_i^1) \oplus C_i^1 \\ H_i^2 &= E_{A_i^2}(B_i^2) \oplus C_i^2 \end{aligned}$$

where A_i^1, B_i^1, and C_i^1 are binary linear combinations of H_{i-1}^1, H_{i-1}^2, M_i^1, and M_i^2 and where A_i^2, B_i^2, and C_i^2 are binary linear combinations of H_{i-1}^1, H_{i-1}^2, M_i^1, M_i^2, and H_i^1. The hash result is equal to the concatenation of H_τ^1 and H_τ^2. Several hash functions in this class have been published as individual proposals between 1989 and 1993. First, it was shown by Hohl *et al.* that the security level of the *compression function* of these hash functions is at most that of a single block length hash function [19]. Next, Knudsen *et al.* showed that for all hash functions in this class, a preimage attack requires at most 2^m operations, and a collision attack requires at most $2^{3m/4}$ operations (for most schemes this can be reduced to $2^{m/2}$) [22].

TABLE I
SECURITY LEVEL FOR MDC-2 AND MDC-4 BASED ON A BLOCK CIPHER WITH BLOCK AND KEY LENGTH EQUAL TO m BITS

	hash function Hash	
	collision	preimage
MDC-2	2^m	$2^{3m/2}$
MDC-4	2^m	$2^{7m/4}$

	compression function h	
	collision	preimage
MDC-2	$2^{m/2}$	2^m
MDC-4	$2^{3m/4}$	$2^{3m/2}$

Several schemes of rate less than 1 have been proposed. From the few that have survived, the most important ones are MDC-2 and MDC-4 with hash rate $1/2$ and $1/4$, respectively [4]; they are also known as the Meyer–Schilling hash functions after the authors of the first paper describing these schemes [34]. MDC-2 has been included in the 1994 edition of ISO/IEC Std.10118-2 [20]; it can be described as follows:

$$\begin{aligned} T_i^1 &= f(u(H_{i-1}^1),\, M_i) = LT_i^1 \parallel RT_i^1 \\ T_i^2 &= f(v(H_{i-1}^2),\, M_i) = LT_i^2 \parallel RT_i^2 \\ H_i^1 &= LT_i^1 \parallel RT_i^2 \\ H_i^2 &= LT_i^2 \parallel RT_i^1. \end{aligned}$$

Here, f denotes the Davies–Meyer hash function (cf. Section III-A), and u and v are mappings from the ciphertext space to the key space such that $u(x) \neq v(y)$. The variables H_0^1 and H_0^2 are initialized with the values IV^1 and IV^2, respectively, and the hash result is equal to the concatenation of H_τ^1 and H_τ^2. The best known preimage and collision attacks on MDC-2 require $2^{3m/2}$ and 2^m operations, respectively (Lai, [26]). However, it is easy to see that the compression function of MDC-2 is rather weak: preimage and collision attacks on the compression function require at most 2^m and $2^{m/2}$ operations (one fixes M_i and varies H_{i-1}^1 and/or H_{i-1}^2). A collision attack on MDC-2 based on DES $(m = 64, k = 56)$ requires at most 2^{54} encryptions (u and v drop the parity bits in every byte and fix the second and third key bits to 01 and 10, respectively).

One iteration of MDC-4 [4] is defined as a concatenation of two MDC-2 steps, where the plaintexts in the second step are equal to H_{i-1}^2 and H_{i-1}^1. The rate of MDC-4 is equal to $1/4$. The best known preimage and collision attacks on MDC-4 require $2^{7m/4}$ and 2^m operations, respectively. This shows that MDC-4 is probably more secure than MDC-2 against preimage attacks. However, a collision for the compression function of MDC-2 with a specified value for H_{i-1}^1 and H_{i-1}^2 also yields a collision for the compression function of MDC-4. Moreover, the authors have demonstrated in [25] that collisions can be found for the compression function of MDC-4 with $2^{3m/4}$ encryptions and the storage of $2^{3m/4}$ m-bit quantities.

The security level of the hash functions MDC-2 and MDC-4 (with fixed IV's) and of their compression functions is listed in Table I. These attacks are described in [25] and [26]. Note that the compression function is not very strong and that the protection of the hash function against collision attacks is not very high if DES is used.

Preneel *et al.* describe in [41] a class of constructions that extend MDC-2 to n parallel iterations, but that keep the key fixed. Compression is achieved by chopping some bits of the output. In between the iterations bits are permuted between the different blocks. This approach results in a tradeoff between performance and security, and requires an internal memory that is larger than suggested by the security level. These two properties are shared with the schemes proposed in this paper. However, as the compression function is not collision resistant, it seems very hard to prove anything about the security of these hash functions.

C. Block Ciphers With $k \geq 2m$

Merkle observed in [31] that if the key length of a block cipher is larger than the block length, it can be used as the compression function of a single block length hash function by just fixing the plaintext, and considering the mapping from key to ciphertext

$$H_i = E_{H_{i-1} \,\|\, M_i}(C)$$

with C as a constant.

In [26], Lai and Massey propose two constructions for hash functions based on their block cipher IDEA (with $m = 64$ and $k = 128$): Abreast-DM and Tandem-DM have hash rate $1/2$ and a claimed security level against preimage and collision attacks equal to 2^{2m}, respectively, 2^m operations.

Note that both MD4 [46] and MD5 [47] can be viewed as a Davies–Meyer construction with an underlying m-bit block cipher with a $4m$-bit "key." Indeed, the compression function has a feedforward from the "plaintext" (the chaining variable H_{i-1}) toward the "ciphertext" (H_i). For MD4 and MD5, the size of the message block in bits $l = k = 512$, and the size of the chaining variable and the hash result are $\ell = l' = m = 128$. From this perspective, both constructions have rate 4. However, Dobbertin's attacks [12], [53], [13] on MD4 and MD5 show that the compression functions are not collision resistant. His attack [12], [53] on "extended MD4" [46], which has a compression function consisting of two parallel and dependent runs of MD4, shows that it is not obvious to increase the security of these constructions.

D. Schemes by Merkle

Merkle proposed a new class of hash functions based on block ciphers with a collision-resistant compression function [32]. The reason why these schemes are treated separately is that, unlike the other proposals in Section III-B, they have a security proof, based on the assumption that the Davies–Meyer single block length hash functions is secure.

The simplest scheme (with rate $1/18.3$ for DES) can be described as follows:

$$H_i = \text{chop}_{16}\left[f\left(0\middle\|H^1_{i-1},\, H^2_{i-1}\middle\|\, M_i\right)\,\middle\|f\left(1\middle\|H^1_{i-1},\, H^2_{i-1}\middle\|\, M_i\right)\right].$$

Here H_{i-1} is a string consisting of 112 bits, the leftmost 55 bits of which are denoted H^1_{i-1}, and the remaining 57 are denoted H^2_{i-1}; M_i consists of seven bits only. The function chop_r drops the r rightmost bits of its argument.

This hash function is similar to MDC-2, but has a collision resistant compression function at the cost of a low speed; Merkle shows that if DES has no weaknesses, finding a collision for this compression function requires at least 2^{55} operations. Note that if DES is being used, additional measures have to be taken to preclude attacks based on weak keys (resulting in a lower speed).

The faster versions are more complex and use six invocations of the block cipher in two layers. The analysis becomes more complex as well. Merkle shows that for the fastest scheme (with rate 0.276 for DES; 0.265 if weak keys are taken into account), finding collisions requires $2^{52.5}$ operations. This lower bound has been improved in [39] to 2^{54}.

This approach performs a remarkable improvement over previous proposals, but has the following disadvantages.

- The security level seems to be limited to $2^{\min\{k,\, m\}}$, which is not sufficient if DES is used, and only marginally sufficient for a 128-bit block cipher.
- The block sizes for the data input are not convenient, i.e., not a multiple of 32 or 64 bits.
- The invocation of the block cipher is in part serial, which is a disadvantage for high-speed hardware implementations.

IV. Model for the New Construction

This paper provides new constructions that extend ideal compression functions of m bits to hash functions for which finding a collision requires strictly more than 2^m operations, and that allow for parallel processing of the individual compression function calls. It has already been argued in Section II that one should try to design a collision-resistant compression function. This seems the only approach possible if one wants to prove something about the security of the hash function. In the following, let $h(\cdot,\, \cdot)$ denote the underlying compression function that takes two inputs, the first of size m bits, the second of size tm bits, and that produces an m-bit output.

The most straightforward approach is to consider n parallel functions and construct a compression function $\mathcal{H}$ of rate $\rho = r/n$ as follows:

$$\begin{aligned} &H^1_i = h(X^1_i,\, Y^1_i) \oplus Z^1_i \\ &\cdots \\ &H^n_i = h(X^n_i,\, Y^n_i) \oplus Z^n_i \end{aligned}$$

where X^j_i, Y^j_i, and Z^j_i are derived from binary linear combinations of H^j_{i-1}, and $M^{j'}_i$, $1 \leq j \leq n$, and $1 \leq j' \leq r$. Note that these functions can be evaluated in parallel, as none of the inputs depends on the outputs of the other functions.

Schemes of this form with $t = 1$ and $n = 2$ have been proposed in the literature, see, e.g., Knudsen *et al.* [22]. As illustrated by the attacks in [22], it is strongly suggested that it is hard to invert individual parts of the compression function: partial inversions may be extended by a meet-in-the-middle attack to an inversion of the compression function. Sorting out this situation for $n = 2$ has taken quite some cryptanalytic effort, and an elapsed time of about seven years. While it is possible to write down some schemes for $n = 3$ or $n = 4$ for which it is not immediately clear (at least to the authors) how to break them, this approach seems to be destined to fail. Moreover, it is not clear how one would be able to prove anything about the security of such a scheme. This leads us to specify the requirement that each individual function is an ideal compression function by itself. In order to avoid trivial attacks, which consist of making the inputs of the different functions equal, $\lceil \log_2 n \rceil$ input bits are fixed to different values. This generalizes the approach of MDC-2 and the schemes by Merkle. This is reflected in notation by giving the individual functions different subscripts.

Definition 1 (Multiple Construction): Let $h(\cdot, \cdot)$ be an ideal collision-resistant compression function that takes two inputs, the first of tm bits and the second of size m bits, and produces an m-bit output. The compression function of a multiple construction with rate $\rho = r/n$ has the following form:

$$\begin{aligned} H_i^1 &= h_1(X_i^1, Y_i^1) \\ &\cdots \\ H_i^n &= h_n(X_i^n, Y_i^n) \end{aligned}$$

where $h_1, \ldots, h_n$ are different instantiations of h (cf. supra) X_i^j and Y_i^j are derived from linear combinations of H_{i-1}^j, and $M_i^{j'}$, $1 \le j \le n$, and $1 \le j' \le r$.

Note that MDC-2 (without the swapping of the right halves) can be described as such a scheme. For MDC-2, $n = 2$, $t = 1$, $X_i^1 = H_{i-1}^1$, $X_{i-1}^2 = H_{i-1}^2$, and $Y_i^1 = Y_i^2 = M_i$. However, as explained earlier, MDC-2 does not have a strong compression function. One could easily generalize MDC-2 to the case with $n = 3$ or $n = 4$. This does not increase the strength of the compression function, but again it is not obvious how to extend attacks on the compression function to attacks on the hash function.

The main design goal is to find linear mappings that result in a compression function for which finding a collisions and a preimage requires at least 2^m operations, and preferably even more. Under this constraint, one can prove the following result for $t = 1$; it provides a lower bound on n, the number of parallel chains.

Proposition 1: Let $\mathcal{H}$ be a multiple construction with $t = 1$ (see Definition 1). For $n \le 2$ finding collisions requires $\le 2^{m/2}$ operations, and for $3 \le n \le 4$ finding collisions requires $\le 2^m$ operations.

Proof: The case $n = 1$ is trivial. If $n = 2$, there are two chaining variables H_{i-1}^1 and H_{i-1}^2 and at least one message variable M_i^1 (note that the rate is strictly positive). One can choose these variables in such a way that one of the outputs, say H_i^1, is constant, by imposing two linear constraints on these variables. One can then use the remaining degrees of freedom to perform a $2^{m/2}$ brute-force collision attack on H_i^2. If $n = 3$, one has three chaining variables H_{i-1}^1, H_{i-1}^2, and H_{i-1}^3 and at least one message variable M_i^1. One can choose these variables in such a way that one of the outputs, say H_i^1, is constant. One can then use the remaining degrees of freedom to perform a 2^m brute-force collision attack on H_i^2 and H_i^3. For $n = 4$, one has four chaining variables H_{i-1}^1, H_{i-1}^2, H_{i-1}^3, and H_{i-1}^4 and at least one message variable M_i^1. Once can choose these variables in such a way that two of the outputs, say H_i^1 and H_i^2, are constant; this requires that one imposes four linear constraints. The remaining degree of freedom can then be used to perform a 2^m brute-force collision attack on H_i^3 and H_i^4. □

Notes:

1) For $n = 5$, one has five chaining variables and one message variable. This offers sufficient degrees of freedom to fix three chaining variables. However, there are then no degrees of freedom left to find a collision for the remaining two chaining variables; therefore, the approaches for $n \le 4$ above will not work in this case. This does not imply that there exists a construction with $n = 5$ which offers a security level $> 2^m$ operations.

2) Proposition 1 assumes implicitly that at least one complete message block is processed in every iteration. This condition could be relaxed, resulting in schemes with $n < 5$.

3) It might be that the linear mappings are defined in such a way that one needs fewer than $2w$ variables to fix w chains. In that case, larger values of n would be required. However, it will be assumed in the following that the matrix of the linear mapping has full rank.

Proposition 1 can be generalized to the case $t \ge 1$.

Proposition 2: Let $\mathcal{H}$ be a multiple construction with $t \ge 1$ (see Definition 1) for which finding collisions requires $>2^{sm/2}$ operations. Then n has to satisfy the following inequality:

$$\left\lceil \frac{2tn - 2r}{2t + 1} \right\rceil > s.$$

Proof: One can fix the inputs to a chains out of the n, and perform a brute-force collision search on the remaining $n - a$ chains. This requires $2^{sm/2}$ operations, with $s = n - a$. As pointed out earlier, one also needs to make sure that sufficient degrees of freedom are available for the brute-force attacks. The total number of variables is equal to $n + r$, and fixing one chain requires that one imposes $t + 1$ constraints on the variables (remember that the matrix of the linear matrix has full rank). On the other hand, a brute-force collision attack on $n - a$ chains requires $(n - a)/2$ "free" variables. This implies that the attack is feasible if a satisfies the following condition:

$$n + r - (t + 1)a \ge \frac{n - a}{2}.$$

This can be solved for a as

$$a \le a^* = \left\lfloor \frac{n + 2r}{2t + 1} \right\rfloor.$$

The effort for the brute-force attack is minimized if a is maximized. The resulting value of s is equal to

$$s = n - a^* = \left\lceil \frac{2tn - 2r}{2t + 1} \right\rceil$$

which proves the proposition. □

Notes:

1) Proposition 2 only provides a *lower* bound on the value of n, as it considers a very simple attack. A more sophisticated attack will be presented in Section VII.

2) For $s > 2$ (as in Proposition 1), one needs for $t = 2$ that $n \ge 4$ and for $t \ge 3$, $n \ge 3$.

3) For a fixed value of r, the security level s grows at most linearly with n, while the rate ρ decreases with $1/n$. If n and r are increased, with $r \approx tn - c$, for a constant c, the security level (for this attack) remains constant, and the rate ρ approaches t.

Propositions 1 and 2 show that for a secure multiple hash function with $t = 1$ at least five parallel chains are required, while $t = 2$ requires at least four parallel chains. Designing such a scheme by trial and error seems to be very hard. This provides additional motivation to search for a more structured approach. Such an approach is developed in the next section.

V. Simple Construction With Quaternary Linear Codes

This section proposes a class of hash functions following the model of Definition 1 for $t = 1$: the compression function consists of n parallel instances of an ideal compression function h: $\{0, 1\}^m \times \{0, 1\}^m \rightarrow \{0, 1\}^m$. The goal is to find n linear combinations of the $n + r$ variables $H_{i-1}^1, \ldots, H_{i-1}^n$, $M_i^1, \ldots, M_i^r$ in such a way that finding a collision for the compression function requires more than 2^m operations. Proposition 1 implies that $n \geq 5$; later conditions will be derived for r, as well as for the rate r/n of the hash function. The simple construction is developed for compression functions with two inputs, each of bit length m. As an example, $m = 64$ if DES is used as the underlying block cipher in a Davies–Meyer construction.

In order to prove the security of the construction, two assumptions are required which are presented in the next subsection.

A. Security Assumptions

The first assumption is clear and obvious from the previous discussion: it is assumed that the underlying m-bit compression function is ideally secure.

Before the second assumption can be stated, some additional definitions are required. Consider a collision (or second-preimage) attack where the two sets of inputs are

$$H_{i-1}^1, \ldots, H_{i-1}^n, M_i^1, \ldots, M_i^r$$

and

$$H'^1_{i-1}, \ldots, H'^n_{i-1}, M'^1_i, \ldots, M'^r_i$$

respectively. Define the *active* inputs as the set of pairs

$$(H_{i-1}^j, H'^j_{i-1}) \quad \text{and} \quad (M_{i-1}^k, M'^k_{i-1})$$

for which $H_{i-1}^j \neq H'^j_{i-1}$ and $M_{i-1}^k \neq M'^k_{i-1}$.

A subfunction $h_i(X_i, Y_i)$ is called *active* (with respect to (w.r.t.) the collision or the second-preimage attack), if either X_i and/or Y_i is computed from active inputs.

A set of subfunctions $h_{i_1}(X_{i_1}, Y_{i_1}), \ldots, h_{i_s}(X_{i_s}, Y_{i_s})$ can be attacked *independently*, if for all $j \in \{1, \ldots, s\}$ it holds that: for all values of the input blocks affecting (X_{i_j}, Y_{i_j}) to h_{i_j} the arguments (X_{i_k}, Y_{i_k}) to h_{i_k} are fixed for

$$k \in \{1, \ldots, j-1, j+1, \ldots s\}.$$

Assumption 2: Assume that a collision for the compression function of a multiple scheme has been found, that is, simultaneously for $h_1, h_2, \ldots h_n$. Let N be the number of active subfunctions and let $N - v_1$ be the maximum number of the N subfunctions that can be attacked independently. Take v as the minimum value of all such v_1's. Then it is assumed that obtaining this collision must have required at least $2^{vm/2}$ encryptions.

In an attempt to find collisions for a multiple scheme it will always be possible to fix the input blocks to some subfunctions and thereby fix the outputs. Let N denote the number of active subfunctions. Assumption 2 states that, if a maximum of $N - v$ of these N functions can be attacked independently, then there exists no better attack than a brute-force attack on the remaining v subfunctions.

Note that in the overall complexity of the collision attack the complexity of the attack on the $N - v$ functions is not considered, which makes the assumption strong and plausible. For example, consider the compression function of MDC-2, $N \leq 2$. As mentioned earlier, it is possible to find collisions for the compression function by fixing the inputs to one of the two subfunctions and do a brute-force attack on the other, that is, with $N = 1$. In this case, $v = 0$, since $N - v = 1$. This implies, from Assumption 2, that collisions for the compression function of MDC-2 must have required at least one operation while the best known attack requires $2^{m/2}$ operations.

B. The New Construction

The following theorem shows how to construct strong hash functions based on ideal m-bit compression functions using nonbinary linear error correcting codes.

Theorem 3: If there exists an $[n, k, d]$ code over GF (2^2) of length n, dimension k, and minimum distance d, with $2k > n$, for $m \gg \log_2 n$, then there exists a parallel hash function based on an ideal compression function h: $\{0, 1\}^m \times \{0, 1\}^m \rightarrow \{0, 1\}^m$, for which finding a collision for the compression function requires at least $2^{(d-1)m/2}$ operations provided that Assumption 2 holds. The hash function has an internal memory of $n \cdot m$ bits, and a rate $\rho = \frac{2k}{n} - 1$.

Proof: The compression function consists of n different functions h_i with $1 \leq i \leq n$, see Definition 1. The input to the compression function consists of $2k$ m-bit blocks: the n variables H_{i-1}^1 through H_{i-1}^n (the output of the n functions of the previous iteration) and the r message blocks M_i^1 through M_i^r, with $r = 2k - n > 0$. In the following, every individual bit of these m-bit blocks is treated in the same way. The bits of two consecutive input blocks are concatenated yielding k elements of GF (2^2). These elements are encoded using the $[n, k, d]$ code, resulting in n elements of GF (2^2). Each of these elements represents the 2-bit inputs to one of the n functions. As an example, if the compression function is built from a block cipher, one bit represents the plaintext block input and the other bit represents the key input to the block cipher. The individual input bits are obtained by representing the elements of GF (2^2) as a vector space over GF (2). This construction guarantees that the conditions for Assumption 2 are satisfied for the value $v = d - 1$. To see this, first note that since the dimension of the code is k, one can rearrange the subfunctions such that the first k subfunctions can be attacked *independently*. It is claimed that in an attack at least $d - 1$ of the last $n - k$ subfunctions are active. To see this, assume that in a collision attack $k^* \le k$ of the first k subfunctions are active and that only $d^* < d-1$ of the last $n-k$ subfunctions are active. It is then possible to find two inputs to the compression function, that differ in the inputs to only one of the first k subfunctions (by fixing the inputs to some $k^* - 1$ subfunctions) and at most d^* of the last $n-k$ subfunctions. But this is a contradiction since it follows from the minimum distance of the code that the inputs to at least d subfunctions in total are different. □

Note:

Section V-C provides two examples of such constructions together with an interpretation of the proof of security.

Apart from the simple security proof and the relatively high rates, the schemes have the advantage that the n operations can be carried out in parallel. The disadvantages of the schemes are the increased amount of internal memory and the cost of the implementation of the linear code (mainly, some EXCLUSIVE-ORS). Note that the time for a block cipher encryption corresponds typically to a few hundred EXCLUSIVE-ORS.

Theorem 3 reduces the question of the existence of efficient hash functions based on a small compression function to the existence of certain quaternary error-correcting codes. The main conditions on the code are that the minimum distance should be as large as possible (at least 3) and also that the dimension k should be as large as possible (and at least $n/2+1$). The Singleton bound states that $k \le n-d+1$. Define the *deficit* $\delta \overset{\text{def}}{=} n-k-d+1$. For an maximum-distance separable (MDS) code, $\delta = 0$. The rate of the hash function can then rewritten as follows:

$$\rho = \frac{2k}{n} - 1 = \frac{2(n-d+1-\delta)-n}{n} = 1 - 2 \cdot \frac{d+\delta-1}{n}.$$

For a quaternary code ($q=4$), $\delta = 0$ only for the $[5, 3, 3]$ Hamming code (cf. Section VIII) resulting in a scheme with rate $1/5$. However, there exist codes which are close to MDS, namely, with $\delta = 1$: for $7 \le n \le 16$ there exist codes with $d = 3$, and for $9 \le n \le 16$ there exist codes with $d = 4$. Larger values of n are not of practical interest. If $\delta = 1$, the rate $\rho = 1 - 6/n$ for $d = 3$, and $1 - 8/n$ for $d = 4$. This means that even for moderate values of n, rates can be achieved that are much higher than existing schemes, and this for a higher security level. The existence of such codes is revisited in Section VIII. In what follows, a complete scheme is described starting from the $[5, 3, 3]$ code, and some details are provided for the $[8, 5, 3]$ code.

C. Two Examples Using the (Shortened) Hamming Codes

The following 3×5 matrix is a generator matrix for the $[5, 3, 3]$ Hamming code over GF(2^2)

$$G = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \boldsymbol{\alpha} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \boldsymbol{\beta} \end{bmatrix} \qquad (3)$$

here $\mathbf{0} = [00]$, $\mathbf{1} = [01]$, $\boldsymbol{\alpha} = [10]$, and $\boldsymbol{\beta} = [11]$. The order of the chaining variables is given by the following vector:

$$V = [H_{i-1}^1, H_{i-1}^2, H_{i-1}^3, H_{i-1}^4, H_{i-1}^5, M_i^1].$$

Now G is transformed into a 6×10 generator matrix G' over GF(2), by replacing each element by a 2×2 matrix. This matrix is the matrix of the linear transformation corresponding to the multiplication by that element. Hence,

$$\mathbf{0} \to \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad \mathbf{1} \to \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\alpha} \to \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \boldsymbol{\beta} \to \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

The corresponding generator matrix G' is then equal to

$$G' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}. \qquad (4)$$

Now one computes the product $V \cdot G'$, resulting in a vector with 10 components. The first two components correspond to the key and the plaintext input to the first compression function h_1. Components three and four correspond to the two inputs of the second function h_2, and so on. This results in the following compression function $\mathcal{H}$:

$$\begin{aligned} H_i^1 &= h_1\left(H_{i-1}^1, H_{i-1}^2\right) \\ H_i^2 &= h_2\left(H_{i-1}^3, H_{i-1}^4\right) \\ H_i^3 &= h_3\left(H_{i-1}^5, M_i^1\right) \\ H_i^4 &= h_4\left(H_{i-1}^1 \oplus H_{i-1}^3 \oplus H_{i-1}^5, H_{i-1}^2 \oplus H_{i-1}^4 \oplus M_i^1\right) \\ H_i^5 &= h_5\left(H_{i-1}^1 \oplus H_{i-1}^3 \oplus H_{i-1}^4 \oplus M_i^1,\right. \\ &\qquad \left. H_{i-1}^2 \oplus H_{i-1}^3 \oplus H_{i-1}^5 \oplus M_i^1\right) \end{aligned}$$

where $h_1, \ldots, h_5$ are different instances of the underlying compression function h.

Under Assumption 2 and according to Theorem 3, a collision of $\mathcal{H}$ requires at least 2^m operations. Consider Assumption 2 and three different cases. First, assume a collision has been found where the inputs differ only in H_{i-1}^1. Then clearly the active subfunctions are h_1, h_4, and h_5, moreover, $N = 3$ and $v = 2$. Second, assume a collision has been found where the inputs differ only in H_{i-1}^1 and H_{i-1}^3. Then clearly the active subfunctions are h_1, h_2, h_4, and h_5, moreover, $N = 4$ and a maximum of two subfunctions, e.g., h_1 and h_2, can be attacked independently of each other yielding $v = 2$. Third, assume a collision has been found where the inputs differ in both H_{i-1}^1 and H_{i-1}^3, that is, $H_{i-1}^1 \ne {H'}_{i-1}^1$ and $H_{i-1}^3 \ne {H'}_{i-1}^3$, but where the attacker's strategy was to choose the values such that $H_{i-1}^1 = H_{i-1}^3$ and ${H'}_{i-1}^1 = {H'}_{i-1}^3$. Then the active subfunctions are again h_1, h_2, h_4, and h_5, and $N = 4$. Note that the outputs of h_4 are fixed and the subfunctions h_1 and h_2 cannot be attacked independently under this attacker's strategy; however, it is easily checked that $v \geq 2$, since again a maximum of two subfunctions can be attacked independently, e.g., h_1 and h_4.

There are generic attacks on the constructions which will be presented in Section VII. The collision attack applied to the above example requires about $3 \cdot 2^{2m/3}$ operations (a second-preimage attack requires $3 \cdot 2^{4m/3}$ operations). Finally, the number of output blocks of the compression function is larger than the security level suggests. To avoid this, it is recommended to use an output transformation which hashes the five blocks to three blocks. Such constructions are discussed in Section VI-B.

A second example is based on an $[8, 5, 3]$ shortened Hamming code, which is obtained by shortening the $[21, 18, 3]$ Hamming code (cf. Section VIII). The resulting hash function has rate $1/4$, and for $m = k = 64$, Theorem 3 shows that finding a collision requires at least 2^{64} encryptions. It is thus comparable to the best scheme by Merkle (cf. Section III-D), but that scheme does not allow for a parallel evaluation. The generator matrix of the $[8, 5, 3]$ Hamming code over GF (2^2) has the following form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & \alpha \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & \beta \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{5}$$

It is now straightforward to derive G' and the hash function in a similar way as for the previous example.

VI. Improved Constructions Using Nonbinary Linear Codes

This section discusses several ways to improve the schemes of Theorem 3.

- A first observation is that the security level of the schemes (for a given rate) can be improved by working with data blocks smaller than m bits. This gives the designer additional degrees of freedom. It will be explained how these constructions can be derived from codes over GF (q), with $q \geq 4$.
- Secondly, the construction can be generalized to compression functions for which the input length is a positive integer multiple of the output length (when using block ciphers, this generalization covers cases where the key length is an integer multiple of the block length).
- An output transformation is added to reduce the size of the output in accordance with the security level, and to avoid potential problems with near-collisions.
- Improved strength against attacks on the hash function can be obtained by a linear transformation of the input variables to the compression function. The security with respect to attacks on the compression is unchanged.

The extended schemes are proposed in Section VI-A, the output transformation is discussed in Section VI-B, and the improved security against attacks on the whole hash function in Section VI-C.

A. The Improved Construction

The first improvement consists of dividing the m-bit words into smaller blocks and to use codes over larger fields. As an example, consider a block cipher with m-bit blocks and m-bit keys for even m. In Theorem 3, codes over GF (2^2) are used, where the two bits of the codewords represent to the plaintext inputs, respectively, the key inputs to the block ciphers. An alternative method is to divide all m-bit blocks into blocks of $m/2$ bits and use codes over GF (2^4). The advantage of this approach, which will be illustrated later in more detail, is that better codes exists over GF (q) if q becomes larger. For example, in a code over GF (2^2) with length $n = 9$ and dimension $k = 6$, the minimum distance is at most 3, e.g., in the shortened Hamming code $[9, 6, 3]$, while over GF (2^3) there exists a $[9, 6, 4]$ code (cf. Section VIII).

The second improvement consists of extending the scheme to compression functions where the input size is any integer multiple of the block length. In the block cipher case, the Davies–Meyer scheme can still be used, but more than one m-bit block enters the key input. This leads to the following theorem.

Theorem 4: Let b be a divisor of m, i.e., $m = b \cdot m_b$, for some positive integer m_b. If there exists an $[n, k, d]$ code over GF $(2^{b(t+1)})$ of length n, dimension k, and minimum distance d, with $(t+1)k > n$, and $m \gg \log_2 n$, then there exists a parallel hash function based on an ideal compression function h: $\{0, 1\}^{tm} \times \{0, 1\}^m \rightarrow \{0, 1\}^m$, for which finding a collision for the compression function requires at least $2^{(d-1)m/2}$ operations provided that Assumption 2 holds. The hash function has an internal memory of $n \cdot m$ bits, has a rate $\rho = (t+1)\frac{k}{n} - 1$, and works on m_b-bit blocks.

Proof: The compression function consists of n different functions h_i with $1 \leq i \leq n$, see Definition 1. The input to the compression function consists of $(t+1)k$ m-bit blocks: the n variables H_{i-1}^1 through H_{i-1}^n (the output of the n functions of the previous iteration) and r message blocks M_i^1 through M_i^r, with $r = (t+1)k - n > 0$. All m-bit blocks are split into subblocks of m_b bits. In the following, every individual bit of these m_b-bit blocks is treated in the same way. The bits of $b(t+1)$ consecutive input blocks are concatenated yielding k elements of GF $(2^{b(t+1)})$. These elements are encoded using the $[n, k, d]$ code, resulting in n elements of GF $(2^{b(t+1)})$. Each of these elements represents the $b(t+1)$-bit inputs to one of the n functions, that is, bt bits correspond to the first input and the remaining b bits correspond the second input to h. The individual input bits are obtained by representing the elements of GF $(2^{b(t+1)})$ as a vector space over GF (2). This construction guarantees that the conditions for Assumption 2 are satisfied for the value $v = d-1$. It follows from the minimum distance of the code that at least d subfunctions are active in a collision. The conclusion follows exactly as in the proof of Theorem 3. □

As an example, a hash function based on a compression function with a $2m$-bit input and an m-bit output (or an m-bit block cipher with an m-bit key) can be constructed by using the code $[8, 6, 3]$ over $\mathrm{GF}(2^4)$, which is obtained by shortening the Hamming code $[17, 15, 3]$. The hash function has rate $1/2$ and an internal memory of $8 \cdot m$ bits, and is thus twice as fast as the example using the code $[8, 5, 3]$ mentioned in Section V-C. With $m = 64$, this construction operates on 32-bit words. One can extend this approach to construct hash functions with codes over $\mathrm{GF}(2^8)$, i.e., operating on 16-bit words, for example, by shortening the $[257, 255, 3]$ Hamming code. However, it will be shown in Section VIII that this does not offer any improvement, unless n is chosen larger than 17. This is interesting from a theoretical point of view but less in practice as the internal memory increases.

B. Output Transformation

The constructions presented in Section VI-A have the following problems.

1) Since every output bit does not depend on all input bits of the compression function, it is relatively easy to find many inputs for which several output blocks of the compression function are equal (such inputs are called “near collisions”).

2) The number of output blocks is typically much larger than the security level suggests. As an example, a construction using the code $[5, 3, 3]$ has hash results of $5m$ bits, but the security level for collision attacks is “only” 2^m, whereas for an ideal construction it would be $2^{2.5m}$.

The solution is to apply an output transformation to the outputs of the compression function. This transformation can be slow, since it has to be applied only once. Therefore, there are many possible constructions.

First an approach is presented that does not affect the provable security of the compression functions. Denote with $n_{\min}$ the smallest possible value of n for a given value of d, such that Theorem 4 holds. If $n_{\min} < n$, compress the n blocks to $n_{\min}$ blocks using the new construction with $n_{\min}$ parallel blocks (this hash function will have a lower rate than the original one). This approach partly solves both problems. However, if a further reduction to less than $n_{\min}$ blocks is required, other approaches are necessary.

The first problem can be overcome taking the following approach. Use the compression function itself as the output transformation, but with the message blocks equal to the values H^1, $H^2, \ldots, H^n$. This has the important advantage that no additional function has to be implemented. In order to use all n inputs, at least $\lceil n/r \rceil$ additional iterations are required. This does not provide sufficient mixing. The number of recommended additional iterations is at least $\lceil n/r \rceil + 1$, and preferably $2 \cdot \lceil n/r \rceil + 1$. Although this approach solves the first of kind of problems, it does not solve the second. If one truncates the output of the last round of the compression function the proof of security fails. However, it can be argued that since a real-life attacker has no control over the blocks which are hashed in the output transformation the security is not threatened. But it is stressed, that there is no proof in the general case, where the attacker is allowed to control all inputs to the compression function.

In constructions using a block cipher the first problem can be solved as follows. One encrypts the n output blocks of the compression function using the block cipher with a fixed, randomly chosen key, such that all output blocks of the encryption depend on all input blocks in a complicated way. One could use, for example, the *all-or-nothing* transform introduced by Rivest [49]. Note that a simple CBC encryption would not be sufficient. Subsequently, the n blocks concatenated with the n encrypted blocks are hashed as above. The second problem can be overcome by the following proposal which can be used instead of or in conjunction with the above first approach. First, one constructs from the m-bit block cipher a large, strong block cipher with block length $n \cdot m$ bits. This block cipher can be slow, since it is applied only once. Subsequently, the n $(n_{\min})$ blocks from the compression function are input to a Davies–Meyer construction where the block cipher key is randomly chosen and fixed (and part of the hash function description). Under Assumption 1 this is a secure hash function. The output can be truncated to any s blocks, where $s \geq d - 1$.

C. Real-Life Attacks

In an attack on a compression function the attacker has full control over all inputs. In an attack on an iterated hash function induced by the compression function an attacker is more restricted. He still has full control over the message variables M_i^k, but the variables H_{i-1}^l are themselves outputs of the compression function in the previous step or are the fixed initial values in the case $i = 1$. As an example, consider the compression function described in the previous section using the code $[5, 3, 3]$. An attacker can fix the variables H_{i-1}^j, for $j = 1, \ldots, 5$, and compute the hash values for all 2^m values of M_i^1. By the birthday paradox, with a high probability he would get a collision in four of the five subfunctions. Such problems can be overcome by applying an affine transformation of the input variables such that the inputs to all five subfunctions depend on the message variables. Note that for any affine transformation of the input variables Theorem 3 would still hold. In the example, one can use the following input vector of the chaining variables:

$$V = [H_{i-1}^1 \oplus M_i^1, H_{i-1}^2 \oplus M_i^1, H_{i-1}^3 \oplus M_i^1, \\ H_{i-1}^4 \oplus M_i^1, H_{i-1}^5, M_i^1].$$

Furthermore, in constructions using a block cipher it is possible to restrict the message variable to as few key inputs as possible to avoid a situation when the attacker can control the keys of the underlying block cipher. The motivation for this is that block ciphers are typically designed to resist attacks where an opponent controls the plaintext or the ciphertext, but where the key is chosen uniformly at random.

VII. A Generic Attack

This section presents a generic attack on the hash functions developed in this paper. The attack makes use of *multicollisions*. A multicollision for h and an element $u \in \mathrm{Range}(h)$ is a set of values $\mathcal{S}$ (with $|\mathcal{S}| \geq 2$), such that $\forall s \in \mathcal{S}, h(s) = u$.

The following lemma is used in the attack (see, for example, Motwani and Raghavan [37, p. 45]).

Lemma 1: When m balls are thrown into n urns, with $m = n \log n$, with probability $1 - O(1)$ every urn contains $O(\log n)$ balls.

An interesting observation is that in this case, the urn with the most balls has about the same number of balls as the expected number of balls in any urn (so the variance of the distribution is very small).

If there are fewer balls, the distribution is less uniform, but the following bound can be proved (see [37, Theorem 3.1, p. 45]).

Lemma 2: When n balls are thrown into n urns, with probability at least $1 - 1/n$ no urn contains more than $\lceil (e \ln n)/\ln \ln n \rceil$ balls.

For $n = 2^{64}$, with probability close to 1 there will be no urn with more than 32 balls. This illustrates that if m is much smaller than in Lemma 1, only multicollisions for small values of $|\mathcal{S}|$ are expected to occur.

Clearly, the complexity for finding a multicollision for a particular value u is higher than for finding a multicollision for one in many values. Although it seems that a multicollision of $\log_2 n$ elements for some element can be found using less than $n \log_2 n$ evaluations, Lemmas 1 and 2 show that the required number of evaluations is not much less than $n \log_2 n$. Moreover, an attacker has no control over the target value u, and will need to store n counters. With $n \log_2 n$ evaluations one can expect many multicollisions of $\log_2 n$ elements. In this case it suffices to keep counters for a few values.

First, a preimage attack is considered. We conjecture that the lower bound for a preimage attack is $2^{(d-1)m}$, but it is an open problem whether our proof techniques can be extended to obtain this bound.

Proposition 3: Consider a multiple hash function construction $\mathcal{H}$ (cf. Definition 1), using an $[n, k, d]$ code over GF $(2^{b(t+1)})$, where $(t+1)k > n$ (cf. Theorem 4). Then a preimage of the compression function of $\mathcal{H}$ can be found in about $\max(2^{m(n-k)}, k \cdot 2^{nm/k})$ operations. The attack requires the storage of about $(t+1)k2^{(n-k)m/k}$ m-bit values.

Proof: First note that it is possible to find an affine transformation of the inputs to the compression functions, such that k subfunctions can be attacked independently. Rearrange the subfunctions such that these k functions come first. The attack is split into two parts. In the first part, one generates a set of multicollisions for each of the k first subfunctions. Then, in the second part, these k sets of multicollisions are combined in all possible ways in order to perform a brute-force attack on the remaining $n-k$ subfunctions. The second part requires about $2^{(n-k)m}$ messages, all of which should hash to the same value in the first k functions. With high probability there will be a match for all n functions. In the first part, by generating $2^{(n-k)\frac{m}{k}+m}$ values of one subfunction, a multicollision on a specific value in the range of a subfunction can be expected where $|\mathcal{S}| \geq 2^{(n-k)\frac{m}{k}}$. Repeating this for each of the first k functions would yield k such sets, which combined would give $2^{(n-k)m}$ inputs all hitting the same output value for the first k subfunctions. Then with high probability a match for all n functions will exist. Note that $(n-k)\frac{m}{k} + m = nm/k$. One problem is whether the entropy of the input to an individual subfunction is sufficiently large to generate that many multicollisions. One subfunction has $2^{(t+1)m}$ inputs, hence it is required that $n/k \leq (t+1)$. However, this is always true since it is also required that $(t+1)k > n$ in the constructions. □

Note: One can show that the effort of the first part dominates that of the second part if k is larger than $1 + 2/n + O(1/n^2)$ and smaller than $n - 1 - 2/n + O(1/n^2)$. This is the case for most values of practical interest, e.g., $n \geq 5$, $d \geq 3$, and t not too large, say $\leq n - 3$.

The following proposition contains the generic attack for collisions.

Proposition 4: Consider a multiple hash function construction $\mathcal{H}$ (cf. Definition 1), using an $[n, k, d]$ code over GF $(2^{b(t+1)})$, where $(t+1)k > n$ (cf. Theorem 4). Then collisions for the compression function of $\mathcal{H}$ can be found in

$$\max\left(2^{m(n-k)/2},\, k \cdot 2^{\frac{(n+k)m}{2k}}\right)$$

operations. The attack requires the storage of about $(t+1)k2^{(n-k)m/(2k)}$ m-bit values.

Proof: This attack is similar to the preimage attack of Proposition 3. In the first part, one generates a set of multicollisions for each of the first k subfunctions. That is, one generates $2^{\frac{n-k}{2}\frac{m}{k}+m}$ values of one subfunction, hence a multicollision will exist where $|\mathcal{S}| \geq 2^{\frac{n-k}{2}\frac{m}{k}}$. Repeating this for each of the first k functions would yield k such sets, which combined would give $2^{\frac{n-k}{2}m}$ texts all hitting the same output value for the first k subfunctions. In the second part, a collision will be found for the remaining subfunctions also with a high probability. Note that

$$\frac{n-k}{2}\frac{m}{k} + m = \frac{(n+k)m}{2k}.$$

Again, one can show that the entropy of the inputs to individual subfunctions is sufficiently large: one subfunction has $2^{(t+1)m}$ inputs, hence it is required that $\frac{n+k}{2k} \leq (t+1)$. However, this is always true since it is also required that $(t+1)k > n$ in the constructions. □

Note: One can show that the effort of the first part dominates that of the second part if k is larger than $1 + 3/(4n) + O(1/n^2)$ and smaller than $n - 2 - 3/(4n) + O(1/n^2)$. This is the case for most values of practical interest, e.g., when $n \geq 5$, $d \geq 4$, and t not too large, say $\leq n - 2$.

Consider a multiple hash constructions based on an $[n, k, d]$ code and an m-bit compression function. It follows from Section V-B that the fastest schemes are for MDS codes. With $n - k = d - 1$, Propositions 3 and 4 state that the generic attacks are close to the lower bounds of security in Theorems 3 and 4. For $n - k = d - 1$ and $d = 3$, the attacks require 2^{2m}, respectively, $k \cdot 2^{(n+k)m/(2k)}$ operations which at least for large values of n and k is close to the lower bounds of Theorem 4. For

$n-k=d-1$ and $d=4$, the attacks require 2^{3m}, respectively, $2^{3m/2}$ operations which match the lower bounds of Theorem 4.

Brudevoll has shown that if $n-k>d-1$, for some constructions there exist attacks (based on multicollisions) with complexities lower than the ones of Propositions 3 and 4 [6]. In these cases, it is advantageous to generate multicollisions for $k' < k$ subfunctions, then combine these to perform a brute-force attack on the $d-1$ remaining active subfunctions. In these cases, preimages and collisions can be found for the compression function in $\max(2^{m(d-1)}, k' \cdot 2^{\frac{(d-1)m}{k'}+m})$, respectively, $\max(2^{m(d-1)/2}, k' \cdot 2^{(d-1)m/2k'+m})$ operations. Examples of such constructions together with other constructions where the best (known) attacks are those of Propositions 3 and 4 are given in [6]. In any case, note that the complexities will never be lower than the bounds of Theorems 3 and 4.

VIII. Error-Correction Codes

The constructions of Sections V and VI rely on the existence of nonbinary linear error-correcting codes of length n, dimension k, and minimum distance d. The conditions on the code are the following.

- The minimum distance d should be sufficiently large, as the security level of the hash function is equal to $2^{(d-1)m/2}$ for collisions.
- The dimension k should be large as well; from the Singleton bound it follows that $k \le n-d+1$. The construction requires that $k > n/(t+1)$, where the original compression function takes an input of size $(t+1)m$ bits and outputs m bits. Recall that the *deficit* δ is defined as $n-k-d+1$. The rate of the hash function can then written as follows:
$$\rho = \frac{(t+1)k}{n} - 1 = t - \frac{(t+1)(d+\delta-1)}{n}.$$
This shows that if n becomes large, the rate of these hash function approaches t. It is shown later that for codes with a fixed value of d and q, δ grows slowly with n. Moreover, the best codes for this construction are MDS codes, i.e., codes with $\delta=0$. However, the existence of nontrivial MDS codes (i.e., MDS codes with $d \ne 1, 2, n$) for large values of n and $k \le q-1$ is an open problem.

First, the easy case of $d=3$ is addressed. This forms a starting point to discuss larger values of d.

A. The Case $d=3$

These codes are the well-known Hamming codes, with parameters given by the following proposition (see, for example, MacWilliams and Sloane [27, pp. 179–180]).

Proposition 5: Let q be a prime power. The (perfect) Hamming codes over GF (q) have the following parameters:
$$\left[n = \frac{q^s-1}{q-1},\, k=n-s,\, d=3\right].$$
They can be shortened up to dimension 1.

Proof: Hamming codes are single-error-correcting codes, or $d=3$. The syndrome of an n-dimensional vector C over GF (q) is equal to $C \cdot H^t$, where H^t is the transpose of the parity matrix. A code can correct a single error if the syndrome of all individual errors is different, and different from 0. The syndrome of a single error is a nonzero multiple of a column of H^t and thus of a row of H. Therefore, it is sufficient that all rows of H are nonzero and are not a multiple of each other. As the parity matrix H has n rows with $n-k$ components, a code can only correct a single error if there are at least n nonzero rows that satisfy the conditions, or
$$\frac{q^{n-k}-1}{q-1} \ge n.$$
It is easy to see that equality can only be achieved if n and k are as stated in the proposition (note that $s=n-k$). If the code is shortened, that is, n and k are reduced by one, the inequality will still be satisfied. □

It follows immediately that the value of δ is equal to $s-2$. For small values of q, one obtains the following results:

$q=4$: If $s=2$, one obtains the $[5, 3, 3]$ code of Section V-C. This is an MDS code, or $\delta=0$. For $s=3$, one finds a $[21, 18, 3]$ code that can be shortened to the $[8, 5, 3]$ code of Section V-C. Note that for this code, and for its shortened versions, one has $\delta=1$.

$q=8$: $s=2$ yields the $[9, 7, 3]$ Hamming code, which is MDS. The case $s=3$ results in the $[73, 70, 3]$ Hamming code.

$q=16$: $s=2$ yields the $[17, 15, 3]$ Hamming code, which is MDS.

Proposition 6: There exist parallel hash functions based on an ideal compression function h: $\{0, 1\}^m \times \{0, 1\}^{tm} \rightarrow \{0, 1\}^m$, with rates close to t for which finding a collision takes at least 2^m, respectively, at least $2^{3m/2}$ operations.

Proof: From Theorem 4 it follows that such hash functions exist if there exist Hamming codes with δ sufficiently small. For Hamming codes
$$\delta = \lfloor \log_q(n) \rfloor - 1$$
and thus the rate is equal to
$$\rho = \frac{(t+1)k}{n} - 1 = t - \frac{(t+1)(\lfloor \log_q(n) \rfloor + 1)}{n}.$$
This implies that if n becomes large, the rate approaches t quickly. □

To illustrate this result: using the $[17, 15, 3]$ Hamming code with $t=1$, the rate becomes
$$\rho = t - \frac{(t+1)2}{17} = \frac{13}{17} = 0.76.$$
In constructions using a block cipher, this has the following implications.

Corollary 1: Provided that Assumption 2 holds, there exist parallel hash functions based on an m-bit block cipher with a tm-bit key with rates close to t for which finding a collision takes at least 2^m operations, respectively, at least $2^{3m/2}$ operations.

At the cost of a larger internal memory, using DES one can obtain hash functions of rate 1, and using IDEA one can

TABLE II

MINIMUM DISTANCE d FOR THE BEST CODES KNOWN OVER GF (2^2) WITH LENGTH n AND DIMENSION k. THE LINES SEPARATE THE AREAS WITH $k > n/2$ AND $k > n/3$. MDS CODES ARE INDICATED WITH A BOLD NUMBER. THE SYMBOL "–" MEANS THAT A CODE WITH $d \geq 3$ WOULD VIOLATE THE SINGLETON BOUND

	dimension k											
length n	3	4	5	6	7	8	9	10	11	12	13	14
5	**3**	–										
6	4		–									
7	4	3		–								
8	5	4	3		–							
9	6	5	4	3		–						
10	6	6	5	4	3		–					
11	7	6	6	5	4	3		–				
12	8	7	6	6	4	4	3		–			
13	9	8	7	6	5	4	4	3		–		
14	10	9	8	7	6	5	4	4	3		–	
15	11	10	8	8	7	6	5	4	4	3		–
16	12	11	9	8	8	7	6	5	4	4	3	

TABLE III

MINIMUM DISTANCE d FOR THE BEST CODES KNOWN OVER GF (2^3) WITH LENGTH n AND DIMENSION k. THE LINES SEPARATE THE AREAS WITH $k > n/2$ AND $k > n/3$. MDS CODES ARE INDICATED WITH A BOLD NUMBER. THE SYMBOL "–" MEANS THAT A CODE WITH $d \geq 3$ WOULD VIOLATE THE SINGLETON BOUND. A $*$ SYMBOL MEANS THAT A CODE WITH MINIMUM DISTANCE $d + 1$ MIGHT EXIST

	dimension k											
length n	3	4	5	6	7	8	9	10	11	12	13	14
5	**3**	–										
6	**4**	**3**	–									
7	**5**	**4**	**3**	–								
8	**6**	**5**	**4**	**3**	–							
9	**7**	**6**	**5**	**4**	**3**	–						
10	**8**	6	5	4	**4**		–					
11	8	7	6	5	4	3		–				
12	9	8	7	6	5	4	3		–			
13	10	9	8	7	6	5	4	3		–		
14	11	10	9	8	7	6	5	4	3		–	
15	12	10*	9*	8*	8	7	6	5	4	3		–
16	12*	11*	10*	9*	8*	8	6*	6	5	4	3	

obtain hash functions of rate 2. For comparison MDC-2 and MDC-4 have rates $1/2$, respectively, $1/4$, and Abreast-DM and Tandem-DM developed for IDEA [26] have rates $1/2$ (cf. Section III).

B. The Case $d \geq 4$

From the previous section one can conclude that MDS codes only exist for values of n below a certain threshold; the value of this threshold increases with q. It is also known that nontrivial MDS codes over GF (q) exist only when $k \leq q - 1$ and $d \leq q$. Once one has an MDS code, shortening it will result in another MDS code with the same minimum distance. This follows from the fact that a necessary and sufficient condition for an MDS code is that every square submatrix of size $\min\{n, n-k\}$ of the matrix A, defined by $G = [I_{k\times k} | A_{k\times(n-k)}]$, is nonsingular [27, p. 321].

For the parameters which are of interest to the new construction, the following theorem presents doubly and triply extended Reed-Solomon codes that are MDS [27, pp. 323–326].

Theorem 5: For any k, $1 \leq k \leq q+1$, there exists a $[q{+}1,\, k,\, q{-}k{+}2]$ cyclic MDS code over GF (q), and there exist $[2^s{+}2,\, 3,\, 2^s]$ and $[2^s{+}2,\, 2^s-1,\, 4]$ MDS codes over GF (2^s).

The fact that some of these codes are cyclic can be used to develop a compact description (cf. Section IX). For $q = 4$, no codes of $d \geq 4$ are obtained using this construction. For larger values of q, one obtains the following results.

> $q = 8$: the preceding theorem results in the following codes with $d \geq 4$: [9, 6, 4], [9, 5, 5], [9, 4, 6], and [10, 7, 4]. Here the first three codes are cyclic.
>
> $q = 16$: there exist MDS codes for all values of n up to 17, $1 \leq k \leq n$. In addition, there exist an [18, 3, 16] and an [18, 15, 4] code. The [17, 14, 4] and [17, 13, 5] codes are cyclic.

As values of n larger than this are not of importance for practical constructions, there is no reason to use values of q larger than 16.

Tables II and III indicate the values of n and k for which a linear code exist with minimum distance $d \geq 3$ for $q = 4$ and 8, respectively. These tables have been obtained from [5]. For $q = 4$, there is only one MDS code with $d \geq 3$, namely, the [5, 3, 3] Hamming code. For $q = 8$, there exist MDS codes satisfying the conditions for $5 \leq n \leq 9$; one also has the [10, 7, 4] code of Theorem 5. For $n \geq 16$ and $d \leq 6$, one can always find a code with $\delta = 1$. This illustrates that not much can be gained from taking $q = 16$. For $q = 16$, MDS codes exist for $n \leq 17$, $1 \leq k \leq n$, and for $[18, 3, 16]$ and $[18, 15, 4]$. This is certainly sufficient for the hash functions considered in this paper.

IX. Some Practical Examples

This section contains some examples of new constructions for several parameters of the underlying compression function. The examples in the first two subsections are especially suited for constructions where the compression function is obtained from a block cipher. In the following, (m, k) denotes an m-bit block cipher with a k-bit key. The complexities of the attacks on the examples are against the compression function. Attacks against the hash function can be higher according to Sections VI-B and VI-C. However, it is recommended when designing a hash function to choose a construction with a large security against attacks on the compression functions.

A. Using an (m, m)-Block Cipher

Tables IV and V list the rates and the best known attacks for the existing constructions and the constructions proposed in this paper respectively.

In what follows, an implementation of the construction using the code $[9, 6, 4]$ is shown. Define GF (2^4) as the extension field GF $(2)[x]/(x^4 + x + 1)$. There are many generator matrices for a $[9, 6, 4]$ linear code over GF (2^4). A generator matrix was chosen which leads to a simple and efficient compression function, as explained later. The generator matrix has the following form:

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \boldsymbol{\alpha} & \boldsymbol{\beta} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \boldsymbol{\beta} & \boldsymbol{\alpha} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \boldsymbol{\alpha} & \mathbf{1} & \boldsymbol{\beta} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \boldsymbol{\alpha} & \boldsymbol{\beta} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \boldsymbol{\beta} & \mathbf{1} & \boldsymbol{\alpha} \end{bmatrix}. \tag{6}$$

Here $\mathbf{0}$ and $\mathbf{1}$ are the additive and multiplicative neutral elements in GF (2^4) and $\boldsymbol{\alpha} = x$, and $\boldsymbol{\beta} = x^3 + x^2 + 1$. The motivation for the choice of the generator matrix is as follows. In an implementation of the compression function the elements of GF (2^4) are represented as elements of a vector space over GF (2). Clearly, multiplications with $\mathbf{0}$ and $\mathbf{1}$ are the easiest to implement. A closer analysis shows that multiplication with $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in the above example can be implemented with one, respectively, two EXCLUSIVE-ORs. An exhaustive search for the matrix with the easiest implementation was not feasible (w.r.t. our computing resources), but by restricting a search to using the elements $\mathbf{0}$, $\mathbf{1}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ a solution close to the optimal one was obtained.

Let $f_i(X, Y)$ be different instants of the function $E_X(Y) \oplus Y$, let X_L and X_R denote the leftmost, respectively, rightmost $m/2$ bits of X, and let $\|$ denote concatenation of $m/2$ bit blocks. Furthermore, let $G^1, \ldots, G^9$ be the nine input blocks coming from the compression function in the previous iteration and let M^1, M^2, M^3 be the three message block inputs. This results in the compression function depicted in Table VI.

TABLE IV
Rates and Complexities of Previous Proposals for (m, m)-Block Ciphers

Scheme	Rate ρ	Collision h	Collision Hash	Reference
MDC-2	1/2	$2^{m/2}$	2^m	[4]
MDC-4	1/4	$2^{3m/4}$	2^m	[4]
Merkle	0.27	2^m	2^m	[32]

TABLE V
Comparison of Constructions Based on Codes Over GF (2^2) and Over GF (2^4) for (m, m)-Block Ciphers

$GF(2^2)$		$GF(2^4)$		Collision
Code	Rate ρ	Code	Rate ρ	
$[5,3,3]$	1/5	$[6,4,3]$	1/3	$\geq 2^m$
$[8,5,3]$	1/4	$[8,6,3]$	1/2	$\geq 2^m$
$[12,9,3]$	1/2	$[12,10,3]$	2/3	$\geq 2^m$
$[9,5,4]$	1/9	$[9,6,4]$	1/3	$\geq 2^{3m/2}$
$[16,12,4]$	1/2	$[16,13,4]$	5/8	$\geq 2^{3m/2}$

As an output transformation one can first hash the nine blocks to seven blocks via the compression function using the code $[7, 4, 4]$ ($n_{\min} = 7$ for $d = 4$) and then hash the seven blocks to three blocks using one of the approaches described in Section VI-B.

B. Using an $(m, 2m)$-Block Cipher

The only known hash functions based on an $(m, 2m)$-block cipher with a $2m$-bit hash result are the Abreast-DM and the Tandem-DM from [26] (cf. Section III-C). Table VII lists the rates and complexities of the best known attacks on the two constructions.

However, as already indicated, there exist more efficient constructions with a higher security level. Table VIII lists the rates and complexities of such constructions. As before, it is possible divide the m-bit blocks into smaller subblocks. For example, the blocks can be divided into halves and expanded with a code over GF (2^6), such as $[65, 63, 3]$.

C. Using the MDx Family

Dobbertin's attack on the extended MD4 [12], [53] shows that for MD4 even two dependent runs of the compression function are not collision resistant. However, it seems unlikely that his attacks extend to compression functions consisting of two or more instantiations of MD5. The methods developed in this paper can be used to construct parallel MD5 hash functions based on linear codes over GF (2^5). In Table IX possible constructions are listed.

Since the assumption for these constructions, that is, that the basic components are secure, does not hold for MD4 and MD5, explicit bounds for the complexities of collision attacks on the compression functions have not been specified. However, it is conjectured that for the constructions using MD5 and codes of minimum distance 4, a collision attack is infeasible. The attack requires a simultaneous collision for at least three different instances with dependent inputs.

TABLE VI
COMPRESSION FUNCTION USING THE CODE $[9, 6, 4]$ IN GF (2^4)

$$\begin{aligned}
H^1 &= f_1(G^1, G^2)\\
H^2 &= f_2(G^3, G^4)\\
H^3 &= f_3(G^5, G^6)\\
H^4 &= f_4(G^7, G^8)\\
H^5 &= f_5(G^9, M^1)\\
H^6 &= f_6(M^2, M^3)\\
H^7 &= f_7(G^1 \oplus G^3 \oplus G^5 \oplus (G^7_R \| G^8_L) \oplus (G^9_R \| M^1_L) \oplus (M^3_{LR} \| M^3_R),\\
&\quad G^2 \oplus G^4 \oplus G^6 \oplus (G^7_L \oplus G^8_R \| G^7_L) \oplus (G^9_L \oplus M^1_R \| G^9_L) \oplus (M^2_L \| M^2_R \oplus M^3_{LR}))\\
H^8 &= f_8(G^1 \oplus G^7 \oplus M^2 \oplus (G^3_R \| G^4_L) \oplus (G^6_{LR} \| G^6_R) \oplus (M^1_{LR} \| M^1_R),\\
&\quad G^2 \oplus G^8 \oplus M^3 \oplus (G^4_R \oplus G^3_L \| G^3_L) \oplus \mathcal{G}^5 \oplus (G^9_L \| G^9_R \oplus M^1_{LR}))\\
H^9 &= f_9(G^1 \oplus G^9 \oplus (G^4_{LR} \| G^4_R) \oplus (G^5_R \| G^6_L) \oplus (G^8_{LR} \| G^8_R) \oplus (M^2_R \| M^3_L),\\
&\quad G^2 \oplus M^1 \oplus \mathcal{G}^3 \oplus (G^6_R \oplus G^5_L \| G^5_L) \oplus \mathcal{G}^7 \oplus (M^2_L \oplus M^3_R \| M^2_L)),
\end{aligned}$$

where $\mathcal{G}^t = (G^t_L \| G^t_R \oplus G^{t+1}_L \oplus G^{t+1}_R)$ and $X_{LR} = X_L \oplus X_R$.

TABLE VII
RATES AND COMPLEXITIES OF PREVIOUS PROPOSALS FOR $(m, 2m)$-BLOCK CIPHERS [26]

Scheme	Rate ρ	Collision h	Collision Hash
Abreast-DM	1/2	2^m	2^m
Tandem-DM	1/2	2^m	2^m

TABLE VIII
RATES AND COMPLEXITIES OF THE PROPOSALS FOR $(m, 2m)$-BLOCK CIPHERS USING CODES OVER GF (2^3)

Code	Rate ρ	Collision
$[4, 2, 3]$	1/2	$\geq 2^m$
$[6, 4, 3]$	1	$\geq 2^m$
$[9, 7, 3]$	4/3	$\geq 2^m$
$[5, 2, 4]$	1/5	$\geq 2^{3m/2}$
$[7, 4, 4]$	5/7	$\geq 2^{3m/2}$
$[10, 7, 4]$	11/10	$\geq 2^{3m/2}$

TABLE IX
RATES AND COMPLEXITIES OF THE PROPOSALS FOR THE MDx FAMILY USING CODES OVER GF (2^5)

Scheme	Rate	Scheme	Rate	Scheme	Rate
	ρ		ρ		ρ
MD4	4	$[5, 3, 3]$	2	$[5, 2, 4]$	1
MD5	4	$[10, 8, 3]$	3	$[10, 7, 4]$	2.5
		$[20, 18, 3]$	3.5	$[20, 17, 4]$	3.25

D. Remarks on Implementations

The description of the hash functions used in the previous sections can be made more compact by considering a cyclic representation of the used codes. As an example, consider the proposal based on the cyclic $[9, 6, 4]$ Reed–Solomon code over GF (2^4); a cyclic representation can be used with

$$g(x) = x^3 + (\boldsymbol{\beta} + \mathbf{1})x^2 + (\boldsymbol{\beta} + \mathbf{1})x + \mathbf{1}$$

where $\boldsymbol{\beta}^4 = \boldsymbol{\beta} + \mathbf{1}$. The cyclic representation leads to a compact and simple description, which is easier to implement and test. The representation given in the previous section, however, has a smaller overhead in terms of the number of EXCLUSIVE-ORS.

X. CONCLUSION

This paper has presented a new method for constructing hash functions based on a small compression function. Using block ciphers such as DES this yields hash functions which are faster and more secure than existing proposals. The method extends to block ciphers such as IDEA and AES where the block size and key size are different. For large values of internal memory, constructions using IDEA exist with rates close to two, which is a factor of four faster than existing proposals. Finally, the application of the method to the MDx family has been discussed. Two schemes derived from the constructions proposed in this paper have been included in the revised edition (2000) of ISO/IEC Std.10118-2 [20].

ACKNOWLEDGMENT

The authors are grateful to E. Brudevoll and to the anonymous referees for many helpful comments. Also, they wish to acknowledge the helpful discussions with T. Helleseth and T. Kløve about codes.

REFERENCES

[1] W. Aiello and R. Venkatesan, "Foiling birthday attacks in length-doubling transformations. Benes: A nonreversible alternative to Feistel," in *Advances in Cryptology, Proc. Eurocrypt'96 (Lecture Notes in Computer Science)*, U. Maurer, Ed. Berlin, Germany: Springer-Verlag, 1996, vol. 1070, pp. 307–320.

[2] W. Aiello, S. Haber, and R. Venkatesan, "New constructions for secure hash functions," in *Fast Software Encryption (Lecture Notes in Computer Science)*, S. Vaudenay, Ed. Berlin, Germany: Springer-Verlag, 1998, vol. 1372, pp. 150–167.

[3] M. Bellare and P. Rogaway, "Toward making UOWHF's practical," in *Advances in Cryptology, Proc. Crypto'97 (Lecture Notes in Computer Science)*, B. Kaliski, Ed. Berlin, Germany: Springer-Verlag, 1997, vol. 1294, pp. 470–484.

[4] B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas, C. H. Meyer, J. Oseas, S. Pilpel, and M. Schilling, "Data authentication using modification detection codes based on a public one way encryption function," U.S. Patent 4 908 861, Mar. 13, 1990.

[5] A. E. Brouwer. Linear code bound. [Online]. Available: http://www.win.tue.nl/win/math/dw/voorlincod.html.

[6] E. Brudevoll, "Iterated cryptographic hash functions," Master thesis, Univ. Bergen, Bergen, Norway, Nov. 1999.

[7] J. Daemen and V. Rijmen. (1999, Sept.) AES proposal Rijndael. [Online]. Available: http://www.nist.gov/aes.

[8] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology, Proc. Crypto'89 (Lecture Notes in Computer Science)*, G. Brassard, Ed. Berlin, Germany: Springer-Verlag, 1990, vol. 435, pp. 416–427.

[9] D. Davies and W. Price, *Security for Computer Networks*, 2nd ed. New York: Wiley, 1989.

[10] B. den Boer and A. Bosselaers, "Collisions for the compression function of MD5," in *Advances in Cryptology, Proc. Eurocrypt'93 (Lecture Notes in Computer Science*, T. Helleseth, Ed. Berlin, Germany: Springer-Verlag, 1994, vol. 765, pp. 293–304.

[11] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[12] H. Dobbertin, "Cryptanalysis of MD4," *J. Cryptol.*, vol. 11, no. 4, pp. 253–271, 1998.

[13] ——, "The status of MD5 after a recent attack," *CryptoBytes*, vol. 2, no. 2, pp. 1–6, Summer 1996.

[14] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD," in *Fast Software Encryption (Lecture Notes in Computer Ssciene)*, D. Gollmann, Ed. Berlin, Germany: Springer-Verlag, 1996, vol. 1039, pp. 71–82.

[15] Federal Information Processing Std. (FIPS) 46, "Data Encryption Standard," Nat. Bur. Stand., U.S. Dept. Commerce, Washington, DC, Jan. 1977.

[16] Federal Information Processing Std. (FIPS) 180-1, "Secure Hash Standard," NIST, U.S. Dept. Commerce, Washington, DC, Apr. 1995.

[17] NIST, "SHA-256, SHA-384, SHA-512," U.S. Dept. Commerce, Draft, Washington, DC, 2000.

[18] Federal Information Processing Std. (FIPS) 197, "Advanced Encryption Standard (AES)," NIST, U.S. Dept. Commerce, Washington, DC, Nov. 26, 2001.

[19] W. Hohl, X. Lai, T. Meier, and C. Waldvogel, "Security of iterated hash functions based on block ciphers," in *Advances in Cryptology, Proc. Crypto'93 (Lecture Notes in Computer Science)*, D. Stinson, Ed. Berlin, Germany: Springer-Verlag, 1994, vol. 773, pp. 379–390.

[20] ISO/IEC, "Information technology—Security techniques—Hash-functions, Part 1: General and Part 2: Hash-functions using an n-bit block cipher algorithm," Std. 10118, revision of 1994 ed., 2000.

[21] L. R. Knudsen, "New potentially 'weak' keys for DES and LOKI," in *Advances in Cryptology, Proc. Eurocrypt'94 (Lecture Notes in Computer Science)*, A. De Santis, Ed. Berlin, Germany: Springer-Verlag, 1995, vol. 959, pp. 419–424.

[22] ——, "A detailed analysis of SAFER K," *J. Cryptol.*, vol. 13, no. 4, pp. 417–436, 2000.

[23] L. R. Knudsen, X. Lai, and B. Preneel, "Attacks on fast double block length hash functions," *J. Cryptol.*, vol. 11, no. 1, pp. 59–72, Winter 1998.

[24] L. R. Knudsen and B. Preneel, "Hash functions based on block ciphers and quaternary codes," in *Advances in Cryptology, Proc. Asiacrypt'96 (Lecture Notes in Computer Science)*, K. Kim and T. Matsumoto, Eds. Berlin, Germany: Springer-Verlag, 1996, vol. 1163, pp. 77–90.

[25] L. R. Knudsen and B. Preneel, "Fast and secure hashing based on codes," in *Advances in Cryptology, Proc. Crypto'97 (Lecture Notes in Computer Science)*, B. Kaliski, Ed. Berlin, Germany: Springer-Verlag, 1997, vol. 1294, pp. 485–498.

[26] X. Lai, "On the design and security of block ciphers," in *ETH Series in Information Processing*, J. L. Massey, Ed. Konstanz, Germany: Hartung-Gorre Verlag, 1992, vol. 1.

[27] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1978.

[28] J. L. Massey, "SAFER K-64: A byte-oriented block-ciphering algorithm," in *Fast Software Encryption (Lecture Notes in Computer Science)*, B. Preneel, Ed. Berlin, Germany: Springer-Verlag, 1995, vol. 1008, pp. 1–17.

[29] S. M. Matyas, C. H. Meyer, and J. Oseas, "Generating strong one-way functions with cryptographic algorithm," *IBM Tech. Discl. Bull.*, vol. 27, no. 10A, pp. 5658–5659, 1985.

[30] A. Menezes, P. C. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 1997.

[31] R. Merkle, *Secrecy, Authentication, and Public Key Systems*. Ann Arbor, MI: UMI Res., 1979.

[32] ——, "One way hash functions and DES," in *Advances in Cryptology, Proc. Crypto'89 (Lecture Notes in Computer Science)*, G. Brassard, Ed. Berlin, Germany: Springer-Verlag, 1990, vol. 435, pp. 428–446.

[33] ——, "A fast software one-way hash function," *J. Cryptol.*, vol. 3, no. 1, pp. 43–58, 1990.

[34] C. H. Meyer and M. Schilling, "Secure program load with manipulation detection code," in *Proc. Securicom 1988*, pp. 111–130.

[35] S. Miyaguchi, M. Iwata, and K. Ohta, "New 128-bit hash function," in *Proc. 4th Int. Joint Workshop on Computer Communications*, Tokyo, Japan, July 13–15, 1989.

[36] J. H. Moore and G. J. Simmons, "Cycle structure of the DES for keys having palindromic (or antipalindromic) sequences of round keys," *IEEE Trans. Software Eng.*, vol. SE-13, no. 2, pp. 262–273, 1987.

[37] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[38] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proc. 21st ACM Symp. Theory of Computing*, 1989, pp. 387–394.

[39] B. Preneel, "Analysis and design of cryptographic hash functions," Ph.D. dissertation, Katholieke Universiteit Leuven, Leuven, Belgium, Jan. 1993.

[40] B. Preneel, "The state of cryptographic hash functions," in *Lectures on Data Security (Lecture Notes on Computer Science)*, I. Damgård, Ed. Berlin, Germany: Springer-Verlag, 1999, vol. 1561, pp. 158–182.

[41] B. Preneel, R. Govaerts, and J. Vandewalle, "On the power of memory in the design of collision resistant hash functions," in *Advances in Cryptology, Proc. Auscrypt'92 (Lecture Notes in Computer Science)*, J. Seberry and Y. Zheng, Eds. Berlin, Germany: Springer-Verlag, 1993, vol. 718, pp. 105–121.

[42] B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: A synthetic approach," in *Advances in Cryptology, Proc. Crypto'93 (Lecture Notes in Computer Science)*, D. Stinson, Ed. Berlin, Germany: Springer-Verlag, 1994, vol. 773, pp. 368–378.

[43] B. Preneel and P. C. van Oorschot, "On the security of iterated MAC algorithms," *IEEE Trans. Inform. Theory*, vol. 45, pp. 188–199, Jan. 1999.

[44] V. Rijmen and B. Preneel, "Improved characteristics for differential cryptanalysis of hash functions based on block ciphers," in *Fast Software Encryption (Lecture Notes in Computer Science)*, B. Preneel, Ed. Berlin, Germany: Springer-Verlag, 1995, vol. 1008, pp. 242–248.

[45] J.-J. Quisquater and J.-P. Delescaille, "'How easy is collision search? Application to DES'," in *Advances in Cryptology, Proc. Eurocrypt'89 (Lecture Notes in Computer Science)*, J.-J. Quisquater and J. Vandewalle, Eds. Berlin, Germany: Springer-Verlag, 1990, vol. 434, pp. 429–434.

[46] R. L. Rivest, "The MD4 message digest algorithm," in *Advances in Cryptology, Proc. Crypto'90 (Lecture Notes in Computer Science)*, S. Vanstone, Ed. Berlin, Germany: Springer-Verlag, 1991, vol. 537, pp. 303–311.

[47] ——, "The MD5 message-digest algorithm," in *Request for Comments (RFC) 1321*, Apr. 1992. Internet Engineering Task Force (IETF). Available: [Online] http://www.ietf.org/.

[48] ——, "The RC5 encryption algorithm," in *Fast Software Encryption (Lecture Notes in Computer Science)*, B. Preneel, Ed. Berlin, Germany: Springer-Verlag, 1995, vol. 1008, pp. 86–96.

[49] ——, "All-or-nothing encryption and the package transform," in *Fast Software Encryption (Lecture Notes in Computer Science)*, E. Biham, Ed. Berlin, Germany: Springer-Verlag, 1997, vol. 1267, pp. 210–218.

[50] D. Simon, "Finding collisions on a one-way street: Can secure hash functions be based on general assumptions?," in *Advances in Cryptology, Proceedings Eurocrypt'98 (Lecture Notes in Computer Science)*, K. Nyberg, Ed. Berlin, Germany: Springer-Verlag, 1998, vol. 1403, pp. 334–345.

[51] P. C. van Oorschot and M. J. Wiener, "Parallel collision search with cryptanalytic applications," *J. Cryptol.*, vol. 12, no. 1, pp. 1–28, 1999.

[52] G. Yuval, "How to swindle Rabin," *Cryptologia*, vol. 3, no. 3, pp. 187–189, 1979.

[53] H. Dobbertin, "Cryptanalysis of MD4," in *Fast Software Encryption (Lecture Notes in Computer Science)*, D. Gollmann, Ed. Berlin, Germany: Springer-Verlag, 1996, vol. 1039, pp. 53–69.