Combining Compression Functions and Block Cipher-Based Hash Functions

Thomas Peyrin[1]
Henri Gilbert[1], Frédéric Muller[2], and Matt Robshaw[1]

[1] France Télécom R&D, Issy les Moulineaux, France,
[2] HSBC, Paris, France
{thomas.peyrin,henri.gilbert,matt.robshaw}@orange-ft.com
frederic.muller@hsbc.fr

Abstract. The design of secure compression functions is of vital importance to hash function development. In this paper we consider the problem of combining smaller trusted compression functions to build a larger compression function. This work leads directly to impossibility results on a range of block cipher-based hash function constructions.

Key words: block ciphers, compression functions, hash functions.

1 Introduction

Cryptographic hash functions are an important tool in cryptography. Informally, a cryptographic hash function H takes an input of variable size and returns a hash value of fixed length while satisfying the properties of preimage resistance, second preimage resistance, and collision resistance [26]. For a secure hash function that gives an n-bit output, compromising these properties should require 2^n, 2^n, and $2^{n/2}$ operations respectively.

The pioneering work of Merkle and Damgård [7,27] showed how to construct a secure hash function from a *compression function* h that has a fixed-length input, consisting of a *chaining variable* and a message extract, and gives a fixed-length output. A variety of interesting results [8,12,13] have provided a greater understanding of the Merkle-Damgård approach to the serial application of such a compression function.

Generally speaking, there are two popular approaches to building a compression function for use in a crytographic hash function. The first is to use a compression function of a dedicated design while the second is to build a compression function around an established, and trusted, block cipher. While most widely-deployed hash functions [30,37] use a compression function of dedicated design, recent attacks [39,40] have demonstrated that there is much to learn. Instead, there is now much renewed interest in using a block cipher as the basis for a compression function.

It might be argued that the compression functions of common dedicated hash functions such as MD5 [37] and SHA-1 [30] are built on block ciphers; by removing the feed-forward from compression functions in the MD-family we are left with a reversible component that can be used as a block cipher (such as SHACAL [9] in the case of SHA-1). But these block ciphers cannot be afforded the same level of trust as the leading standardised block ciphers [29,31], and instead block cipher-based hash functions are traditionally viewed as techniques to build a secure compression function from a trusted and standardised cipher. Much progress on using block ciphers in this way has already been made. Black *et al* [2] built on the work of Preneel [32] to present a range of secure $2n$- to n-bit compression functions built around an n-bit block cipher that takes an n-bit key. Among these are the well-known Davies-Meyer, Matyas-Meyer-Oseas, and Miyaguchi-Preneel constructions. We therefore have many secure compression functions in hand whose chaining variable is the same size as the block size. However, a hash function built on a compression function with n bits of output can only offer a security level of at most $2^{n/2}$ operations. Since a security level of 2^{128} bits is often desired, we need to construct compression functions with outputs of at least 256 bits, a requirement that cannot be immediately met by the standardised block ciphers in hand.

Our difficulties begin, therefore, when we try to build secure compression functions whose output size is greater than the block size of the underlying block cipher. This is not a new problem and there has been mixed success in constructing $2n$-bit hash functions from an n-bit block cipher [4,5,14,19,21,33,35]. While limitations have been identified in many constructions [14], Hirose [10] has demonstrated the security of a family of double block-length hash functions using two independent block ciphers with key length twice the block length. This is a property shared by AES-256 [29] and IDEA [20] among others with a particular instance of this construction being the long-standing ABREAST-DM [19].

While the case of block ciphers provided the initial motivation for our work, our results are essentially about compression functions. In this paper we explore the problem of combining compression functions that we know to be secure. These smaller compression functions can be of any type—dedicated, number theoretic, block cipher-based—and our aim is to build a secure compression function with a longer chaining variable. Thus the results are broader than block cipher-based hashing, though this is where there is an immediate, practical, and at times surprising, impact. The paper is organised as follows. In Section 2 we establish the framework and we make some initial observations in Section 3. After discussing some generic attacks in Section 4, we derive criteria for combining compression functions in Section 5 and demonstrate a range of impossibility results and potential constructions in Section 6. We then draw our conclusions and highlight opportunities for future work.

2 Notation and Model

In this paper we consider building larger compression functions from smaller trusted ones. We will assume that the underlying secure compression functions have k inputs of n bits and that the output is n bits in length. Details on the construction of secure compression functions will not be important to our results. However, in the specific case of a block cipher with equal key and block size we have $k = 2$, while for a key size twice the block size we have $k = 3$. We could also use a compression function based on a tweaked block cipher [3,23] or a dedicated design (if we were willing to claim their security as secure compression functions) and we might then have $k > 3$ depending on the sizes of the chaining variable and message input. This flexible approach was pursued by Knudsen and Preneel in a series of papers [16,17,18].

This work is not a proof oriented paper, so we follow [18]: a collision resistant hash function or compression function outputing n bits is called *ideal* if the best algorithm to find a collision is a brute-force collision search; such an attack requires on average $\Theta(2^{n/2})$ evaluations of the hash function. Similarly, a preimage (resp. 2^{nd}-preimage) resistant hash function or compression function with n-bit output is called *ideal* if the best algorithm to find a preimage (resp. 2^{nd}-preimage) is a brute-force preimage (resp. 2^{nd}-preimage) search; such an attack requires on average $\Theta(2^n)$ evaluations of the hash function.

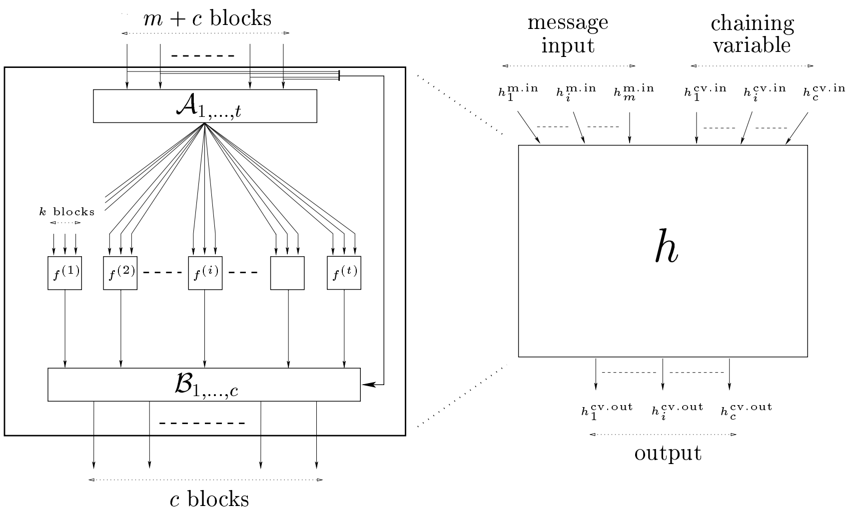

Fig. 1. The compression function h built from t compression functions $f^{(i)}$ each taking k inputs of n bits and delivering an n-bit output. m stands for message and cv for chaining variable.

In our constructions we will use t ideal n-bit compression functions to construct a secure compression function h that compresses $(m+c)n$ bits to cn bits. One important aspect to what follows in this paper is that we require the t internal ideal compression functions to act independently. Exactly how these are instantiated is outside the scope of this paper, but it is an important issue in practice. It is, however, an issue that has been addressed before and, under the assumption that the underlying block cipher is good, we can enforce independence of the fundamental compression functions by fixing bits of the underlying "keys" to distinct values [18] or by using constants [11] to diversify the "keys" used in the compression function.

We will describe the inputs (resp. outputs) to the internal component compression functions as *internal inputs* (resp. *internal outputs*). These are distinguished from the *external inputs* and *external outputs* to the larger compression function h that we are trying to build. The $m+c$ inputs to h, each of n bits, will be denoted by $h_1^{\mathrm{m.in}}, \ldots, h_m^{\mathrm{m.in}}, h_1^{\mathrm{cv.in}}, \ldots, h_c^{\mathrm{cv.in}}$ and we denote the c n-bit output blocks by $h_1^{\mathrm{cv.out}}, \ldots, h_c^{\mathrm{cv.out}}$.

The internal inputs will be derived as a linear combination of the external inputs to h, and we will derive the output from h as a linear combination of the internal outputs from the t ideal compression functions. Thus, the kt inputs to the internal compression functions $f_j^{(i)}$ ($1 \le i \le t$ and $1 \le j \le k$) will be linear functions of the external inputs and for each compression function $f^{(i)}$ we have

$$\begin{pmatrix} f_1^{(i)} \\ \vdots \\ f_k^{(i)} \end{pmatrix} = \mathcal{A}_i \cdot (h_1^{\mathrm{m.in}}, \ldots, h_m^{\mathrm{m.in}}, h_1^{\mathrm{cv.in}}, \ldots, h_c^{\mathrm{cv.in}})^T.$$

where $\mathcal{A}_i$ is a $(k \cdot n \times (m+c) \cdot n)$ binary matrix, consisting of $(n \times n)$ blocks which are either zero or the identity matrix, corresponding to the compression function $f^{(i)}$. Taken together, such matrices define a mixing layer among the inputs to the t compression functions and we call this the *input layer*. Similarly, the external outputs from h are any linear combination of the t compression function outputs. This is the *output layer* and for the external outputs $h_i^{\mathrm{cv.out}}$ ($1 \le i \le c$) we have

$$\begin{pmatrix} h_1^{\mathrm{cv.out}} \\ \vdots \\ h_c^{\mathrm{cv.out}} \end{pmatrix} = \mathcal{B}(f_{\mathrm{out}}^{(1)}, \ldots, f_{\mathrm{out}}^{(t)}).$$

where $\mathcal{B}$ is a $(c \cdot n \times t \cdot n)$ binary matrix, consisting of $(n \times n)$ blocks which are either zero or the identity matrix. This is illustrated in Figure 1. Note that we allow the possibility of a feedforward of the external inputs *around* the compression functions. We actually ignore this feature in the remainder of the paper, since we observe that incorporating a feedforward according to Figure 1 does not help prevent the attacks we consider in this paper.

We also recall the established fact [19,25] that

> "...applying any simple (in both directions) invertible transformation to the input and to the output of the hash round function yields a new hash round function with the same security as the original one. "

We accept that such invertible transformations may well be applied to the external inputs and outputs of h before the input layer and after the output layer. But since they can have no cryptanalytic effect we ignore them.

Finally, we emphasize that we have restricted ourselves to parallel constructions where we compute $f_{\text{out}}^{(i)}$ as a linear combination of the external inputs. This is a natural limitation that encompasses most previously established schemes and offers obvious performance benefits in hardware implementation. We also note that the structural observations of Joux [12], Dean [8], and Kelsey and Schneier [13], do not relate to the task of building a larger compression function from a layer of parallel compression functions, but only to the usual Merkle-Damgård iteration of the final compression function that results.

3 First Observations

Our model for combining compression functions is both natural and powerful. To illustrate we might consider some of the more prominent block cipher-based compression functions, and Appendix A shows how the compression function of MDC-2 fits our framework with parameters $c = 2$, $t = 2$, $k = 2$, and $m = 1$ (the two internal compression functions being Matyas-Meyer-Oseas constructions), while the schemes proposed by Nandi *et al.* [28] have (c, t, k, m) parameter sets $(2, 3, 2, 1)$ and $(2, 3, 3, 2)$. Other schemes with appropriate parameters are provided below.

Name		c	t	k	m	*Cryptanalysis*
MDC-2	[5]	2	2	2	1	[32]
PBGV	[33]	2	2	2	2	[19]
ABREAST-DM	[19]	2	2	3	1	-
PARALLEL-DM	[21]	2	2	2	2	[14]
Hirose family	[10]	2	2	3	1	-
Nandi *et al.* N_1	[28]	2	3	2	1	[15]
Nandi *et al.* N_2	[28]	2	3	3	2	[15]

Like other compression function-based work, we cover instances where the underlying block cipher has different block and key lengths. However, unlike many previous constructions, we consider using t internal compression functions to derive c blocks of output with $t \geq c$. This allows us to make a fundamental distinction between previous work and that presented in this paper.

We identify the size of the output chaining variable that is required, and hence the number of output blocks c. Then, by considering established attacks, we achieve bounds on t that give us the minimum number of compression functions required to achieve the desired security level. We achieve this by a suitable analysis of the *output layer*. Our goal is to derive schemes that offer an optimal level of security of 2^{nc} work effort for preimage attacks and $2^{\frac{nc}{2}}$ for collision attacks. This nicely complements the work of Knudsen and Preneel [16,17,18], where the security of potentially non-optimal constructions is analysed via consideration of the *input layer*.

First, we observe the following series of implications. Given a set of parameters (c, t, k, m) for some construction, we use $(c, t, k, m) \in S$ to denote that a construction with ideal collision resistance with these parameters exists and we use $(c,t,k,m) \notin S$ to denote the fact that no such scheme can exist for this parameter set.

Implications 1 *Given c, t, k, and m all ≥ 1, we have the following four sets of pairwise equivalent implications:*

$$
\begin{array}{ll}
(c,t,k,m) \notin S \Rightarrow (c,t,k,m+1) \notin S & (c,t,k,m+1) \in S \Rightarrow (c,t,k,m) \in S \\
(c,t,k,m) \in S \Rightarrow (c,t,k+1,m) \in S & (c,t,k+1,m) \notin S \Rightarrow (c,t,k,m) \notin S \\
(c,t,k,m) \in S \Rightarrow (c,t+1,k,m) \in S & (c,t+1,k,m) \notin S \Rightarrow (c,t,k,m) \notin S \\
(c,t,k,m) \notin S \Rightarrow (c+1,t,k,m) \notin S & (c+1,t,k,m) \in S \Rightarrow (c,t,k,m) \in S.
\end{array}
$$

Justification: Suppose that there exists a secure design with parameter set (c,t,k,m). If we replace one message block by a constant then we still have a secure scheme. Thus the first implications are true. If we can use one additional input for every inner compression function, then we can use them so that none has any influence over the output. Thus, the second set of implications are true. If we have an additional compression function, we can still build a secure scheme by simply ignoring it. Thus the third set of implications is true. The final implications reflect the natural conjecture that constructing an ideal compression function of output size $c+1$ blocks is harder than constructing an ideal compression function of output size c blocks. □

The above implications are simple but useful. For MDC-2 the corresponding parameter set is $(2,2,2,1)$; a double block-length construction using two compression functions, each taking two equal-sized inputs (key and message) and processing one message block at each iteration. As shown in Section 4, $(2,2,2,1) \notin S$. Yet, there has been much effort in building schemes with a better rate, *i.e.* hashing more than one message at each iteration, for which one corresponding parameter set would be $(2,2,2,2)$. But we have that $(2,2,2,1) \notin S \Rightarrow (2,2,2,2) \notin S$ and such efforts cannot succeed[3].

4 Generic Attacks

In this section we consider two attacks that have been used in the literature. By generalising these attacks we are able to make statements about the impossibility of certain constructions. More importantly, we extract criteria for the successful design of a compression function with an intended level of security.

4.1 Attack method: DF

The first generic attack depends on what we term the number of *degrees of freedom*. It resembles the classic divide-and-conquer strategy from other cryptanalytic fields and can be applied to many proposals. The idea is to isolate, and

[3] To avoid any confusion we emphasize that the double block-length construction of Hirose [10] has parameter set $(2,3,2,1)$ since it uses a block cipher with a key that is twice the block size.

attack, a linear combination of the output blocks but to keep at least one external input block free from conditions. Then, the free input can be determined separately at the end of the attack. Attacks on MDC-2 provide a good example [32] and an equivalent representation of MDC-2 is provided in Figure 3. To find a preimage, one can attack the two branches independently. Finding a preimage for one branch will fix two inputs to the overall compression function and since we have three external inputs M, H_1, and H_2 there remains one external input free, *i.e.* one degree of freedom. Thus, we can independently use the free input to obtain a preimage for the other branch by brute force. The attack has work effort proportional to 2^n operations instead of the intended 2^{2n}. A collision attack works in a corresponding way. Consideration of this attack gives some of the bounds in [16,17,18]. We use it again here.

4.2 Attack method: MUL

The second attack uses *multi-collisions* and *multi-preimages* and is described in [36,15]. Similar considerations were used in a different way in [18]. For the attack to be successful, the compression function must satisfy several structural conditions. First, the attacker identifies a linear combination Z of the external outputs of h that depends on a non-empty set G_Z of compression functions $\{f^{(i)}\}$. Next, the attacker identifies two external input blocks X and Y. The external input X should influence the internal inputs to a subset G_X of the compression functions in G_Z. Similarly the external input block Y should influence the internal inputs to a subset, G_Y, of the compression functions in G_Z. It is important to identify X and Y (and hence G_X and G_Y) so that $G_X \cap G_Y = \emptyset$.

We now describe the attack in terms of finding preimages. The attacker fixes values to all the external input blocks except the previously identified inputs X and Y. Then, each value of X (resp. Y) is used to generate an internal output value for each $f^{(i)}$ in G_X (resp. G_Y). Thus, the attacker effectively compiles two lists L_X and L_Y each containing 2^n elements where, for every possible value of X and Y, all the internal outputs of the set of $\{f^{(i)}\}$ in G_X and G_Y are stored. Using Wagner's technique [38] these two lists can be joined in 2^n operations to obtain a third list L_Z that contains all (X, Y) (with $X \in L_X$ and $Y \in L_Y$) yielding the target image for the external output block. Since L_X and L_Y both have almost 2^n elements, we expect L_Z to contain almost 2^n elements.

At this stage we have found 2^n preimages to one external output block at a cost proportional to 2^n operations. If h has c output blocks, then an entry in the list L_z will give a good preimage for *all* c external output blocks with a probability of $2^{-(c-1)n}$. Thus, we repeat this procedure for $2^{(c-2)n}$ allocations of the $m + c - 2$ input variables distinct from x and y in order to find a valid preimage with a probability close to 1. The attack requires $2^{(c-1)\cdot n}$ operations instead of $2^{c \cdot n}$ in the ideal case. The collision attack works in a similar fashion.

5 Security Criteria

The compression function h that we wish to build takes $m + c$ external input blocks and each internal compression function $f^{(i)}$ takes k internal input blocks, defined by input matrices $\mathcal{A}_i$. Since we can apply any invertible transformation to the inputs of h, the important criteria for the input layer is the dimension of the vector space generated by columns of the matrices $\mathcal{A}_i$. This is already explored in existing work [14]. Considering the results in Section 4, we can make the following observations.

- To prevent attack DF, every external output block $h_i^{\text{cv.out}}$ must depend on all external input blocks $h_1^{\text{m.in}}, \ldots, h_m^{\text{m.in}}, h_1^{\text{cv.in}}, \ldots, h_c^{\text{cv.in}}$ no matter which invertible transformations of the external inputs and outputs are used.
- We say that an *identified pair* of external input blocks is a pair (A, B) where A and B both appear within the internal inputs to some $f^{(i)}$. (For example, with $f^{(i)}(A, B \oplus C)$, the identified pairs (A, B), (A, C), and (B, C) appear in $f^{(i)}$.) Then, in order to prevent attack MUL, *every possible pair* of external input blocks must appear as an identified pair for every invertible combination of external output blocks $h_i^{\text{cv.out}}$. This applies, no matter which invertible transformations of the external inputs and outputs are used.

We now consider the secure combination of independent compression functions.

5.1 Deriving valid parameter sets

Rather than using the identified attacks and their generalisations to break specific proposals, we use them to derive general lower bounds on the number of smaller ideal compression functions needed to derive a larger ideal compression function. More precisely, for a set of k-input secure compression functions, *i.e.* compressing kn to n bits, we ascertain the minimum number $t_{\min}$ of compression functions required to build a secure compression function producing cn bits, since they must resist DF and MUL attacks. To do this, we adopt a two-phase approach. First we establish a bound d on the number of compression functions we require when considering any single linear combination of the c output blocks. We then derive a bound $t_{\min}$ on the minimum number of compression functions that are required when simultaneously considering all c output blocks in the chaining variable (see Table 1).

Initial bounds on *d*. First, we consider attack DF and we observe that since each compression function takes k input blocks, and that there are $m+c$ external input blocks to h, then we must have at least $\lceil \frac{m+c}{k} \rceil$ compression functions. Thus, every external output block depends on at least $\lceil \frac{m+c}{k} \rceil$ internal output blocks. This is required for every linear combination of the external outputs and so we have $d \geq \lceil \frac{m+c}{k} \rceil$.

Table 1. The minimum number $t_{\min}$ of compression functions required to resist DF and MUL attacks, for parameter set $(c, t_{\min}, k, m)$.

Parameters			*Basic Bounds*		*Improved*	
c	k	m	d	$t_{\min}$	d	$t_{\min}$
2	2	1	2	3	3	5
2	2	2	2	3	3	5
2	3	1	1	2	-	-
2	3	2	2	3	3	5
3	2	1	2	4	3	6
3	2	2	3	6	4	7
3	3	1	2	4	3	6
3	3	2	2	4	3	6
4	2	1	3	7	4	8
4	2	2	3	7	4	8
4	3	1	2	5	3	7
4	3	2	2	5	3	7

Improved bounds on *d*. By considering attack DF we can derive the basic bounds on d given above. However a generic analysis allows us to improve on this bound by ensuring that a proposed configuration of compression functions also resists attack MUL. While the style of analysis is generic and can be reused for different parameter sets, it is most easily described by reference to one particular instance.

Suppose that we consider the parameter set given by $m + c = 3$ and $k = 2$ with A, B, and C denoting the three n-bit inputs to the compression function. Our basic bound gives $d \geq 2$, so here we assume that $d = 2$. Suppose that an external output block $h_i^{\mathrm{cv.out}}$, or more generally a linear combination Z of one or more output blocks, is bound to only two compression functions f_1 and f_2. Then we have that $Z = f_1(X_1, X_2) \oplus f_2(X_3, X_4)$ where X_1, X_2, X_3, and X_4 are linear combinations of A, B, and C.

The rank of the vector space $\langle X_1, X_2, X_3, X_4 \rangle$ spanned by $X_1, \ldots, X_4$ must be equal to three since otherwise attack DF would apply. Therefore, one can extract from $\langle X_1, X_2, X_3, X_4 \rangle$ three elements which together form a basis of $\langle A, B, C \rangle$. Without loss of generality, we assume that $\langle X_1, X_2, X_3 \rangle = \langle A, B, C \rangle$ and there exist binary coefficients α_i so that $X_4 = \alpha_1 A \oplus \alpha_2 B \oplus \alpha_3 C$. We cannot have α_1 or α_2 equal to zero, since otherwise the pairs (A, C) and (B, C) would not be encountered in either f_1 or f_2 and the attack MUL would apply. So we can assume without loss of generality, that $\alpha_1 = 1$ and $\alpha_2 = 1$. If we now apply the invertible change of variables $A' = A \oplus B$, $B' = B$, and $C' = C$, Z can be rewritten as $Z = f_1(A' \oplus B', B') \oplus f_2(C', A' \oplus \alpha_3 C')$. Since (B', C') is not encountered in either f_1 or f_2, then the attack MUL applies. Thus $d \geq 3$. Note that such reasoning also applies when $m + c \geq 3$, thus if $m + c \geq 3$ and $k = 2$ we have $d \geq 3$.

This style of reasoning allows us to improve most of the bounds on d by considering the applicability of the second generic attack MUL. The sole exception is the parameter set $c = 2$, $k = 3$, and $m = 1$ which corresponds to the provably secure scheme of Hirose and will be discused in Section 6.2.

Initial bounds on t. We now turn bounds on d into bounds on the minimum number of compression functions that must be used, $t_{\min}$. While any linear combination of the c external outputs must depend on at least d inner compression functions, a bound on the minimal number $t_{\min}$ of compression functions is not immediate. Here we derive a value for t independently of the analysis needed to derive d.

In the simple case that $c = 2$ a combinatorial style of reasoning can be used and this shows that $t_{\min} \geq \frac{3d}{2}$ if d is even and $t_{\min} \geq \frac{3(d-1)}{2} + 2$ otherwise. However a more flexible approach, scaling better to larger parameters, uses an analogy with coding theory.

Consider vectors of t elements (corresponding to the number of internal compression functions) and attach to each external output block $h_i^{\text{c.out}}$ a vector v_i whose value is determined by whether an internal compression function influences $h_i^{\text{cv.out}}$. If compression function $f^{(j)}$ is active in $h_i^{\text{cv.out}}$ then set the j^{th} entry of v_i to 1, otherwise it has the value 0. For example, if $t = 3$ and for some proposed construction only $f^{(1)}$ and $f^{(3)}$ are involved in $h_i^{\text{cv.out}}$, then we set $v_i = (1, 0, 1)$.

In turning our result on d into a constraint on $t_{\min}$, we consider the problem of looking for a binary code of length t with minimal distance d and dimension c. The Singleton bound yields $c \leq t - d + 1$ and so $t \geq c + d - 1$. The Hamming bound is tighter, but is more involved and given in Appendix B.

Improved bounds on t. It is interesting to note that configurations with particular features might allow a dedicated, and potentially tighter, analysis for the bounds on t. An example is given in Appendix C. However since such analysis does not apply to the general model we have established, (it relies on a particular form to the input layer), we do not use it in the derivation of the bounds in Table 1.

6 Constructions

Given a set of parameters (c, t, k, m) it is easy to use the newly established bounds to check whether, according to our criteria, the scheme is necessarily insecure. Turning this around, if one wants to build a scheme with some predefined c, k, and m then one can compute a lower bound $t_{\min}$ on the number of internal compression functions that must be used, in a parallel configuration that we consider in Figure 1.

6.1 Impossible constructions

Using the bounds established in Section 5 we first consider interesting parameter sets such as $c \in \{2,3,4\}$, $k \in \{2,3\}$, and $m \in \{1,2\}$. These correspond to cases where we aim to obtain double, triple, or quadruple block-length constructions, using a block cipher with key size the same or twice the block size, and processing either one or two blocks of message.

We use the bounds on d and once c, k, and m are chosen we search for the smallest t that satisfy our bounds. We thus derive an integer t_{min} for the minimum number of independent compression functions that must be used in the specified construction. Note that a given t_{min} does not mean that secure schemes with t_{min} inner compression functions necessarily exist. Rather, no secure scheme can exist with fewer independent compression functions of the stated type.

Immediately there are interesting results and we note that secure schemes with (c,t,k,m) parameters $(2,3,2,1)$ or $(2,3,3,2)$ are impossible. These correspond to the schemes of Nandi *et al.* [28]. Since our bounds are derived by generalising attacks on [28] we expect this to be the case. However, constructions using four inner compression functions, would still be insecure.

Indeed, for the most natural case with $c = 2$, $k = 2$, and $m = 1$, the case of DES and AES-128, one must use at least five inner compression functions in a parallel framework to obtain a secure hash function offering 64-bit and 128-bit security respectively. This is more than one might have expected. The case of a quadruple block-length output is even more dramatic. If one wished to design a compression function that used AES-128 as a building block but offered 256-bit security, then one would be required to use at least eight parallel instantiations of AES-128 to produce a secure compression function.

6.2 Proposed constructions

Figure 2 shows a $(2,5,2,1)$-scheme that is secure against the attacks considered in this paper. Further research will determine whether other attacks apply. However, this scheme is one from a range of double block-length hash function constructions that might be instantiated with AES-128 or other block ciphers with identical block and key sizes. Note that this is the only such construction that remains uncompromised. Figure 2 also depicts a $(2,5,3,2)$-construction that resists our generic attacks, meets our bound, and could be instantiated with AES-256 or a cipher like IDEA (or even TWO-KEY TRIPLE-DES) with a key length twice the block length. The parameter set $(2,2,3,1)$ is covered by Hirose.

A particularly simple set of parameters satisfies $k \geq m + c$ when all external inputs can be accommodated within each internal compression function and $d = 1$. Thus, we derive a secure compression function with $t = c$ without requiring additional internal compression functions. We only need to ensure that all external input blocks are used directly in every internal compression function with any free internal inputs fixed to a constant value. Then every external output needs to be bound to one, and only one, internal compression function. Hirose [10] has already studied members of this family of block cipher based

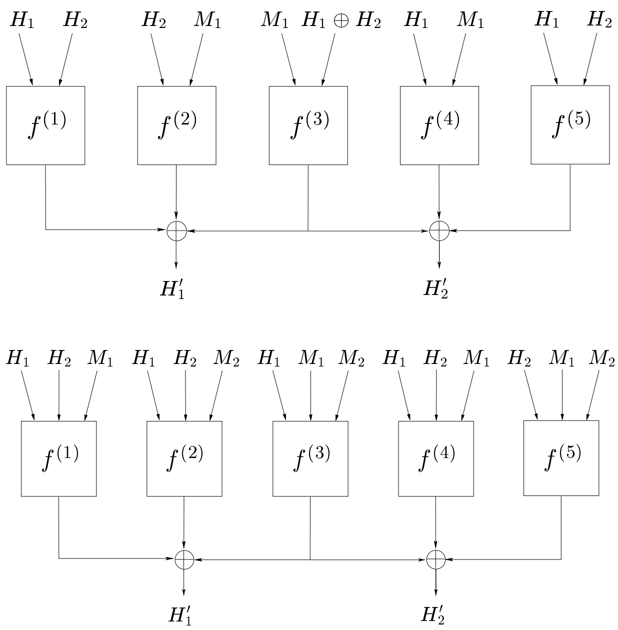

Fig. 2. A $(2, 5, 2, 1)$ and a $(2, 5, 3, 2)$ construction. For the first cosntruction each (independent) inner compression function can be instantiated using a block cipher with equal key and block size. For the second construction, the key size is double the block size. M_1, M_2 are n-bit message blocks; H_1, H_2 are n-bit incoming chaining variable blocks and H_1', H_2' are n-bit output chaining variable blocks.

hash functions and proved their security in both the random oracle model and in the ideal cipher model when the compression functions are instantiated using a Davies-Meyer construction.

7 Conclusions

In this paper we have analyzed techniques to construct a larger compression function by combining smaller, trusted, compression functions. By generalising attacks in the literature, we are able to establish conditions on the type and number of components that are required to ensure that the constructions are not vulnerable to a range of powerful and general attacks.

This work has a direct and immediate application to the construction of block cipher-based hash functions for which the length of the hash output is greater than the block size of the underlying block cipher. The most important conclusion to draw is that it is actually rather difficult to use multiple instantiations of a block cipher to build a secure compression function; or at least to do so in a particularly efficient way. For example, when using AES-128 for double block-length hashing, one must use at least five parallel instantiations of AES-128 to derive a compression function offering 128-bit security respectively. To achieve 256-bit security, one must use eight. This is a surprisingly high number of block cipher calls, particularly so when we consider that this is merely to avoid the application of generic attacks.

While there are many possible generalisations to the framework used in this paper, we have provided a natural and broad framework for the analysis of schemes of this type. Extensions to this work, including identifying schemes that achieve the most efficient permissible bounds, is the subject of ongoing research.

Acknowledgements

The authors would like to thank Sébastien Kunz-Jacques, Yannick Seurin, and the progam committee for their valuable comments.

References

1. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62-73. 1993.
2. J. Black, P. Rogaway, and T. Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer-Verlag, 2002.
3. J. Black, M. Cochran, and T. Shrimpton. On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT '05*, volume 3494 of *Lecture Notes in Computer Science*, pages 526-541. Springer-Verlag, 2005.
4. L. Brown, J. Pieprzyk, and J. Seberry. LOKI - a Cryptographic Primitive for Authentication and Secrecy Applications. In J. Pieprzyk and J. Seberry, editors, *Advances in Cryptology – AUSCRYPT '90*, volume 453 of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 1990.
5. D. Coppersmith, S. Pilpel, C.H. Meyer, S.M. Matyas, M.M. Hyden, J. Oseas, B. Brachtl, and M. Schilling. Data Authentication Using Modification Dectection Codes Based on a Public One Way Encryption Function. U.S. Patent No. 4,908,861, March 13, 1990.
6. J-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430-448. Springer-Verlag, 2005.
7. I. Damgård. A Design Principle for Hash Functions. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1989.
8. R.D. Dean. *Formal Aspects of Mobile Code Security*. PhD thesis, Princeton University, 1999.
9. H. Handschuh, L.R. Knudsen, and M.J.B. Robshaw. Analysis of SHA-1 in Encryption Mode. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 70-83. Springer-Verlag, 2001.

10. S. Hirose. Provably Secure Double-block-length Hash Functions in a Black-box Model. In C. Park and S. Chee, editors, *Information Security and Cryptology – ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342. Springer-Verlag, 2004.
11. S. Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In M.J.B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science.*
12. A. Joux. Multi-collisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer-Verlag, 2004.
13. J. Kelsey and B. Schneier. Second Preimages on n-bit Hash Functions for Much Less Than 2^n Work. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490. Springer-Verlag, 2005.
14. L.R. Knudsen and X. Lai. New Attacks on All Double Block Length Hash Functions of Hash Rate 1, Including the Parallel-DM. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 410–418. Springer-Verlag, 1994.
15. L.R. Knudsen and F. Muller. Some Attacks Against a Double Length Hash Proposal. In B. Roy, editor, *Advances in Cryptology – ASIACRYPT '05*, volume 3788 of *Lecture Notes in Computer Science*, pages 462–473. Springer-Verlag, 2005.
16. L.R. Knudsen and B. Preneel. Hash Functions Based on Block Ciphers and Quaternary Codes. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology – ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 77–90. Springer-Verlag, 1996.
17. L.R. Knudsen and B. Preneel. Fast and Secure Hashing Based on Codes. In B.S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 485–498. Springer-Verlag, 1997.
18. L.R. Knudsen and B. Preneel. Construction of Secure and Fast Hash Functions Using Nonbinary Error-Correcting Codes. *IEEE Transactions on Information Theory*, 48(9):2524–2539, 2002.
19. X. Lai and J.L. Massey. Hash Functions Based on Block Ciphers. In R. A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer-Verlag, 1992.
20. X. Lai, J.L. Massey, and S. Murphy. Markov Ciphers and Differential Cryptanalysis. In D.W. Davies, editor, *Advances in Cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer-Verlag, 1991.
21. X. Lai, C. Waldvogel, W. Hohl, and T. Meier. Security of Iterated Hash Functions Based on Block Ciphers. In D.R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 379–390. Springer-Verlag, 1993.
22. S. Lucks. A Failure-Friendly Design Principle for Hash Functions. In B. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 474-494. Springer-Verlag, 2005.
23. M. Liskov, R.L. Rivest, and D. Wagner. Tweakable Block Ciphers. In M. Yung, editor, *Advances in Cryptology – CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 31-46. Springer-Verlag, 2002.
24. R. Matsumoto, K. Kurosawa, and T. Itoh. Primal-Dual Distance Bounds of Linear Codes with Application to Cryptography. IACR Cryptology ePrint Archive, Report 2005/194. Available from: `http://eprint.iacr.org`.

25. W. Meier and O. Staffelbach. Nonlinearity Criteria for Cryptographic Functions. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 549–562. Springer-Verlag, 1989.
26. A.J. Menezes, S.A. Vanstone, and P.C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.
27. R.C. Merkle. One Way Hash Functions and DES. In G. Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1989.
28. M. Nandi, W. Lee, K. Sakurai, and S. Lee. Security Analysis of a 2/3-rate Double Length Compression Function in Black-box Model. In H. Gilbert and H. Handschuh, editors, *Fast Software Encryption – FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 243–254. Springer-Verlag, 2005.
29. National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, November 2001 . Available from: `http://csrc.nist.gov`.
30. National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard, August 2002 . Available from: `http://csrc.nist.gov`.
31. National Insitute of Standards and Technology. SP800-67: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, May 2004 . Available from: `http://csrc.nist.gov`.
32. B. Preneel. *Analysis and Design of Cryptographic Hash Functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.
33. B. Preneel, A. Bosselaers, R. Govaerts, and J. Vandewalle. Collision-free Hash Functions Based on Block Cipher Algorithms. In *Proceedings 1989 International Carnahan Conference on Security Technology (Oct 3–5 1989: Zurich, Switzerland)*, pages 203–210. IEEE, 1989. IEEE catalog number 89CH2774-8.
34. B. Preneel, R. Govaerts, and J. Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In D.R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer-Verlag, 1993.
35. J.-J. Quisquater and M. Girault. 2n-bit Hash-functions Using n-bit Symmetric Block Cipher Algorithms. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – EUROCRYPT '89*, volume 434 of *Lecture Notes in Computer Science*, pages 102–109. Springer-Verlag, 1989.
36. B. Preneel, R. Govaerts, and J. Vandewalle. On the Power of Memory in the Design of Collision Resistant Hash Functions. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology – ASIACRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 105-121. Springer-Verlag, 1992.
37. Ronald L. Rivest. RFC 1321: The MD5 Message-Digest Algorithm, April 1992 . Available from: `http://www.ietf.org/rfc/rfc1321.txt`.
38. D. Wagner. A Generalized Birthday Problem. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer-Verlag, 2002.
39. X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer-Verlag, 2005.
40. X. Wang, Y.L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer-Verlag, 2005.

Appendix A: Some Established Constructions

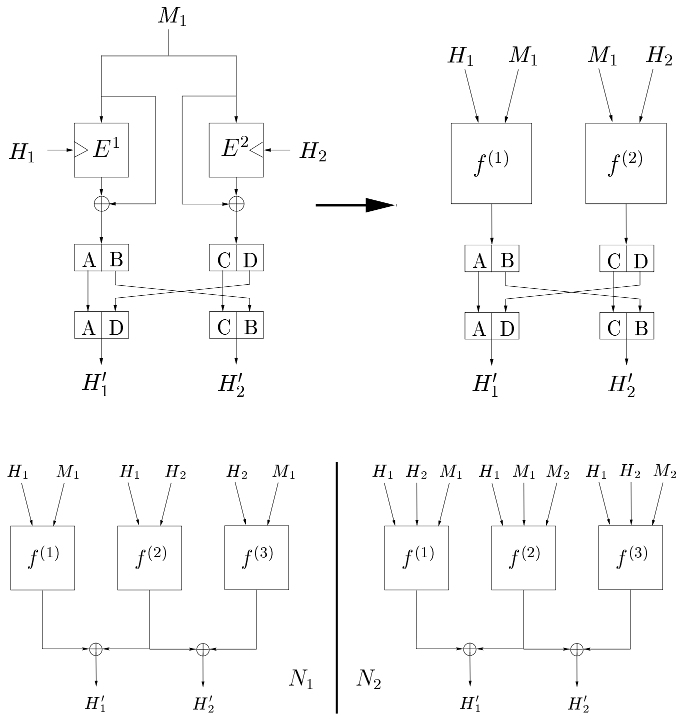

Fig. 3. Mapping the compression functions of MDC-2 and Nandi *et al.* to our framework. Recall that simple invertible transformations such as a swap can be ignored [19]. M_1, M_2 are n-bit message blocks; H_1, H_2 are n-bit incoming chaining variable blocks and H'_1, H'_2 are n-bit output chaining variable blocks.

Appendix B: The Hamming Bound

While it is more difficult to exploit, the Hamming bound is tighter than the Singleton bound. Here we give an improved version of the Hamming bound [24]:

$$\begin{cases} c \leq t - \log_2 \left(\sum_{i=0}^{\frac{d-1}{2}} \binom{t}{i} \right) & \text{if } d \text{ is odd, and} \\ c \leq t - \log_2 \left(\binom{t-1}{\frac{d}{2}-1} + \sum_{i=0}^{\frac{d}{2}-1} \binom{t}{i} \right) & \text{if } d \text{ is even.} \end{cases}$$

This can be used to give a bound on t in terms of c and d. The table below allows us to compare the Singleton and the Hamming bound for some parameter sets used in Table 1.

Parameters		*Bounds*	
c	d	Singleton	Hamming
2	1	2	2
2	2	3	3
2	3	4	5
3	2	4	4
3	3	5	6
3	4	6	7
4	2	5	5
4	3	6	7
4	4	7	8

Appendix C: Preferred bounds on t in a restricted model

While the bounds derived in this appendix do not apply to the general model, it is interesting to see what can be achieved with some minor restrictions to the general framework. Here we consider the impact of a simplified input layer and we assume that each of the kt internal inputs is one of the $m+c$ external inputs. This is far more restrictive than the general case of a linear combination of the external inputs and so it is not surprising that we can derive better bounds.

From the previous analysis we know that every possible pair of external inputs must be present in at least one of the internal compression functions involved in any linear combination of the external output blocks. We have $N_C = (m+c)\cdot(m+c-1)/2$ different pairs. In each internal compression function, we can have at most $N_K = k\cdot(k-1)/2$ pairs present. Each of the N_c pairs must appear in at least c different internal compression functions since otherwise there would exist a linear combination of the external outputs which would involve none of these internal compression functions and attack MUL would apply. We thus have:

$$t \geq \frac{c\cdot(m+c)\cdot(m+c-1)}{k\cdot(k-1)}.$$

This reasoning can also be applied to attack DF since we have at most $m+c$ different vectors as input to the internal compression functions. Each external input block must appear in at least c different internal compression functions, otherwise some linear combinations of the external outputs would not depend on this external input block. We can put k blocks in one inner function and thus we have:

$$t \geq \frac{c\cdot(m+c)}{k}.$$

These bounds are often much better than the general case and illustate the importance of the input layer. A weak input layer can dramatically increase the minimum number of compression functions required for a secure construction.