On the (In)Security of IDEA in Various Hashing Modes*

Lei Wei[1], Thomas Peyrin[1], Przemysław Sokołowski[2], San Ling[1], Josef Pieprzyk[2], and Huaxiong Wang[1]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
`wl@pmail.ntu.edu.sg`, `thomas.peyrin@gmail.com`
[2] Macquarie University, Australia

Abstract. In this article, we study the security of the `IDEA` block cipher when it is used in various simple-length or double-length hashing modes. Even though this cipher is still considered as secure, we show that one should avoid its use as internal primitive for block cipher based hashing. In particular, we are able to generate instantaneously free-start collisions for most modes, and even semi-free-start collisions, pseudo-preimages or hash collisions in practical complexity. This work shows a practical example of the gap that exists between secret-key and known or chosen-key security for block ciphers. Moreover, we also settle the 20-year-old standing open question concerning the security of the Abreast-DM and Tandem-DM double-length compression functions, originally invented to be instantiated with `IDEA`. Our attacks have been verified experimentally and work even for strengthened versions of `IDEA` with any number of rounds.

Key words: `IDEA`, block cipher, hash function, cryptanalysis, collision, preimage

1 Introduction

Hash functions are considered as a very important building block for many security and cryptography applications. Informally, a hash function H is a function that takes an arbitrarily long message as input and outputs a fixed-length hash value of size n bits. In cryptography, we want these functions to fulfill three security requirements, namely collision resistance

* The first, fourth and sixth authors are supported by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03 and the first author is also supported by the Singapore Ministry of Education under Research Grant T206B2204 and by the NTU NAP Startup Grant M58110000. The second author is supported by the Lee Kuan Yew Postdoctoral Fellowship 2011 and the Singapore National Research Foundation Fellowship 2012.

and (second)-preimage resistance. It should be impossible for an adversary to find a collision (two different messages that lead to the same hash value) in less than $2^{n/2}$ hash computations, or a (second)-preimage (a message hashing to a given challenge) in less than 2^n hash computations. Most of nowadays hash functions divide the whole input message into blocks after padding it, and then process the blocks in an iterative way. A very known and utilised example is the Merkle-Damgaård algorithm [12, 33], which uses an n-bit compression function h in order to process the m message blocks M_i: $CV_{i+1} = h(CV_i, M_i)$, where CV_i is the n-bit internal state (or chaining variable) that is initialized by a fixed public value $CV_0 = IV$ and the final hash value is H_m. This algorithm is very interesting because it allows to reduce the collision/preimage security of the hash function to the collision/preimage security of the compression function. However, in order to guarantee the soundness of the construction, a designer must ensure that an attacker can not break the collision/preimage resistance of the compression function. One can identify different security properties for a compression function:

- *free-start collision*: in less than $2^{n/2}$ computations, find two different pairs $(CV, M) \neq (CV', M')$ such that they lead to the same compression function output value: $h(CV, M) = h(CV', M')$,
- *semi-free-start collision*: in less than $2^{n/2}$ computations, find one chaining variable CV and two different message blocks $M \neq M'$ such that they lead to the same compression function output value: $h(CV, M) = h(CV, M')$,
- *preimage*: in less than 2^n computations, find one chaining variable CV and one message block M such that they lead to a given output challenge X: $h(CV, M) = X$.

Note that a semi-free-start collision for the compression function where the chaining variable CV is not chosen by the attacker directly leads to a collision for the whole hash function. In any case, a semi-free-start collision is very dangerous since it means that for some choices of IV, the attacker knows how to generate a collision. Even free-start collision are considered serious as they invalidate the collision resistance assumption on the compression function and we have seen many free-start collision attacks eventually turning into full hash collision attacks in the recent history (for example free-start collision attacks for `MD5` were quickly identified [14], then upgraded to semi-free-start collision attacks [15] and eventually to full collision attacks [38]). As for preimage attacks on the compression function (also known as pseudo-preimages), they are very relevant since there exist a meet-in-the-middle algorithm that in most cases can turn them into a preimage attack for the full hash function.

The separation between a block cipher and a compression function has always been blurry. Constructions are known to turn the former into the latter [7, 36] or the latter into the former [31]. For example, the Davies-Meyer mode [1] converts a secure block cipher E into a secure compression function and is incorporated in a large majority of the currently known hash functions. While very satisfying solutions exist to transform a secure n-bit block cipher into an n-bit compression function (Davies-Meyer, Miyaguchi-Preneel, Matyas-Meyer-Oseas modes [1] or see [7, 36] for a systematic study of this problem), there is still a lot of research being actively conducted on double-block length compression functions (where the block cipher size is n bits and the compression function output size is $2n$), from simple-key block ciphers such as AES-128 or double-key such as AES-256 [11].

A major difference between the cryptanalysis of block ciphers and compression functions is that the attacker can fully control the inner behavior of the compression function. In other words, the attacker can use more efficiently the freedom degrees available on the input (i.e. the number of independent binary variables he has to determine). A new security model for block ciphers, the so-called *known-key model* [24], was recently proposed in order to fill the gap between these two situations. In this model, the secret key is known to the adversary and its goal is to distinguish the behavior of a random instance of the block cipher from the one of a random permutation by constructing a set of (plaintext, ciphertext) pairs satisfying an *evasive* property. Such a property is easy to check but impossible to achieve with the same complexity and a non-negligible probability using oracle accesses to a random permutation and its inverse. In general, these known-key attacks are not regarded as problematic when the block cipher is used in a classical "secret key" setting. Moreover, it is rare that such threats are extended to attacks on the compression function.

A potential candidate for hashing is the 64-bit block cipher IDEA [26, 39] that uses 128-bit keys. While a simple-length hashing mode would only provide a 64-bit hash output, insufficient for most of nowadays security applications, a double-block length construction (DBL) would allow 128-bit hash outputs which can be sufficient in some scenarios. As IDEA handles double-length keys, more freedom in the constructions is possible. In fact, the well known Abreast-DM and Tandem-DM modes were specifically created to perform hashing with IDEA (see page 2 and Section

6 of [39]). These modes were later studied in much details [16, 17, 28, 30], but the security they provide when instantiated with IDEA remains a 20-year-old standing open question. In classical "secret key" setting, IDEA has already been studied a lot [2–6, 9, 10, 13, 18] and is still considered as a secure cipher despite its age and despite the current best attack [5] that requires 2^{63} data (half the codebook) and 2^{114} computations to recover the secret key for IDEA reduced to 7.5 rounds over a total of 8.5 (the attack on the full cipher from [5] is very marginal with $2^{126.8}$ computations and the one from [22] requires 2^{126} computations and 2^{52} chosen plaintexts). One can also cite the work of [6], that exposes a weak-key class of size 2^{64}. Note also that a first step towards analysis of IDEA in hashing mode was done in [21] where a 3-round chosen-key attack is described and in [9] where the authors show how to find a free-start near collision (only a subset of the output collides) when IDEA is plugged into the Hirose DBL mode [9] (and also a free-start collision if the internal constant c is controlled by the attacker).

Our contribution. In this paper, we study the security of the IDEA block cipher [26, 39] when plugged into various block cipher based compression function constructions, such as the classical Davies-Meyer mode [1], also DBL constructions such as Hirose [19, 20], Abreast-DM and Tandem-DM [27, 39], Peyrin *et al.* (II) [35] or MJH-Double [29]. Even if this cipher is still considered as secure in the classical "secret key" setting, its security remains an open problem in hashing mode. Depending on the IDEA-based hash construction, we show that an attacker can find free-start collisions instantaneously, preimages or semi-free-start collisions practically. For some modes, we even describe a method to compute collisions for the whole hash function. These attacks are based on weak-keys utilisation, but in contrary to the "secret key" setting where the goal of the attacker is to exhibit the biggest weak-key class possible, in hashing mode the goal is to find and exploit the weakest of all keys. We use the fact that the key 0 in IDEA is extremely weak, actually rendering the whole encryption process a T-function [23], already known as dangerous for building a hash function [34]. While weak-keys are already known to be dangerous for block cipher-based hash functions, our method use a novel and non-trivial almost half-involution property for IDEA. Even strengthened versions of the cipher with any number of rounds can be attacked with about the same complexities. This work is one more example that one has to be very careful when hashing with a block cipher that presents any weakness when the key is known or controlled by the attacker. In particular, one should strictly avoid the use of a block cipher for which weak-keys exist, even if only a single weak-key is known.

2 The `IDEA` block cipher

The International Data Encryption Algorithm (`IDEA`) is a 64-bit block cipher handling 128-bit keys and designed by Lai and Massey [26, 39] in 1990. While its use is reducing over the recent years, it remains deployed in practice and has not been broken yet despite its advanced age. It has a very simple design, performing 8.5 rounds composed of only 16-bit wide XOR, additions and multiplications. More precisely, one round is composed of three layers: first the key addition layer (denoted KA), a multiplication-addition layer (denoted MA) and a middle words switching layer (denoted S). For the eighth round, the switching is omitted.

Let X^i represent the 64-bit internal state of `IDEA` before application of the i-th round and we can view it as four 16-bit subwords $X^i = (X^i_1, X^i_2, X^i_3, X^i_4)$, with $1 \leq i \leq 9$. Also, $Y^i = (Y^i_1, Y^i_2, Y^i_3, Y^i_4)$ will stand for the intermediate internal state value of `IDEA` during the i-th round, right between the KA and the MA layers. We denote by $\oplus$ the bitwise XOR operation, by $\boxplus$ the addition modulo 2^{16} and by $\odot$ the multiplication modulo $2^{16}+1$, where the value 0 is considered as 2^{16} and vice-versa. Finally, $Z^i = (Z^i_1, Z^i_2, Z^i_3, Z^i_4, Z^i_5, Z^i_6)$ represents the six 16-bit subkeys used during the i-th round (only the first four subkeys for the last half round).

The KA layer simply incorporates four subkeys:

$$Y^i_1 = X^i_1 \odot Z^i_1, \qquad Y^i_2 = X^i_2 \boxplus Z^i_2, \qquad Y^i_3 = X^i_3 \boxplus Z^i_3, \qquad Y^i_4 = X^i_4 \odot Z^i_4.$$

The MA layer first computes $B = Z^i_6 \odot ((Y^i_2 \oplus Y^i_4) \boxplus (Z^i_5 \odot (Y^i_1 \oplus Y^i_3)))$ and $A = B \boxplus (Z^i_5 \odot (Y^i_1 \oplus Y^i_3))$. Then, after application of the S layer we have:

$$X^{i+1}_1 = Y^i_1 \oplus B, \qquad X^{i+1}_2 = Y^i_3 \oplus B, \qquad X^{i+1}_3 = Y^i_2 \oplus A, \qquad X^{i+1}_4 = Y^i_4 \oplus A.$$

All the subkeys are simply determined by choosing consecutive bits in the 128-bit master key according to the Table 2 given in Appendix A. Finally, ciphering the plaintext P with `IDEA` to obtain the ciphertext C is defined as: $C = \mathrm{KA} \circ \mathrm{S} \circ \{\mathrm{S} \circ \mathrm{MA} \circ \mathrm{KA}\}^8(P)$. Figure 1 provides a schematic view of one round of `IDEA`.

Currently, the best cryptanalysis work published on `IDEA` [5] can reach 7.5 rounds with 2^{63} data (half the codebook) and 2^{114} computations. Concerning weak-keys, the current biggest weak-key class contains 2^{64} elements and has been published in [6].

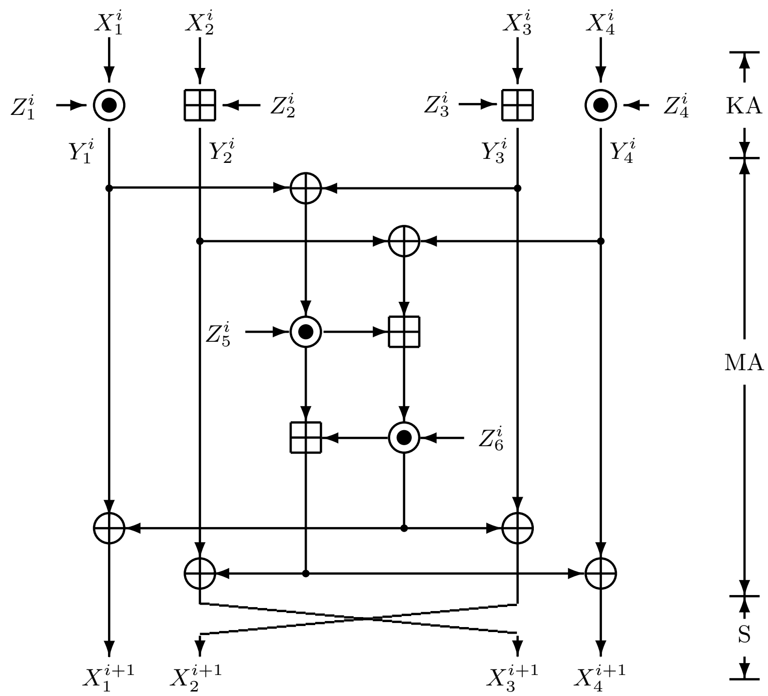

Fig. 1. One round of `IDEA`

3 Hashing with a double-length key block cipher

We will study the security of the various block cipher-based constructions that can use `IDEA` as the internal primitive. Therefore, we only consider the ones that use a double-key block cipher. More precisely, we denote $C = E_K(P)$ the process of ciphering the 64-bit plaintext P with `IDEA` using the 128-bit key K.

3.1 Simple-length compression function

A simple-length compression function construction with `IDEA` will provide a 64-bit output CV_{i+1}.

Davies-Meyer is the most usual simple-length mode [1] and it handles 128-bit message blocks: $CV_{i+1} = E_M(CV_i) \oplus CV_i$. Most standardized hash functions are actually implementing this mode, with an ad-hoc internal block cipher. While some weaknesses such as fixed-points are known, its security in terms of preimage and collision resistance have been studied and proved in the ideal cipher model [7]. Namely, we should expect at least 2^{32} and 2^{64} computations respectively to generate a (semi)-free-start collision or preimage for the compression function. Note that Miyaguchi-Preneel and Matyas-Meyer-Oseas simple-block length modes [1] are not considered in this article since they require the internal primitive to have the same block and key size, which is not the case for the `IDEA` block cipher.

3.2 Double-length compression function

A more interesting design strategy with `IDEA` would be to define double-block length constructions, in order to get 128-bit output, represented by two 64-bit words $CV1_i$ and $CV2_i$. This problem has already been studied a lot and remains a very active research domain, even when the internal primitive is a double-key block cipher.

Abreast-DM and Tandem-DM will of course be considered in this article since they both have been especially designed for `IDEA` [27, 39]. Tandem-DM handles a 64-bit message block M. We define $W = E_{CV1_i||M}(CV2_i)$ and then we have

$$\begin{aligned} CV1_{i+1} &= E_{M||W}(CV1_i) \oplus CV1_i, \\ CV2_{i+1} &= W \oplus CV2_i. \end{aligned}$$

Abreast-DM also handles a 64-bit message block M:

$$\begin{aligned} CV1_{i+1} &= E_{M||CV2_i}(\overline{CV1_i}) \oplus CV1_i, \\ CV2_{i+1} &= E_{CV1_i||M}(CV2_i) \oplus CV2_i, \end{aligned}$$

where $\overline{X}$ stands for the bitwise complement of X.

Hirose proposed a construction that contains two independent block cipher instances [19], later improved to only a single instance [20] by using a constant c to simulate the two independent ciphers:

$$\begin{aligned} CV1_{i+1} &= E_{CV2_i||M}(CV1_i) \oplus CV1_i, \\ CV2_{i+1} &= E_{CV2_i||M}(CV1_i \oplus c) \oplus CV1_i \oplus c. \end{aligned}$$

Peyrin *et al.* described in [35] a compression function (denoted Peyrin *et al.*(II)) that utilizes 5 calls to independent $3n$-to-n-bit compression functions, advising to be instantiated with double-key internal block ciphers such as `AES`-256 or `IDEA`. It handles two 64-bit message blocks $M1$ and $M2$:

$$CV1_{i+1} = f_1(CV1_i, CV2_i, M1) \oplus f_2(CV1_i, CV2_i, M2) \oplus f_3(CV1_i, M1, M2),$$
$$CV2_{i+1} = f_3(CV1_i, M1, M2) \oplus f_4(CV1_i, CV2_i, M1) \oplus f_5(CV2_i, M1, M2),$$

where the functions f_i can be build for example by using the `IDEA` block cipher into a Davies-Meyer mode and we can simulate their independency by XORing distinct constants to the plaintext inputs, as it is done in [20]: $f_i(U, V, W) = E_{U||V}(W \oplus i) \oplus W$ (note that XORing the constants on the key input would be avoided in practice because it would lead to very frequent rekeying and therefore reduce the overall performance of the hash function). Since no real candidate was proposed by the authors, all possible position permutations of the three f_i inputs will be considered. Note that when cryptanalysing this scheme, we will attack the functions f_i independently. Thus, we will not use any weakness coming from potential dependencies between the functions f_i (apart of course that all 5 functions are based on `IDEA`).

MJH-Double is a rate 1 double-block length compression function recently published by Lee and Stam [29]. It uses a double-key block cipher and handles two 64-bit message blocks $M1$ and $M2$:

$$CV1_{i+1} = E_{M2||CV2_i}(CV1_i \oplus M1) \oplus CV1_i \oplus M1,$$
$$CV2_{i+1} = g \cdot (E_{M2||CV2_i}(f(CV1_i \oplus M1)) \oplus f(CV1_i \oplus M1)) \oplus CV1_i,$$

where f is an involution with no fixed point and $g \neq 0, 1$ is a constant.

For all these double-block length proposals, the conjectured security is 2^{64} and 2^{128} computations respectively to generate a (semi)-free-start collision or preimage for the compression function or hash function. We summarize all of them in Appendix D.

4 Weak-keys for `IDEA`

Weak-keys for `IDEA` has already been studied in details [6, 10, 18], but what we are looking for is slightly different. Indeed, for block cipher cryptanalysis, since the attacker can not control the key input he looks for the biggest possible class of weak-keys, so as to get the highest possible probability that a weak-key will indeed be chosen. In the case of compression function cryptanalysis, the key input is fully known or even controlled by the attacker. The goal is therefore not to find the biggest possible class of weak-keys, but to find the weakest possible key. As we will show for `IDEA`, even if only one weak-key is found, its weakness might directly lead to successful attacks on the whole compression or hash function.

4.1 Analysis of the internal functions

When looking at the internal round function of `IDEA`, one might wonder what would be a weak-key. In `IDEA`, the most annoying functions for the cryptanalyst are clearly the multiplications in $\mathbb{Z}_{2^{16}+1}$. Indeed, these operations are strongly non-linear and provide good diffusion between the different bit positions. On the contrary, XOR operations are linear and do not provide any diffusion between the bit positions, while the additions in $\mathbb{Z}_{2^{16}}$ can be easily approximated linearly and the diffusion between the bit positions only happens through the carry. Moreover, XOR and additions are even weaker in `IDEA` since no rotations are present, comparing with Addition-Rotation-XOR (ARX) designs. Here the rotation is done through the multiplications in $\mathbb{Z}_{2^{16}+1}$ and our goal is therefore to avoid them.

When adding $(a+b) \bmod 2^{16}$, we can avoid any diffusion by forcing one operand to 0. When multiplying $(a \odot b) = (a \cdot b) \bmod 2^{16}+1$, the good diffusion will happen especially when $(a \cdot b) \geq 2^{16}+1$. An easy way to avoid this is to fix one of the two operands to 1. In that case, we have $(a \odot 1) = (a \cdot 1) \bmod 2^{16}+1 = a \bmod 2^{16}$. As already remarked in [10], a good choice is also 0, since

$$\begin{aligned}(a \odot 0) \bmod 2^{16} &= ((a \cdot 2^{16}) \bmod (2^{16}+1)) \bmod 2^{16} \\ &= (((a \cdot 2^{16} + a) + (2^{16}+1) - a) \bmod (2^{16}+1)) \bmod 2^{16} \\ &= (0 + 2^{16} + 1 - a) \bmod 2^{16} = 1 - a \bmod 2^{16} \\ &= 2 + (2^{16} - 1 - a) \bmod 2^{16} = (2 + \overline{a}) \bmod 2^{16}\end{aligned}$$

and the multiplication is reduced to only a complement and an addition with a constant.

4.2 Weak-keys classes

Based on the remark that the operand 0 is very weak for both multiplications and additions, Daemen *et al.* [10] generated a class of weak-keys. A first obvious candidate is the null key (all bits set to zero), which will force all the subkeys to zero as well. As a consequence, all subkeys additions can be simply removed and all subkeys multiplications can be replaced by a complement (or XOR with `0xffff`) and an addition with value 2. At this point, all the operations in `IDEA` with null key are either XOR or additions. Therefore, by inserting differences only on the Most Significant Bit (MSB) of the four 16-bit plaintext input words, the attacker is ensured that only the MSB of the four output words will contain a difference. Even better, the mapping from an MSB input difference pattern to an MSB output difference pattern is completely deterministic (is it linear since on the MSB no carry is propagated). Such a property is largely sufficient to consider the null key as weak. This reasoning can be generalized by observing that the attacker does not necessarily need all subkeys to be null, but only the ones that are multiplied to an internal word which contains a MSB difference. Since the MSB differential paths are quite sparse, many of the null constraints on the subkeys are relaxed and one finally gets 2^{35} weak-keys.

4.3 The null weak-key

We show that the null key is particularly weak for hash function utilization. Even if other keys belong to a weak-key class, they do not present the same special properties as the null key.

Almost half-involution When using the null key, we remark that all subkeys will be null as well. Then, all rounds layers will be the same and we write KA_0 and MA_0 the KA and MA layers with null subkeys. A nice practical feature of `IDEA` is that the decryption is done using the very same algorithm as encryption, but with different subkeys. The decryption subkeys for the MA layer are the same as the encryption ones since the MA layer is an involution (i.e. $\mathrm{MA}=\mathrm{MA}^{-1}$). The decryption subkeys for the KA layer are the respective multiplicative and additive inverses of the encryption subkeys. However, note that a null subkey is both its own multiplicative and additive inverse and the KA layer becomes an involution as well (i.e. $\mathrm{KA}_0=\mathrm{KA}_0^{-1}$). To summarize, using the null key, we are ensured that $\mathrm{KA}_0=\mathrm{KA}_0^{-1}$ and $\mathrm{MA}_0=\mathrm{MA}_0^{-1}$. Note that we trivially have $\mathrm{S}=\mathrm{S}^{-1}$.

Now, since the KA layer and S layer commute, `IDEA` with null key can be rewritten as

$$\begin{aligned} C &= \mathrm{KA}_0 \circ \mathrm{S} \circ \{\mathrm{S} \circ \mathrm{MA}_0 \circ \mathrm{KA}_0\}^8(P) \\ &= \mathrm{KA}_0 \circ \mathrm{S} \circ \{\mathrm{S} \circ \mathrm{MA}_0 \circ \mathrm{KA}_0\}^3 \circ \mathrm{S} \circ \mathrm{MA}_0 \circ \mathrm{KA}_0 \circ \{\mathrm{S} \circ \mathrm{MA}_0 \circ \mathrm{KA}_0\}^4(P) \\ &= \underbrace{\mathrm{KA}_0 \circ \mathrm{MA}_0 \circ \{\mathrm{S} \circ \mathrm{KA}_0 \circ \mathrm{MA}_0\}^3}_{\sigma^{-1}} \circ \underbrace{\mathrm{KA}_0 \circ \mathrm{S}}_{\theta} \circ \underbrace{\{\mathrm{MA}_0 \circ \mathrm{KA}_0 \circ \mathrm{S}\}^3 \circ \mathrm{MA}_0 \circ \mathrm{KA}_0}_{\sigma}(P) \end{aligned}$$

which eventually gives $C = \sigma^{-1} \circ \theta \circ \sigma(P)$. One can check that since KA_0, MA_0 and S are involutions, the operation denoted by σ^{-1} is indeed the inverse of the one denoted by σ. Thus, using the notation

$$P \xrightarrow{\sigma^{-1}} U \xrightarrow{\theta} V \xrightarrow{\sigma} C$$

where U and V are internal state values, we have

$$P \xleftarrow{\sigma} U \xrightarrow{\theta} V \xrightarrow{\sigma} C.$$

We will use this almost half-involution property in Section 6 to find free-start collisions and even hash function collisions for some `IDEA`-based constructions.

T-function: When using the null key, we have already described that all operations remaining are either XOR or additions. These operations are triangular functions [23] (or T-functions) in the sense that any output bit at position i only depends on the input bits located at a position i or lower. A composition of T-functions is itself a T-function, therefore the whole permutation defined by `IDEA` with the null key is a T-function. As shown in [34], this property might be very dangerous in a hash function design. We will explain in Section 7 how to exploit this weakness and compute preimages by guessing the input words bit layer by bit layer.

5 Simple collision attacks

As shown by Daemen *et al.* [10], when using the null key for the encryption process of `IDEA`, differences inserted uniquely on the MSB of the four 16-bit input plaintext words will lead to differences on the MSB of the four 16-bit output ciphertext words. Moreover, since this difference mapping is linear (the difference on the carry is not propagated further than the MSB), all possible differential characteristics have a differential probability 1. For example, we denote by $\delta_{MSB} =$ `0x8000` the 16-bit word with difference only on the MSB and by $\Delta_{MSB} = (\delta_{MSB}, \delta_{MSB}, \delta_{MSB}, \delta_{MSB})$ the 64-bit difference composed of 4 words with difference δ_{MSB}. Then, Δ_{MSB} propagates to itself with probability 1 through one round of IDEA, or through its last half-round. Therefore, we have with probability 1

$$\Delta_{MSB} \xrightarrow{\texttt{IDEA}_{K=0}} \Delta_{MSB}.$$

Note that instead of using δ_{MSB} only, one can generalize the input difference space and obtain other very good differential paths for the encryption of IDEA with the null key. However, we omit this generalization here since the methods described in later sections already provide much better attacks.

Davies-Meyer. Finding a free-start collision on Davies-Meyer mode instantiated with IDEA is very easy. Since the difference Δ_{MSB} is mapped to itself through the IDEA encryption process with the null key, the attacker only has to pick $M = 0$. Then, any value of CV with difference Δ_{MSB} applied to it will lead to a collision with probability 1. We give in Appendix C.1 examples of such a free-start collision.

Hirose. The same method as for Davies-Meyer mode can be applied to the Hirose mode in order to find free-start collisions. The attacker fixes $CV2 = 0$ and $M = 0$ so as to force the null key to both encryptions. Then, any value of $CV1$ with a difference Δ_{MSB} applied to it will lead to a collision with probability 1, since Δ_{MSB} will appear on the plaintext input of both encryptions with the null key. We give in Appendix C.3 examples of such a free-start collision.

Abreast-DM. This technique seems impossible to apply to the Abreast-DM mode since forcing a difference Δ_{MSB} on any of the two encryptions plaintext input will imply a difference inserted in the key input of the other encryption block. Therefore, one cannot use Δ_{MSB} difference on plaintext input with null key in both encryption blocks. Even if the attacker tries to attack only one encryption block with this method, the other block will not be controlled and he will have to deal with random differences on its output. These random differences cannot be dealt with some birthday technique because fixing all inputs of one encryption block will fix all inputs of the other one as well.

Tandem-DM. This technique seems impossible to apply to the Tandem-DM mode for the exact same reasons as for Abreast-DM.

Peyrin *et al.*(II). We have to separate in two groups the possible instances of this construction, obtained by permuting the position of the three inputs of each internal function f_i. If all compression function inputs $CV1$, $CV2$, $M1$ and $M2$ appear in at least one of the `IDEA` key inputs of any f_i internal function, then the attack will not apply. Indeed, since all inputs will be involved at least one time, the attacker will necessarily have to insert a difference in at least one `IDEA` key input and he will not be able to use the differential path with probability 1. Note that these instances would be avoided in practice because they would lead to more frequent re-keying and therefore reduce the overall performance of the hash function. If this condition is not met, then we can apply the following free-start collision attack. Let $X \in \{CV1, CV2, M1, M2\}$ denote the input that is missing in all the `IDEA` key inputs of the compression function. The attacker simply fixes the difference Δ_{MSB} on X (one can give any value to X) and all other inputs are set to 0 in order to get the null key in every internal `IDEA`. The attacker ends up with several Davies-Meyer in parallel, with either no difference at all or with null key and Δ_{MSB} as plaintext input difference. Thus, he obtains a collision with probability 1. If $X \notin \{CV1, CV2\}$, then this attack finds semi-free-start collisions.

MJH-Double. The MJH-Double mode prevents this simple attack since even if we fix $CV2 = 0$ and $M2 = 0$ in order to get the null key in both encryptions, it is hard to force the difference Δ_{MSB} on both their plaintext inputs. Indeed, the f operation will randomize the difference and in order for the attack to run, we would require $\Delta_{MSB} \xrightarrow{f} \Delta_{MSB}$ which is unlikely to happen.

6 Improved collision attacks

In this section, using the almost half-involution property with the null key, we will show how to get the same difference on the input and on the output of the `IDEA` ciphering process with good probability. Then, we will use this weakness to derive our collision attacks, for any number of rounds.

6.1 Exploiting the almost half-involution

We have already shown in Section 4 that when the key is null, `IDEA` encryption process can be rewritten as

$$P \xleftarrow{\sigma} U \xrightarrow{\theta} V \xrightarrow{\sigma} C$$

where

$$\sigma = \{\mathrm{MA}_0 \circ \mathrm{KA}_0 \circ \mathrm{S}\}^3 \circ \mathrm{MA}_0 \circ \mathrm{KA}_0 \qquad \text{and} \qquad \theta = \mathrm{KA}_0 \circ \mathrm{S}.$$

We denote ΔU the XOR difference between two 64-bit internal state values U and U', i.e $\Delta U = U \oplus U'$, and δU_i represents the 16-bit difference on the i-th word of ΔU, that is $\Delta U = (\delta U_1, \delta U_2, \delta U_3, \delta U_4)$. Let us consider two random 64-bit internal state values U and U' such that $\delta U_2 = \delta U_3$ and we denote this 16-bit difference δ_M. For truly random values U and U', this condition happens with probability 2^{-16}. One can check that applying θ on U and U' to obtain V and V' respectively will lead to $\delta V_2 = \delta V_3 = \delta_M$ since layer S only switches the two middle words and layer KA_0 has no effect on them (addition of null subkeys).

Let δ_L and δ_R represent the difference on δU_1 and δU_4 respectively, i.e. $\Delta U = (\delta_L, \delta_M, \delta_M, \delta_R)$. Applying function θ to U and U', we would like the same differences to appear on internal state V and V': $\Delta V = (\delta_L, \delta_M, \delta_M, \delta_R)$. The previous condition with probability 2^{-16} already ensures the two middle differences being the same δ_M. Concerning differences δ_L and δ_R, they will both be unaffected by layer S, but they might be modified through layer KA_0 that applies a multiplication with a null subkey. Therefore, we need to study the probability that a random difference δ is mapped to itself through a multiplication by the null subkey. We show in Appendix B that this probability is equal to $2^{-1.585}$ and finally we have $\Pr[(\delta_L, \delta_M, \delta_M, \delta_R) \xrightarrow{\theta} (\delta_L, \delta_M, \delta_M, \delta_R)] = 2^{-3.17}$.

At this point, we proved that for randomly chosen internal state values U and U', we will observe with probability $2^{-19.17}$ the same difference on U and V, i.e. $\Delta U = \Delta V$.

One can see that computing backward from internal states U to P or forward from V to C, the function σ is applied. Our final goal is to have the same difference on P and C. However, this seems unlikely to happen since U and V have different values, the forward and backward computations of σ should be completely unrelated, even with the same input difference. Yet, this reasoning does not take in account the fact that while U and V have distinct values, they are far from being independent:

$V = \theta(U)$ with θ being a very light function. Moreover, we remarked that almost each time that we got the same difference on P and C, the same differences were observed as well in all rounds of the forward and backward σ computations (the round success probability increasing with the number of rounds already processed). Because all the rounds are not independent and because U and V are strongly related, it is very difficult to compute theoretically the probability of observing the same difference on P and C and we leave this as an open problem. Therefore, we measured it by choosing random values of U, δ_L, δ_M, δ_R, computing $V = \theta(U)$, and checking for collisions on the difference of P and C. The probability obtained was $2^{-16.26}$ for about 2^{28} tests (note that this probability somehow contains the $2^{-3.17}$ probability computed previously, but we can not separate them because the two events are not independent).

To conclude, the probability that two randomly chosen internal state values U and U' give the same difference on P and C is equal to $2^{-16-16.26} = 2^{-32.26}$ (instead of 2^{-64} expected for a random function). In other words, using the birthday paradox, one can find such a pair with about $2^{16.13}$ computations.

Interestingly, we have observed that most of the pairs fulfilling the differential path for the full `IDEA` will also be valid for a strengthened version of the cipher with any number of additional rounds. Since the subkeys are always null, strengthening the cipher would mean that $\sigma = \{\mathrm{MA}_0 \circ \mathrm{KA}_0 \circ \mathrm{S}\}^t \circ \mathrm{MA}_0 \circ \mathrm{KA}_0$ for any $t > 3$. We checked that the probability that two randomly chosen internal state values U and U' give the same difference on P and C tends to $2^{-32.54}$ when t tends to infinite. Thus, similarily to the method presented in the previous section, the attacks using this almost half-involution property will work for any number of rounds.

6.2 Improving collision attacks

Davies-Meyer. A first obvious application of having the same difference in P and C is collision search on Davies-Mayer mode, where the feed-forward will cancel the two differences in the output. The attack finds collisions for the whole hash function and the procedure is very simple: we start from the IV and add random differences in the first message block M_0. This will cause random differences in the the first chaining variable CV_1. For the second message block M_1, we will set all its bits 0 ($M_1 = 0$), forcing the internal `IDEA` computation to use the null key. Since we estimated in the previous section that with the null key a random pair of inputs has a probability $2^{-32.26}$ to give the same input/output difference, one can use the birthday paradox to generate a collision on CV_2 with only $2^{16.13}$ distinct message blocks M_0. We give in Appendix C.2 examples of hash collisions for the Davies-Meyer mode. Note that finding semi-free-start collisions with this technique is impossible since we would have to insert differences in the message input, which forbids the use of the null key in the internal cipher.

Hirose. We already showed how to find free-start collisions for the Hirose mode. However, finding semi-free-start collisions with this technique is impossible since we would have to insert differences in the message input, which forbids the use of the null key in the internal cipher. Also, concerning hash collisions, it seems hard as well because forcing the null key during iteration i requires us to obtain a chaining variable $CV2_{i-1} = 0$ during the previous iteration. This half-preimage already costs the same complexity as a generic collision search on the entire compression function.

Abreast-DM. One can derive a free-start collision attack for the Abreast-DM compression function using this technique. The attacker first fixes $CV1 = 0$ and $M = 0$. Then, he builds a set of $2^{48.13}$ distinct values $CV2$ and checks if a pair of this set leads to a collision. The probability that a pair leads to a collision on the first (top) branch is $2^{-32.26}$ (since the internal cipher on this part has the null key), and 2^{-64} on the other half. Overall, using the birthday paradox on the set of $2^{48.13}$ values $CV2$ is sufficient to have a good chance to obtain a collision. Note that finding a semi-free-start collision for the compression function or a collision for the hash function seems impossible with this method, for the same reasons as the Hirose mode.

Tandem-DM. The situation of Tandem-DM is absolutely identical to the Abreast-DM one: one can find free-start collisions for compression function using this technique. The attacker first fixes $CV1 = 0$ and $M = 0$. Then, he builds a set of $2^{48.13}$ distinct values $CV2$ and checks if a pair of this set leads to a collision. The probability that a pair leads to a collision on the first (top) branch is $2^{-32.26}$ (since the internal cipher on this part has the null key), and 2^{-64} on the other half. Overall, using the birthday paradox on the set of $2^{48.13}$ values $CV2$ is sufficient to have a good chance to obtain a collision. Again, finding a semi-free-start collision for the compression function or a collision for the hash function seems impossible with this method, for the same reasons as the Hirose mode.

Peyrin *et al.*(II). We showed in previous section how to find (semi)-free-start collisions with probability 1 for a certain subset of Peyrin *et al.*(II) constructions, but here we provide attacks on a bigger subset. If all compression function inputs $CV1$, $CV2$, $M1$ and $M2$ appear in at least one of the `IDEA` key inputs of f_1, f_2, f_3 (left side) and in at least one of the `IDEA` key inputs of f_3, f_4, f_5 (right side), then the attack will not apply. Indeed, for both left side and right side of the compression function, the attacker will necessarily have to insert a difference in at least one key input (since all inputs will be involved) and he will not be able to use the null key completely. Note that these instances would be avoided in practice because they would lead to more frequent rekeying and therefore reduce the overall performance of the hash function. However, if this condition is not met, then we can apply the following free-start collision attack. Let $X \in \{CV1, CV2, M1, M2\}$ denote the input that is missing in all the `IDEA` key inputs of f_1, f_2, f_3 (wlog the reasoning is the same with f_3, f_4, f_5). The attacker first fixes all inputs but X to 0 in order to get the null key in every internal `IDEA` on the left side. Then he chooses $2^{48.13}$ random values for X and checks among them if any pair collides on the whole compression function output. Since he has a probability $2^{-32.26}$ to get a collision on the left side and 2^{-64} on the right side, using a birthday search the attacker finds a solution with complexity $2^{48.13}$. Again, if $X \notin \{CV1, CV2\}$, then this attack finds semi-free-start collisions. However, finding a collision for the hash function seems impossible with this method, because at least one of the chaining variable inputs $CV1$ and $CV2$ will be present as key input for one of the `IDEA` internal emcryption. Setting this word to 0 is equivalent to a half-preimage that already costs the same complexity as a generic collision search on the entire hash function.

MJH-Double. One can derive a semi-free-start collision attack on the MJH-Double compression function instantiated with `IDEA`. The attacker first fixes $CV2 = 0$ and $M2 = 0$ and this will force the null key in both encryptions. Now he chooses a random value for $CV1$ (note that actually this value could be fixed by the challenger) and builds a set of $2^{32.26}$ values $M1$. In this configuration, it is easy to see that one will have random differences on the plaintext inputs to both encryptions. Since the null key is used for both, we have a probability $2^{-64.52}$ that a pair of

$M1$ leads to a collision after the feed-forward of both encryptions (on the output of the bottom block and just before the application of g on the top block). Therefore, with a birthday technique, one can find such a pair with only $2^{32.26}$ computations. Note that while this pair will directly lead to a collision on the bottom $CV1$ output, the difference on $M1$ is injected two times before computing the top $CV2$ output. Two times of the same difference will cancel themselves and we eventually get a full semi-free-start collision. Note that it seems hard to extend this attack to a hash collision since the attacker would require to force the incoming chaining variable $CV2$ to be equal to 0 and this half-preimage already costs the same complexity as a generic collision search on the entire hash function.

7 Preimage attacks

We showed in Section 4 that if used with the null key, the whole permutation defined by `IDEA` is a T-function. Since any output bit at position i only depends on the input bits located at a position i or lower, we reuse the idea of preimage attack for hash functions based on T-functions [34] where the preimage is computed bit layer by bit layer, starting from the LSB. However, here our situation is different than the functions studied in [34] since we do not have any truncation or reduction of the internal state at the end of the process.

We denote by p the probability that given a random challenge, our algorithm outputs a preimage for this challenge. We denote by s the average number of preimage solutions that the algorithm will output, given that at least one is found. The average number of solutions outputted by our algorithm is then $A = s \cdot p$. For an n-bit ideal compression function, a generic attack restricted to C computations can generate $A = C \cdot 2^{-n}$ solutions on average. Thus, we can consider that a preimage attack is found if we exhibit an algorithm that outperforms this generic complexity.

Davies-Meyer. Since the key is fixed to 0 and since the plaintext and ciphertext sizes are the same, we trivially have that $A = 1$. We measured[3] that $p = 2^{-17.50}$, thus we directly deduce that $s = A/p = 2^{17.5}$. A straightforward implementation is a recursive depth first search, attacking the T-function by bit layer from the LSB to the MSB of the 16-bit state words. Wrong candidates at lower layers are discarded thanks to

[3] from 2^{31} random challenges, we measured that $p = 2^{-17.50}$ and $s = 2^{17.74}$.

an early-abort strategy. On average, the amount of `IDEA` encryptions required to find all the possible preimages (if at least one can be found) can be estimated as $C \simeq 16 \cdot 2^4 \cdot s = 2^{25.5}$, since we have 16 bit layers, each having 4 bits of input, and on average the number of candidates in one layer is s. This is a very conservative estimation since only $p = 2^{-17.50}$ of the challenges on average will eventually lead to a solution and the early-abort strategy will make the actual search of very low complexity. In the ideal case, with $C = 2^{25.5}$ computations allowed, an attacker should only be able to generate $A = 2^{25.5-64} = 2^{-38.5}$ solutions on average for an ideal 64-bit compression function. We give an example of a preimage in Appendix C.4.

Hirose. We can reuse the attack on Davies-Meyer, but only one of the two branches will be controlled, with the other behaving randomly. We first find a preimage for the first branch (with probability $2^{-17.5}$) and then use the $2^{17.5}$ solutions on average to also match the second branch (with probability $2^{17.5-64} = 2^{-46.5}$). Therefore, our preimage search algorithm have parameters $p = 2^{-17.5-46.5} = 2^{-64}$ and $s = 1$, while the average number of preimage solutions found is $A = 2^{-64}$. The complexity of the search is equivalent to the Davies-Meyer case, $C = 2^{25.5}$. For an attacker using at most $2^{25.5}$ computations on an ideal 128-bit compression function, the average number of solutions he could find is only $2^{-102.5}$.

Abreast-DM. Similarly to Hirose, by setting for example $M = CV1 = 0$, one can attack one branch bit layer by bit layer while the other branch will behave randomly. The complexity analysis is identical to Hirose's case.

Tandem-DM. Similarly to Hirose, by setting $M = CV1 = 0$, one can attack one branch bit layer by bit layer while the other branch will behave randomly. The complexity analysis is identical to Hirose's case.

Peyrin *et al.*(II). If all compression function inputs $CV1$, $CV2$, $M1$ and $M2$ appear in at least one of the `IDEA` key inputs of f_1, f_2, f_3 (left side) and in at least one of the `IDEA` key inputs of f_3, f_4, f_5 (right side), then the attack will not apply (because the attacker will not be able to use the null key completely). Otherwise, similarly to Hirose, by setting all `IDEA` keys to 0 on one side, one can attack it bit layer by bit layer while the other side will behave randomly. The complexity analysis is identical to Hirose's case.

MJH-Double. The attacker first fixes $M2 = CV2 = 0$ so as to get the null key for both `IDEA` encryptions. Then, similarly to the Davies-Meyer case, he find a preimage with probability $p = 2^{-17.5}$ for one of the two sides and this defines the value of $M1 \oplus CV1$. In order to get the preimage on the second side as well, the attacker only has to modify the value of $M1$ accordingly. If a solution is found on the first side, the attacker therefore gets $s = 2^{17.5}$ preimages. On average, he finds $A = 1$ solutions and the complexity is again $2^{25.5}$ computations. For an attacker using at most $2^{25.5}$ computations on an ideal 128-bit compression function, the average number of solutions he should find is only $2^{-102.5}$.

8 Results and implementations

We depict in Table 1 our results for the block cipher to compression function modes considered in this article when instantiated with `IDEA`. We implemented all attacks of reasonable complexities and provide in Appendix C the collision/preimage examples obtained.

Table 1. Summary of results for block cipher to compression function modes when instantiated with `IDEA` (we did not include MDC-2 as it does not provide ideal collision resistance). The preimage complexity results find s preimages on average with a certain probability p, for a total average of $A = s \cdot p$ solutions. The results for Peyrin *et al.*(II) construction, marked with a *, depend on the instance considered (see relevant parts of Sections 5, 6 and 7 for more details).

Mode	hash output size	compression function: free-start collision attack	compression function: semi-free-start collision attack	compression function: preimage attack complexity (s, p)	hash function collision attack
Davies-Meyer [1]	64	2^1		$2^{25.5}$ $(2^{17.5}, 2^{-17.5})$	$2^{16.13}$
Hirose [19, 20]	128	2^1		$2^{25.5}$ $(1, 2^{-64})$	
Abreast-DM [27, 39]	128	$2^{48.13}$		$2^{25.5}$ $(1, 2^{-64})$	
Tandem-DM [27, 39]	128	$2^{48.13}$		$2^{25.5}$ $(1, 2^{-64})$	
Peyrin *et al.*(II) [35]	128	2^1 / $2^{48.13\star}$	2^1 / $2^{48.13\star}$	$2^{25.5}$ $(1, 2^{-64})^\star$	
MJH-Double [29]	128	$2^{32.26}$	$2^{32.26}$	$2^{25.5}$ $(2^{17.5}, 2^{-17.5})$	

Conclusion

In this article, we showed collision and preimage attacks for several single and double-length block cipher based compression function constructions when instantiated with the block cipher `IDEA`. Namely, we analyzed all known double-key schemes such as Davies-Meyer, Hirose, Abreast-DM, Tandem-DM, Peyrin *et al.* (II) and MJH-Double. While most of these constructions are conjectured or proved to be secure in the ideal cipher model, we showed that their security is very weak when instantiated with the block cipher `IDEA`, which remains considered as secure in the secret key model. In particular, we answer in the negative for the 20-year-old standing open question concerning the security of the Abreast-DM and Tandem-DM instantiated with `IDEA`. All our practical attacks have been implemented and they can work even for any number of `IDEA` rounds. Our results indicate that one has to be very careful when hashing with a block cipher that presents any weakness when the key is known or controlled by the attacker. Also, since we extensively use the presence of weak-keys for `IDEA`, as a future work it would be interesting to look at the security of hash functions based on block ciphers for which some key sets are known to be weaker than others.

Acknowledgments

The authors would like to thank the anonymous referees for their helpful comments.

References

1. A. Menezes, P. van Oorschot, and S. Vanstone. CRC-Handbook of Applied Cryptography. CRC Press, 1996.
2. Eyüp Serdar Ayaz and Ali Aydin Selçuk. Improved DST Cryptanalysis of IDEA. In Eli Biham and Amr M. Youssef, editors, *Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.
3. Eli Biham, Orr Dunkelman, and Nathan Keller. New Cryptanalytic Results on IDEA. In Lai and Chen [25], pages 412–427.
4. Eli Biham, Orr Dunkelman, and Nathan Keller. A New Attack on 6-Round IDEA. In Alex Biryukov, editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 211–224. Springer, 2007.
5. Eli Biham, Orr Dunkelman, Nathan Keller, and Adi Shamir. New Data-Efficient Attacks on Reduced-Round IDEA. Cryptology ePrint Archive, Report 2011/417, 2011.
6. Alex Biryukov, Jorge Nakahara Jr., Bart Preneel, and Joos Vandewalle. New Weak-Key Classes of IDEA. In Robert H. Deng, Sihan Qing, Feng Bao, and Jianying Zhou, editors, *ICICS*, volume 2513 of *Lecture Notes in Computer Science*, pages 315–326. Springer, 2002.
7. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002.

8. Gilles Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *LNCS*. Springer, 1990.
9. Donghoon Chang. Near-Collision Attack and Collision-Attack on Double Block Length Compression Functions based on the Block Cipher IDEA. Cryptology ePrint Archive, Report 2006/478, 2006. `http://eprint.iacr.org/`.
10. Joan Daemen, René Govaerts, and Joos Vandewalle. Weak Keys for IDEA. In Stinson [37], pages 224–231.
11. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.
12. Ivan Damgård. A Design Principle for Hash Functions. In Brassard [8], pages 416–427.
13. Hüseyin Demirci, Ali Aydin Selçuk, and Erkan Türe. A New Meet-in-the-Middle Attack on the IDEA Block Cipher. In Matsui and Zuccherato [32], pages 117–129.
14. Bert den Boer and Antoon Bosselaers. Collisions for the Compressin Function of MD5. In *EUROCRYPT*, pages 293–304, 1993.
15. Hans Dobbertin. Cryptanalysis of MD5 compress. Presented at the rump session of EUROCRYPT 1996, 1996.
16. Ewan Fleischmann, Michael Gorski, and Stefan Lucks. On the Security of Tandem-DM. In Orr Dunkelman, editor, *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 84–103. Springer, 2009.
17. Ewan Fleischmann, Michael Gorski, and Stefan Lucks. Security of Cyclic Double Block Length Hash Functions including Abreast-DM. Cryptology ePrint Archive, Report 2009/261, 2009. `http://eprint.iacr.org/`.
18. Philip Hawkes. Differential-Linear Weak Key Classes of IDEA. In *EUROCRYPT*, pages 112–126, 1998.
19. Shoichi Hirose. Provably Secure Double-Block-Length Hash Functions in a Black-Box Model. In Choonsik Park and Seongtaek Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342. Springer, 2004.
20. Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.
21. John Kelsey, Bruce Schneier, and David Wagner. Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 1996.
22. Dmitry Khovratovich, Gaetan Leurent, and Christian Rechberger. Narrow Bicliques: Cryptanalysis of Full IDEA. In *EUROCRYPT*, 2012, to appear.
23. Alexander Klimov and Adi Shamir. Cryptographic Applications of T-Functions. In Matsui and Zuccherato [32], pages 248–261.
24. Lars R. Knudsen and Vincent Rijmen. Known-Key Distinguishers for Some Block Ciphers. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 315–324. Springer, 2007.
25. Xuejia Lai and Kefei Chen, editors. *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*. Springer, 2006.
26. Xuejia Lai and James L. Massey. A Proposal for a New Block Encryption Standard. In *EUROCRYPT*, pages 389–404, 1990.

27. Xuejia Lai and James L. Massey. Hash Function Based on Block Ciphers. In *EUROCRYPT*, pages 55–70, 1992.
28. Jooyoung Lee and Daesung Kwon. The Security of Abreast-DM in the Ideal Cipher Model. Cryptology ePrint Archive, Report 2009/225, 2009. `http://eprint.iacr.org/`.
29. Jooyoung Lee and Martijn Stam. MJH: A Faster Alternative to MDC-2. In Aggelos Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 213–236. Springer, 2011.
30. Jooyoung Lee, Martijn Stam, and John P. Steinberger. The Collision Security of Tandem-DM in the Ideal Cipher Model. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 561–577. Springer, 2011.
31. Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
32. Mitsuru Matsui and Robert J. Zuccherato, editors. *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003, Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*. Springer, 2004.
33. Ralph C. Merkle. One Way Hash Functions and DES. In Brassard [8], pages 428–446.
34. Frédéric Muller and Thomas Peyrin. Cryptanalysis of T-Function-Based Hash Functions. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC*, volume 4296 of *Lecture Notes in Computer Science*, pages 267–285. Springer, 2006.
35. Thomas Peyrin, Henri Gilbert, Frédéric Muller, and Matthew J. B. Robshaw. Combining Compression Functions and Block Cipher-Based Hash Functions. In Lai and Chen [25], pages 315–331.
36. Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In Stinson [37], pages 368–378.
37. Douglas R. Stinson, editor. *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*. Springer, 1994.
38. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
39. Xuejia Lai. On the Design and Security of Block Ciphers. Hartung-Gorre Verlag, Konstanz, 1992.

A The `IDEA` subkeys

B Proof of difference preservation through multiplication with a null subkey

We prove in this section that for randomly chosen values a and a' with $a \oplus a' = \delta$, the probability that the difference δ is preserved after multiplication by the null subkey is equal to $2^{-1.585}$. The condition we expect can be translated into the following equation

$$\delta = a \oplus a' = (a \odot 0) \oplus (a' \odot 0).$$

i-th round	$Z_1^{(i)}$	$Z_2^{(i)}$	$Z_3^{(i)}$	$Z_4^{(i)}$	$Z_5^{(i)}$	$Z_6^{(i)}$
1	0-15	16-31	32-47	48-63	64-79	80-95
2	96-111	112-127	25-40	41-56	57-72	73-88
3	89-104	105-120	121-8	9-24	50-65	66-81
4	82-97	98-113	114-1	2-17	18-33	34-49
5	75-90	91-106	107-122	123-10	11-26	27-42
6	43-58	59-74	100-115	116-3	4-19	20-35
7	36-51	52-67	68-83	84-99	125-12	13-28
8	29-44	45-60	61-76	77-92	93-108	109-124
OT	22-37	38-53	54-69	70-85		

Table 2. Key bits used for subkeys $Z_j^{(i)}$ in the i-th round of `IDEA`

Since the $\odot$ operation is equivalent to a complement (or XOR with `0xffff`) and an addition with value 2, we can rewrite

$$\begin{aligned}\delta &= ((a \oplus \texttt{0xffff}) + 2) \oplus ((a' \oplus \texttt{0xffff}) + 2)\\ \delta &= ((a \oplus \texttt{0xffff}) + 2) \oplus ((a \oplus \delta \oplus \texttt{0xffff}) + 2)\\ \delta &= (b + 2) \oplus ((b \oplus \delta) + 2)\\ \delta \oplus (b + 2) &= (b \oplus \delta) + 2\end{aligned}$$

where $b = a \oplus \texttt{0xffff}$. One can check that the least significant bit condition of this equation is always fulfilled.

If the second least significant bit of b is 0 (probability 1/2), then $(b + 2) = b \oplus 2$ and the equation is fulfilled if and only if the second least significant bit of $(b \oplus \delta)$ is also 0 (probability 1/2). Overall, this situation happens with probability 1/4.

If the second least significant bit of b is 1 (probability 1/2), then we will have a carry propagating and we require the second least significant bit of $(b \oplus \delta)$ to be also 1 (probability 1/2). If the third least significant bit of b is 0 (probability 1/2), then $(b+2) = b \oplus 6$ and the equation is fulfilled if and only if the third least significant bit of $(b \oplus \delta)$ is also 0 (probability 1/2). Overall, this situation happens with probability $(1/4)^2$.

Continuing this reasoning over all the bits layers, we obtain that the success probability is equal to

$$\sum_{i=1}^{14} (1/4)^i = 2^{-1.585}.$$

C Collision and preimage examples

C.1 Free-start collision for Davies-Meyer mode

CV_i : `0x9efc 0x14ef 0x85d6 0xc557`
CV_i' : `0x1efc 0x94ef 0x05d6 0x4557`

$M = M'$: `0`

$CV_{i+1} = H(CV_i, M)$: `0x7f11 0x83f1 0x7617 0x8af3`
$CV_{i+1}' = H(CV_i', M')$: `0x7f11 0x83f1 0x7617 0x8af3`

C.2 Hash function collision for Davies-Meyer mode

We use as initial value the first 64 output bits of the `SHA-2` computation of the string "IDEA":

`SHA-2`("$IDEA$") = "`9f8c7b26cde59ca3dacc74ec7afda737ac1d15aa5239206416f79019dbd7ec37`"

IV: $IV_1 =$ `0x9f8c`, $IV_2 =$ `0x7b26`, $IV_3 =$ `0xcde5`, $IV_4 =$ `0x9ca3`

M_1: `0xdacc 0xdacc 0xdacc 0xdacc 0xdacc 0xdacc 0xcadc 0x0282`
M_1': `0xdacc 0xdacc 0xdacc 0xdacc 0xdacc 0xdacc 0xcade 0x1a3f`

$CV_1 = H(IV, M_1)$: `0xb782 0x4583 0x83b6 0x0bef`
$CV_1' = H(IV, M_1')$: `0x1ce2 0x8553 0xe656 0x4387`

$CV_2 = H(CV_1, 0)$: `0xdffd 0x3ffd 0x8e7d 0x6e7d`
$CV_2' = H(CV_1', 0)$: `0xdffd 0x3ffd 0x8e7d 0x6e7d`

C.3 Free-start collision for Hirose mode

For Hirose mode, we used as constant c the first 64 output bits of the `SHA-2` computation of the string "IDEA":

`SHA-2`("$IDEA$") = "`9f8c7b26cde59ca3dacc74ec7afda737ac1d15aa5239206416f79019dbd7ec37`"

c : `0x9f8c 0x7b26 0xcde5 0x9ca3`

$CV1_i$: `0x93e8 0x4d86 0x45a5 0xa829`
$CV1_i'$: `0x13e8 0xcd86 0xc5a5 0x2829`

$CV2_i = CV2'_i$: `0`

$M = M'$: `0`

$CV1_{i+1}$: `0x2101 0x23c9 0xde42 0xdc96`
$CV1'_{i+1}$: `0x2101 0x23c9 0xde42 0xdc96`
$CV2_{i+1}$: `0x0009 0x0401 0x3d38 0x3934`
$CV2'_{i+1}$: `0x0009 0x0401 0x3d38 0x3934`

C.4 Preimage for Davies-Meyer mode

Since a random 64-bit challenge has preimage(s) with a probability p, we show the preimage of a challenge which we are sure at least one preimage exists (similar to a second-preimage search). In order to get the challenge, we use as input the first 64 output bits of the `SHA-2` computation of the string "IDEA", and provide one of the preimages found:

`SHA-2`("$IDEA$") = "`9f8c7b26cde59ca3dacc74ec7afda737ac1d15aa5239206416f79019dbd7ec37`"

The challenge $H($`0x9f8c7b26cde59ca3`$, 0)$: `0x20ad1fc924e61ba2`

$CV_{i+1} = H(CV_i, M)$: `0x20ad 0x1fc9 0x24e6 0x1ba2`
M : `0`

CV_i : `0x1860 0x002e 0x2d82 0x0200`

CV_i is one preimage out of $2^{23.585}$ for CV_{i+1}, the search takes $2^{25.486}$ `IDEA` encryptions, and the average cost per preimage is around $2^{1.9}$.

D Double-key block cipher based compression functions

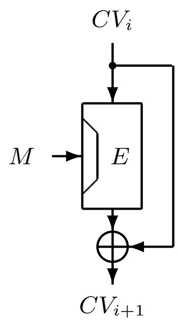

Fig. 2. Davies-Meyer

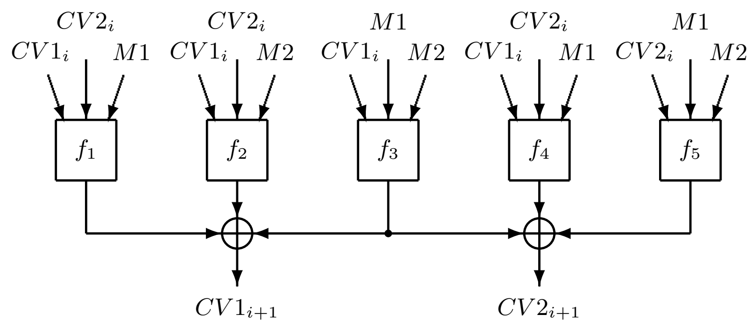

Fig. 3. Peyrin *et al.* (II)

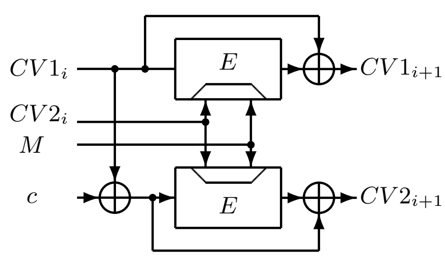

Fig. 4. Hirose

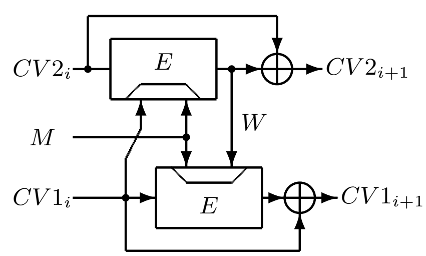

Fig. 5. Tandem-DM

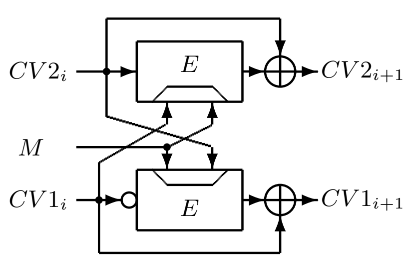

Fig. 6. Abreast-DM

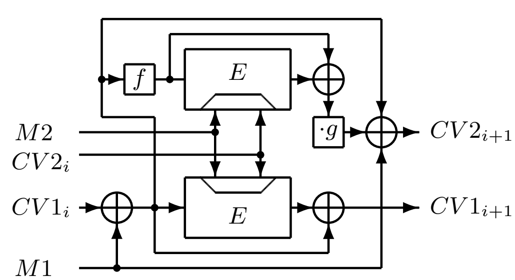

Fig. 7. MJH-Double