Cryptanalysis of Tweaked Versions of SMASH and Reparation

Pierre-Alain Fouque, Jacques Stern,
and Sébastien Zimmer

CNRS-École normale supérieure-INRIA, Paris, France
{Pierre-Alain.Fouque,Jacques.Stern,Sebastien.Zimmer}@ens.fr

Abstract. In this paper, we study the security of permutation based hash functions, *i.e.* blockcipher based hash functions with fixed keys. SMASH is such a hash function proposed by Knudsen in 2005 and broken the same year by Pramstaller *et al.* Here we show that the two tweaked versions, proposed soon after by Knudsen to thwart the attack, can also be attacked in collision in time $\mathcal{O}(n2^{n/3})$. This time complexity can be reduced to $\mathcal{O}(2^{2\sqrt{n}})$ for the first tweak version, which means an attack against SMASH-256 in $c \cdot 2^{32}$ for a small constant c. Then, we show that an efficient generalization of SMASH, using two permutations instead of one, can be proved secure against collision in the ideal-cipher model in $\Omega(2^{n/4})$ queries to the permutations. In order to analyze the tightness of our proof, we devise a non-trivial attack in $\mathcal{O}(2^{3n/8})$ queries. Finally, we also prove that our construction is preimage resistant in $\Omega(2^{n/2})$ queries, which the best security level that can be reached for 2-permutation based hash functions, as proved in [12].

1 Introduction

Hash functions have recently been the subject of many attacks, revealing weaknesses in widely trusted hash functions such as MD5 or SHA-1. For this reason, some recent papers deal with new designs for hash functions such as SMASH [6] or Radiogatún [2,3]. Most of previous constructions of hash functions use blockciphers, since we know how to build such secure and efficient primitives and since good constructions of compression functions based on them are known. However, in the proofs of classical constructions of compression function, such as Davies-Meyer used in MD5 and SHA-1, the assumption made on the blockcipher is very strong, namely that *for each key*, or message for hash function, *the blockcipher acts as a random permutation.* Such an assumption, which has been introduced by Shannon and formalized in the ideal-cipher model in 1998 by Bellare *et al.* in [1], is impossible to check. For instance, it is possible that among the 2^{512} possible keys of the SHACAL blockcipher used for SHA-1, some weak keys exist, which could be used by an attacker. One solution to restrict the power of the adversary is to fix the key as it is the case in SMASH.

In order to study blockcipher for such constructions, Knudsen and Rijmen at Asiacrypt last year [7] proposed to use the notion of known-key distinguisher.

R. Avanzi, L. Keliher, and F. Sica (Eds.): SAC 2008, LNCS 5381, pp. 136–150, 2009.

The latter is an adversary which tries to distinguish a blockcipher from random permutation when the key is known. This model seems to be less permissive than the ideal-cipher model required to study the Davies-Meyer construction. Knudsen and Rijmen mention that this approach could be used to analyze hash function constructions, but the security model is not formally defined, seems to be hard to formalize, and to take into account for a security proof.

Even if there is no security model well adapted to study this alternate construction mode, it is particularly interesting since the assumption on the blockcipher seems to be more realistic. We refer to *permutation* based hash functions to precise that we do not use the flexibility of having many permutations using a blockcipher. In the constructions we are interested in, we only require one or two permutations to behave as random permutations, thus the probability to have a weak key is low and cannot be used by the adversary. Finally, another practical advantage of such constructions is that the key schedule of some blockciphers is more costly than the encryption processes and so avoiding the key schedule algorithm is interesting in term of speed and in term of space for hardware implementation.

1.1 Related Work

At FSE 2005, Knudsen [6] proposed a design for a compression function using only *one* permutation and a particular instance called SMASH. Soon after, Pramstaller *et al.* [10] broke it in collision very efficiently and Lamberger *et al.* [9] broke it in second preimage. That is why Knudsen proposed two tweaks to avoid the attacks, so that the expected complexity of any collision attack is still $\mathcal{O}(2^{n/2})$.

The security of 1-permutation based hash functions has been studied at Eurocrypt 2005 by Black *et al.* [4]. They show a very interesting impossibility result: in the ideal-cipher model, the number of queries to the permutation required to attack in collision the hash function is very low, linear in the bitsize of the input/output permutation. This result seems to rule out the construction of compression function using *one* permutation. However, it has one drawback which is very important in practice: even if the number of queries is low, the overall time complexity of the attack presented is very high, namely $O(n2^n)$. Therefore, in a computational model which would take into accounts the time or space complexity of the attack, such a construction could be possible. That is why the result of [4] does not completely rules out the construction of hash function based on one permutation such as SMASH. Finally, in the same vein, Steinberger and Rogaway at Eurocrypt'08 extend this result for many permutations against collision and preimage attacks. The main result interesting for us, is that with two permutations the preimage and collision resistance cannot be proved if more than $O(2^{n/2})$ queries are made to the permutations.

1.2 Our Results

In this paper, we first exhibit a new collision attack against the two tweaked versions of SMASH with complexity in time and memory of order $O(n2^{n/3})$, generating a 2-block collision. For the first tweak version, the attack can be improved and the complexity reduced to approximately 2^{32} for $n = 256$. To avoid our attack, we propose to replace one special operation, namely the multiplication by a constant in an extension field of GF(2), by a strong permutation. This modification has already been proposed by Thomsen [13], but has never been analyzed. We prove that a collision attack against this new scheme requires at least $2^{n/4}$ queries to the permutations. In order to better evaluate its collision resistance in term of number of queries, we devise an attack that requires $2^{3n/8}$ queries but needs $O(2^{3n/4})$ in time and works only if the Merkle-Damgård strengthening is not used. Finally, we prove that the number of queries required to attack the preimage is at least of $2^{n/2}$. Note that this latter bound is optimal according to Steinberger and Rogaway attack. Therefore our construction has also a theoretical interest since it is the first 2-permutation based hash function provably collision and preimage resistant. It gives a lower bound for the best collision resistance that we can obtain with such a construction and proves that the best preimage collision resistance of these schemes is in $\Theta(2^{n/2})$ queries.

Remark that, even if we are not able to prove better bounds, this does not say that our function is weak since we are not aware of an attack requiring less than the birthday attack for collision if the Merkle-Damgård strengthening used. For the preimage, since the attack of Steinberger and Rogaway requires $O(2^n)$ time complexity, we propose an attack requiring $O(2^{n/2})$ time complexity for the compression function and $O(2^{3n/4})$ for the full hash function.

1.3 Organization of the Paper

In section 2, we recall the security model and the designs of SMASH and of our generalization. We propose our collision attack on SMASH in section 3 and study the resistance of our new hash function against collision attacks in section 4. Finally, in section 5, we study the resistance against preimage of our construction.

2 Construction and Security Model

2.1 Security Model

The ideal-cipher model. To model blockciphers, we use the ideal-cipher model introduced by Shannon. In this model, the adversary is not computationally limited and the blockcipher is viewed as a family of functions $E\colon \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ such that for each k, $E(k,\cdot)$ is a permutation on $\{0,1\}^n$. For every key k, $E(k,\cdot)$ is chosen uniformly at random in the set of all permutations on n bits. This implies that, for the adversary, for each key $k \in \{0,1\}^\kappa$, $E(k,\cdot)$ is a random and independent permutation.

The adversary A is given access to the oracles E and E^{-1}, which is denoted by $A^{E,E^{-1}}$: it can ask at most Q oracle queries to either E or E^{-1} and the answer of a query (K_i, X_i) for E is $Y_i = E(K_i, X_i)$ and the answer of a query (K_i, Y_i) for E^{-1} is $X_i = E^{-1}(K_i, Y_i)$.

Remarks on the security model. In our 2-permutation based construction, the keys k_1 and k_2 chosen for the construction are public and given to the adversary. As the permutations $E(k, \cdot)$ for $k \neq k_1, k_2$ are independent of $E(k_1, \cdot)$ and $E(k_2, \cdot)$, we assume *w.l.o.g* that the adversary does not ask oracle queries (k, x) to E or (k, y) to E^{-1} with $k \neq k_1, k_2$. For the sake of simplicity we denote $\pi_1 = E(k_1, \cdot)$ and $\pi_2 = E(k_2, \cdot)$ and give oracle access to π_1, π_1^{-1}, π_2 and π_2^{-1}. Note that in this case we do not lean upon the whole power of the ideal-cipher model, we only require that π_1 and π_2 were chosen independently and uniformly at random in the set of all permutations. We do not use the fact that for every $k \neq k_1, k_2$, $E(k, \cdot)$ is a permutation chosen uniformly at random in the set of all the permutations.

Collision resistance. If H is a hash function, the goal of the adversary is to break the collision resistance of H, that is to find two different messages (M, M') such that $H(M) = H(M')$. The ability of the adversary to break H collision resistance is denoted $\mathsf{adv}_H^{\mathsf{Coll}}(A)$ and is equal to:

$$\Pr\left[H(M) = H(M') \wedge M \neq M' | A^{E,E^{-1}} \Rightarrow (M, M')\right]$$

The probability is taken over the random coins of A and over all the possible blockcipher E where E is generated as specified above. The notation $A^{E,E^{-1}} \Rightarrow (M, M')$ means that A, after at most Q queries to E or E^{-1}, outputs (M, M'). We denote by $\mathsf{adv}_H^{\mathsf{Coll}}(Q)$ the maximum of $\mathsf{adv}_H^{\mathsf{Coll}}(A)$ over all the adversaries A which can make at most Q queries.

Assumptions. We assume that the adversary does not ask a query for which it already knows the answer; namely, it does not ask the same query twice or if it asks (k, x) to E, which returns y, it does not ask (k, y) to E^{-1}, and vice versa. Furthermore, we assume that when an adversary outputs (M, M'), it has already computed $H(M)$ and $H(M')$, *i.e.* it has already made all the oracle queries required to compute $H(M)$ and $H(M')$.

Measures of the complexity. There are two classical ways to measure the complexity of the attack. On one hand, one can say that this complexity is equal to the time complexity of the adversary. This is the complexity we are interested in, in practice, and this is the complexity that Knudsen consider in his paper about SMASH [6] and that we consider in our attack of SMASH. We refer to this complexity as the *practical complexity.* On the other hand, the attack complexity can be measured by the number of queries made to the oracles. This is the complexity oftenly used in proofs [11,5], or on the contrary to show that proofs cannot be established [4,12]. We use this complexity in our security proofs. It is refered in the following as the *query complexity.* Note that the practical complexity is always greater than the query complexity.

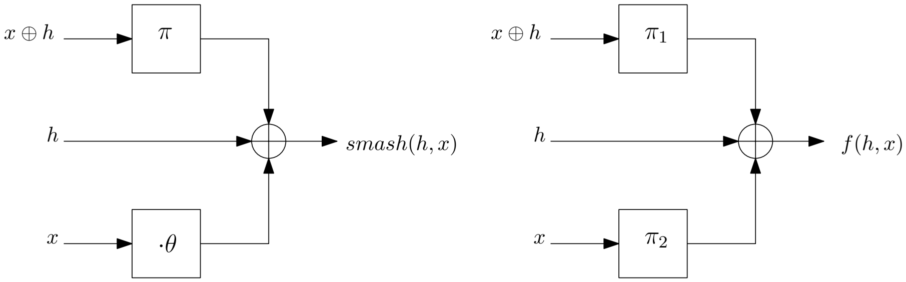

Fig. 1. SMASH compression function and our 2-permutation based compression function

2.2 SMASH Construction and Generalization

In this subsection, we introduce successively the original operating mode of SMASH, the modifications proposed by Knudsen and our new construction which is a generalization of the SMASH design.

Smash. Firstly we present the original version of SMASH. Let $\pi = E(0^n, \cdot)$ be a random permutations, $IV \in \{0,1\}^n$ be a fixed string, $\theta \neq 0, 1$ be a fixed element of $GF(2^n)$, the finite field of 2^n elements, and $smash\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be the function defined by:

$$smash^{\pi}(h,x) = \pi(h \oplus x) \oplus h \oplus \theta \cdot x,$$

where $\cdot$ denotes the multiplication in $GF(2^n)$.

Given (IV, π, θ), the hash of a message $x = (x_1, \ldots, x_\ell) \in \{0,1\}^{n \cdot \ell}$ is given by $SMASH(x) = h_{\ell+1}$ where $h_0 = \pi(IV) \oplus IV = smash^{\pi}(IV, 0^n)$, $h_k = smash^{\pi}(h_{k-1}, x_k)$, for all $1 \leq k \leq \ell$, and $h_{\ell+1} = \pi(h_\ell) \oplus h_\ell = smash^{\pi}(h_\ell, 0^n)$.

Tweaked Versions of Smash. After the attack of [10], it has been proposed two ways to modify the scheme [10,6], namely: "One is to use different permutations π for every iteration. Another is to use a secure compression function $(\ldots)$ after the processing of every t blocks of the message for, say $t = 8$ or $t = 16$".

We call the modification which consists in using a different permutation for every iteration, the *first modification* and the modification which consists in using a secure compression function (as $\pi(h) \oplus h$ for example) after the processing of every t blocks of the message, the *second modification*.

Our Generalized Construction. Let $IV \in \{0,1\}^n$ be a fixed string, $\pi_1 = E(0^n, \cdot)$ and $\pi_2 = E(1^n, \cdot)$ be two random permutations, and $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be the function defined by:

$$f(h,x) = \pi_1(x) \oplus \pi_2(x \oplus h) \oplus h$$

Given (IV, π_1, π_2), the hash of a message $x = (x_1, \ldots, x_\ell) \in \{0,1\}^{n \cdot \ell}$ is $H(x) = h_\ell$ where $h_0 = IV$, $h_k = f(h_{k-1}, x_k)$ for all $1 \leq k \leq \ell$.

Padding. The constructions introduced before require that the message length is a multiple of a fixed integer which depends on the block size. To extend this construction to arbitrary length messages, one can add an injective padding to the message, such as the classical padding proposed for SMASH: add a '1' and as many '0' as required. In SMASH, it is also required to add the so-called Merkle-Damgård strenghtening, that is to concatenate the encoded length of the message at the end of the message. We also advice to add the Merkle-Damgård strenghtening for our construction, since, even if the security proof we are able to establish does not require it, the best known collision attacks against the construction without strengthening are strictly more efficient than the best known collision attacks against the construction with the strengthening.

3 A Collision Attack against All Versions of SMASH

In this section, we present a collision attack against SMASH in $\mathcal{O}(n2^{n/3})$. Since it generates a 2-block collision, it can be mounted against the two modifications of SMASH, as long as $t \geq 3$ (we remind that t denotes the number of iterations using the classical SMASH compression function before the use of an alternate secure compression function). Then we present an improvement of the attack which can be used to reduce the complexity of the attack. It can be applied against the first modification and then the attack generates two $2^{\sqrt{n}-1}$-block long collision messages and has a practical complexity of $\mathcal{O}(2^{2\sqrt{n}})$. This means that for $n = 256$, there is an attack in $c \cdot 2^{32}$ where c is a small constant. It also can be applied against the second modification if $t = 8$ or $t = 16$, but its impact is more limited.

3.1 Generic Attack

This subsection describes an attack against the collision resistance of the two modifications of SMASH with practical complexity of $\mathcal{O}(n2^{n/3})$. It generates a 2-block collision.

Note that the generic collision attack presented in [4] by Black *et al.* also applies to SMASH used with the first modification and finds a collision with a query complexity of at most $\mathcal{O}(2(n+1))$ but a practical complexity greater than $\mathcal{O}(2^n)$. Therefore, this attack does not negate the security level expected by Knudsen [6], namely a practical security of $\mathcal{O}(2^{n/2})$. The attack presented in [10] by Pramstaller *et al.* is very efficient against the original version of SMASH, but as they precise in their paper, it does not apply to the two modifications.

In the following, we use the notations already introduced in subsection 2.2. Let π and π' be the two permutations used respectively in the first and in the second iteration. Let (α_1, β_1) and (α'_1, β'_1) be 2 pairs such that $\pi(\alpha_1) = \beta_1$ and $\pi'(\alpha'_1) = \beta'_1$. Let us define $\gamma_1 = \beta_1 \oplus \theta \cdot \alpha_1$, $\gamma'_1 = \beta'_1 \oplus \theta \cdot \alpha'_1$, $x_1 = \alpha_1 \oplus h_0$, $h_1 = smash^{\pi}(h_0, x_1) = \beta_1 \oplus \theta \cdot \alpha_1 \oplus (\theta + 1) \cdot h_0$, and $x'_1 = \alpha'_1 \oplus h_1$. Consequently, for $h_2 = smash^{\pi'}(h_1, x'_1)$, we get:

$$h_2 = \gamma'_1 \oplus (\theta + 1) \cdot \gamma_1 \oplus (\theta + 1)^2 \cdot h_0.$$

Let (α_2, β_2) and (α'_2, β'_2) be 2 other pairs such that $\pi(\alpha_2) = \beta_2$ and $\pi'(\alpha'_2) = \beta'_2$. Let us define similarly as above $\gamma_2 = \beta_2 \oplus \theta \cdot \alpha_2$, $\gamma'_2 = \beta'_2 \oplus \theta \cdot \alpha'_2$, $x_2 = \alpha_2 \oplus h_0$, $h'_1 = smash^{\pi}(h_0, x_2) = \beta_2 \oplus \theta \cdot \alpha_2 \oplus (\theta + 1) \cdot h_0$ and $x'_2 = \alpha'_2 \oplus h'_1$. For $h'_2 = smash^{\pi'}(h'_1, x'_2)$, we get:

$$h'_2 = \gamma'_2 \oplus (\theta + 1) \cdot \gamma_2 \oplus (\theta + 1)^2 \cdot h_0.$$

First, notice that if $h_2 = h'_2$, then $SMASH(x_1, x'_1) = SMASH(x_2, x'_2)$. We have $h_2 = h'_2$ if and only if $\gamma'_1 \oplus (\theta + 1) \cdot \gamma_1$ equals $\gamma'_2 \oplus (\theta + 1) \cdot \gamma_2$, which is equivalent to:

$$(\theta + 1) \cdot \gamma_1 \oplus (\theta + 1) \cdot \gamma_2 \oplus \gamma'_1 \oplus \gamma'_2 = 0. \tag{1}$$

The attack can be easily deduced from this relation.

Let us makes $2q$ queries to π to generate 2 sequences with q elements $(\alpha_{1,i}, \beta_{1,i})$ and $(\alpha_{2,i}, \beta_{2,i})$ and $2q$ queries to π' to generate 2 sequences with q elements $(\alpha'_{1,i}, \beta'_{1,i})$ and $(\alpha'_{2,i}, \beta'_{2,i})$. Let us compute the associated $\gamma_{j,i} = \beta_{j,i} \oplus \theta \cdot \alpha_{j,i}$ and $\gamma'_{j,i} = \beta'_{j,i} \oplus \theta \cdot \alpha'_{j,i}$, for $j = 1, 2$ and $1 \leq i \leq q$.

If $q = 2^{n/4}$, the birthday paradox says that with high probability there exists a quadruple $(\gamma_{1,a}, \gamma'_{1,b}, \gamma_{2,c}, \gamma'_{2,d})$ such that equation (1) is true. However finding such a quadruple requires a time complexity of $\mathcal{O}(n2^{n/2})$. For $q = 2^{n/3}$, the algorithm presented in [14] allows to find such a quadruple in time $\mathcal{O}(n2^{n/3})$ and space $\mathcal{O}(2^{n/3})$. Therefore, using this algorithm, we can mount an attack with query complexity of $\mathcal{O}(2^{n/3})$ and practical complexity of $\mathcal{O}(n2^{n/3})$ which is much smaller than the practical complexity of $\mathcal{O}(2^{n/2})$ that one could expect.

3.2 Improvements of the Attack

The improvement presented in this subsection comes from the generalization presented in [14] of the 4-list algorithm. The more lists there are, the smaller the practical complexity is. The main drawback of this improvement is that it generates longer colliding messages and therefore cannot be used completely against the second modification.

Let assume that instead of searching for 2-block colliding messages, we are searching for 3-block colliding messages. Using the same notations as above, let us introduce π'' the permutation used in the third iteration and (α''_1, β''_1) and (α''_2, β''_2) two pairs such that $\pi''(\alpha''_1) = \beta''_1$ and $\pi''(\alpha''_2) = \beta''_2$. If we define similarly as above $x''_1 = \alpha''_1 \oplus h_2$ and $x''_2 = \alpha''_2 \oplus h'_2$, and generalize previous notations, we have that

$$h_3 = \gamma''_1 \oplus (\theta + 1) \cdot \gamma'_1 \oplus (\theta + 1)^2 \cdot \gamma_1 \oplus (\theta + 1)^3 \cdot h_0$$
$$h'_3 = \gamma''_2 \oplus (\theta + 2) \cdot \gamma'_2 \oplus (\theta + 2)^2 \cdot \gamma_2 \oplus (\theta + 1)^3 \cdot h_0$$

Therefore, $h_3 = h'_3$ if and only if $(\theta+1)^2 \cdot (\gamma_1 \oplus \gamma_2) \oplus (\theta+1) \cdot (\gamma'_1 \oplus \gamma'_2) \oplus \gamma''_1 \oplus \gamma''_2 = 0$.

This leads to an attack which generates 6 lists and tries to find one element in every list such that the xor of theses elements is equal to 0. This can be generalized to k-block long messages. We can show that $h_k = h'_k$ if and only if:

$$\bigoplus_{i=0}^{k} (\theta + 1)^{k-i} \cdot (\gamma_1^{(i)} \oplus \gamma_2^{(i)}) = 0. \tag{2}$$

The algorithm in [14] finds such a $2k$-tuple in time $\mathcal{O}(k \cdot 2^{n/(1+\log_2(2k))})$ and requires $2k$ lists of size $\mathcal{O}(2^{n/(1+\log_2(2k))})$, therefore it requires to make $\mathcal{O}(k \cdot 2^{n/(1+\log_2(2k))})$ queries to generate all these lists. The complexity of the attack is optimal for $2k = 2^{\sqrt{n}}$ and in this case the practical complexity is equal to $\mathcal{O}(2^{2\sqrt{n}})$.

This improvement can be applied for all values of k when the first modification is used and therefore this version of SMASH can be attack in $\mathcal{O}(2^{2\sqrt{n}})$, generating messages of $2^{\sqrt{n}-1}$ blocks. For $n = 256$, this means a complexity of $c \cdot 2^{32}$ for a small constant c and messages of 2^{15} 256-bit blocks, that is of 1 Mo.

However, it can be applied only for $k \leq t - 1$ when the second modification is used. Therefore against this modification, the improved attack has a practical complexity of $\mathcal{O}(t \cdot 2^{n/(2+\log_2(t-1))})$, that is $\mathcal{O}(2^{n/4})$ and $\mathcal{O}(2^{n/5})$ for $t = 8$ and $t = 16$ respectively, as proposed by Knudsen [6] (we remind that t denotes the number of iterations using the classical SMASH compression function before the use of an alternate secure compression function). For $n = 256$, this gives a complexity of approximately 2^{64} and 2^{52}.

4 Collision Resistance of the Generalized Design

Now, we examine the collision resistance of the generalized version we propose. Firstly, we prove that a collision attack requires at least $\Omega(2^{n/4})$ queries to succeed with good probability. Secondly, we give a collision attack against our scheme with a query complexity of $\mathcal{O}(2^{3n/8})$, but a practical complexity of $\mathcal{O}(2^{3n/4})$. Most often this attack generates two messages of different length and therefore does not work anymore if the Merkle-Damgård strengthening is used. In this latter case, the best attack we have against our scheme is the birthday paradox attack with $\mathcal{O}(2^{n/2})$ queries and a practical complexity of $\mathcal{O}(n2^{n/2})$.

4.1 Security Proof

The attack presented in [4] shows in particular that one cannot expect to *prove* the collision resistance of SMASH if more than $\mathcal{O}(n)$ queries are made. On the contrary, we prove here that if we replace the multiplication by θ by a strong permutation (modelized by an ideal cipher), then one can prove that at least $\Omega(2^{n/4})$ queries are required to break collision resistance, and therefore that such an attack has a practical complexity greater than $2^{n/4}$. This proof is valid even if the Merkle-Damgård strengthening is not used.

Theorem 1. *Let A be a computationally unbounded adversary which makes at most Q queries. Its advantage in breaking H collision resistance is upper bounded by:*

$$\mathsf{adv}_H^{\mathsf{Coll}}(A) \leq \frac{2Q^4}{2^n}.$$

Proof. A collision adversary is allowed to make at most Q queries to either π_1, π_2, π_1^{-1}, or π_2^{-1}. We show that the probability that the adversary finds a collision

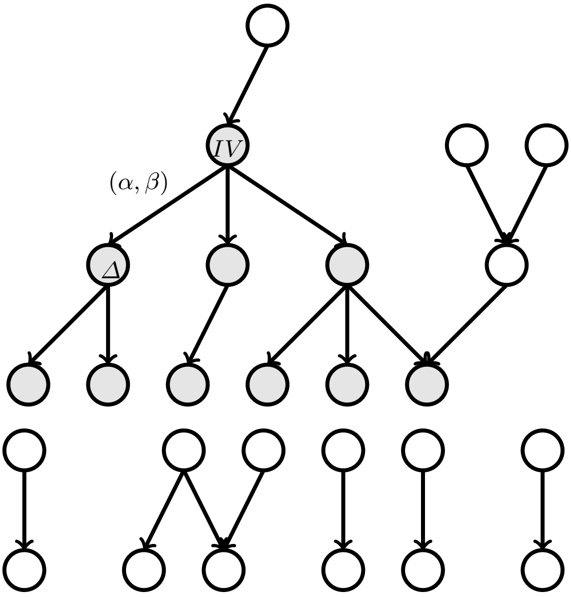

Fig. 2. An example of graph. In gray is the tree T.

for H is upper bounded by $Q^4/2^n$. The permutations π_1, π_2 and the initial value IV are chosen randomly.

The graph construction. First, we introduce the following graph construction. Let $R_1 = \{(\alpha_i, \beta_i)_{1 \le i \le q_1}\}$ be q_1 pairs such that $\pi_1(\alpha_i) = \beta_i$ and $R_2 = \{(\alpha'_j, \beta'_j)_{1 \le j \le q_2}\}$ be q_2 pairs such that $\pi_2(\alpha'_j) = \beta'_j$. We define $\Delta_{i,j} = \alpha_i \oplus \alpha'_j$ and $\tilde{\Delta}_{i,j} = \beta_i \oplus \beta'_j \oplus \alpha_i \oplus \alpha'_j$ for $1 \le i \le q_1$ and $1 \le j \le q_2$. We construct a labelled directed graph $G = (V, E)$. The set of vertices V contains the bit strings $\Delta_{i,j}$, $\tilde{\Delta}_{i,j}$ and IV (that is at most $2q_1 \cdot q_2 + 1$ nodes). The set of edges E contains the directed edges $(\Delta_{i,j}, \tilde{\Delta}_{i,j})$ labelled with (α_i, β_i) denoted $\Big((\Delta_{i,j}, \tilde{\Delta}_{i,j}), \alpha_i, \beta_i\Big)$ (there are exactly $q_1 \cdot q_2$ labelled directed edges, possibly several edges between the same pair of nodes).

We define a path in the graph G as a sequence of edges $p = (e_1, \ldots, e_\ell)$ such that for each of its edge e_i, $1 \le i \le \ell - 1$ the output vertex is equal to the input vertex of e_{i+1}. Let us denote $\Delta \overset{p}{\rightsquigarrow} \Delta'$ which means that either $\Delta = \Delta'$ (and p is empty) or there exists a path $p = (e_1, \ldots, e_\ell)$ for which the input vertex of e_1 is Δ and the output vertex of e_ℓ is Δ'.

Correspondence between the hash function and the graph construction. A message $x = (x_1, \ldots, x_\ell)$ is said to be *valid* if one can compute its digest value thanks to the already made requests, that is if and only if: $h_0 = IV$ and for every $k \ge 1$, $(x_k, \pi_1(x_k)) \in R_1$ and $(x_k \oplus h_{k-1}, \pi_2(x_k \oplus h_{k-1})) \in R_2$, with $h_k = \pi_1(x_k) \oplus \pi_2(x_k \oplus h_{k-1}) \oplus h_{k-1}$. Let us denote by M the set of all the valid messages. Let P be the set of all non-empty paths in G with IV as input node, that is $P = \{p \neq \emptyset \,|\, \exists\, \varDelta \in V, IV \overset{p}{\rightsquigarrow} \varDelta\}$. We now show that there is a bijection between P and M.

Let $p = (e_1, \ldots, e_\ell)$ be a non-empty path from IV to a node $\varDelta$. For this path p we construct a message $x = (x_1, \ldots, x_\ell)$, such that $H(x) = \varDelta$, where x is defined as follows. For the k^{th} edge e_k, by construction, there exists (a unique) (i_k, j_k) such that $e_k = \Big((\varDelta_{i_k,j_k}, \tilde{\varDelta}_{i_k,j_k}), \alpha_{i_k}, \beta_{i_k}\Big)$, and we define $x_k = \alpha_{i_k}$. Using the same notations as in the definition of H one can easily check that $h_0 = IV = \varDelta_{i_1,j_1}$, and for all other $1 \leq k \leq \ell$, $h_k = \varDelta_{i_{k+1},j_{k+1}} = \tilde{\varDelta}_{i_k,j_k}$:

$$\begin{aligned} h_k &= f(h_{k-1}, x_k) = \pi_1(x_k) \oplus \pi_2(x_k \oplus h_{k-1}) \oplus h_{k-1} \\ &= \pi_1(\alpha_{i_k}) \oplus \pi_2(\alpha_{i_k} \oplus \varDelta_{i_k,j_k}) \oplus \varDelta_{i_k,j_k} = \pi_1(\alpha_{i_k}) \oplus \pi_2(\alpha'_{j_k}) \oplus \alpha_{i_k} \oplus \alpha'_{j_k} \\ &= \beta_{i_k} \oplus \beta'_{j_k} \oplus \alpha_{i_k} \oplus \alpha'_{j_k} = \tilde{\varDelta}_{i_k,j_k} = \varDelta_{i_{k+1},j_{k+1}}. \end{aligned}$$

Therefore x is valid and $H(x) = h_\ell = \tilde{\varDelta}_{i_\ell,j_\ell} = \varDelta$. We say that p induces the message x. One can check easily that if $p \neq p'$ induce respectively x and x', then $x \neq x'$.

Conversely, let $x = (x_1, \ldots, x_\ell)$ be a valid message and p be the path defined as $p = (e_1, \ldots, e_\ell)$ with $e_k = ((h_{k-1}, h_k), x_k, \pi_1(x_k))$ (we remind that $h_0 = IV$ and $h_k = f(h_{k-1}, x_k)$). The path p is clearly in P. We say that x induces p. One can check easily that if $x \neq x'$ induce respectively a path p and p' in G then $p \neq p'$.

Therefore, finding two colliding messages in M is equivalent to find two paths in P with the same output nodes. We say that these two paths collide and that there is a collision in G.

Upper bound of the collision probability. Consider now the collision adversary. Let us assume that it has already made q_1 queries to π_1 or π_1^{-1} and q_2 queries to π_2 or π_2^{-1}. These queries induce two sets R_1 and R_2, and a graph G defined as above. We also introduce the following sets:

$$\begin{aligned} T &= \{\varDelta \in V \,|\, \exists\, p, IV \overset{p}{\rightsquigarrow} \varDelta\} \\ A &= \{\alpha \,|\, \exists\, 1 \leq j \leq q_2, \exists\, \varDelta \in T, \alpha'_j \oplus \varDelta = \alpha\} \\ B &= \{\gamma \,|\, \exists\, 1 \leq j \leq q_2, \exists\, \varDelta' \in V, \beta'_j \oplus \alpha'_j \oplus \varDelta' = \gamma\} \end{aligned}$$

Without loss of generality, we can assume that the adversary is ready to make a query to π_1 or π_1^{-1}. Let us denote by $(\tilde{\alpha}, \tilde{\beta} = \pi_1(\tilde{\alpha}))$ the pair induced by this query. With this query the graph G expands, new edges are generated. Let us denote by $\tilde{G}$ the graph after this expansion and similarly $\tilde{T}$ the expansion of T and $\tilde{P}$ the expansion of P.

We now show that if there is a collision in $\tilde{G}$, then $\tilde{\beta} \oplus \tilde{\alpha} \in B$ and $\tilde{\alpha} \in A$ with high probability. Assume that there is a collision in $\tilde{G}$ but not in G. Let $\varDelta$ be a node in $\tilde{G}$, let p, p' be two paths in $\tilde{P}$ such that $p \neq p'$, $IV \overset{p}{\rightsquigarrow} \varDelta$ and $IV \overset{p'}{\rightsquigarrow} \varDelta$ in $\tilde{G}$. Let us denote $(IV, \varDelta_1, \ldots, \varDelta_\ell = \varDelta)$ the sequence of vertices crossed by p

in $\tilde{G}$ and $(IV, \Delta'_1, \ldots, \Delta'_m = \Delta)$ the sequence of vertices crossed by p' in $\tilde{G}$. As there is not any collision in G, then either p or p' is not in P. Let us say it is p.

Note that with high probability Δ is already in G and was not generated by the expansion. If it were not the case, then there would be $i \neq j$ such that $\Delta = \tilde{\beta} \oplus \beta'_i \oplus \alpha'_i = \tilde{\beta} \oplus \beta'_j \oplus \alpha'_j$. This implies that $\beta'_i \oplus \alpha'_i = \beta'_j \oplus \alpha'_j$. The probability that there exists such a pair (i, j) is upper bounded by $q_2^2/2^n$. Let us assume that such a pair does not exist and therefore that Δ is already in G.

Let a be the smallest integer such that there exists r suffix of p with $\Delta_a \overset{r}{\rightsquigarrow} \Delta_\ell$ in G (hence $\Delta_a \in V$), that is r exists before the expansion. Due to the previous remark, a exists and $a \leq \ell$. As $\Delta_{a-1} \notin r$, it means that the edge (Δ_{a-1}, Δ_a) is generated by the expansion, that is there exists j such that $\Delta_{a-1} = \tilde{\alpha} \oplus \alpha'_j$ and $\Delta_a = \tilde{\beta} \oplus \tilde{\alpha} \oplus \beta'_j \oplus \alpha'_j$. Therefore we have $\tilde{\beta} \oplus \tilde{\alpha} \in B$.

Similarly, let b be the greatest integer such that, there exists r' prefix of p with $IV \overset{r'}{\rightsquigarrow} \Delta_b$ in G (hence $\Delta_b \in T$). As $\Delta_{b+1} \notin r'$, it means that the edge (Δ_b, Δ_{b+1}) is generated by the expansion, that is there exists j such that $\Delta_b = \tilde{\alpha} \oplus \alpha'_j$ and $\Delta_{b+1} = \tilde{\beta} \oplus \tilde{\alpha} \oplus \beta'_j \oplus \alpha'_j$. Therefore we have $\tilde{\alpha} \in A$.

If it is π_1 which was queried by $\tilde{\alpha}$, then the collision probability is upper bounded by:

$$\frac{\#B}{2^n - q_1} \leq \frac{\#V \cdot q_2}{2^n - q} \leq \frac{2(2q_1q_2 + 1)q_2}{2^n} \leq \frac{2q^3}{3 \cdot 2^n} \leq \frac{q^3}{2^n},$$

where $q = q_1 + q_2$. The last inequality is true because the function $x \mapsto 2(2(q - x)x + 1)x$ reaches its maximum for $x \approx 2q/3$ and is smaller than $2q^3/3$ at this point. The collision probability can be similarly upper bounded by $q^3/2^n$ if it is π_1^{-1} which was queried.

Therefore, at the q^{th} iteration, the success probability is lower than $2q^3/3 \cdot 2^n + q^2/2^n$, and at the end the success probability is lower than
$\sum_{q=1}^{Q} \left(2q^3/3 \cdot 2^n + q^2/2^n\right) \leq Q^4/2^n$. □

4.2 Attacks

Now we present an attack against the entire hash function, this gives upper bounds of its collision resistance. Before that, note that the birthday paradox allows to easily construct a collision attack which succeeds with probability nearly 1 with $\mathcal{O}(2^{n/2})$ queries to π_1 and π_2 and time complexity of $\mathcal{O}(n2^{n/2})$. This attack generates two 1-block messages which collide and therefore works even if the Merkle-Damgård strengthening is used (see the full version of the paper for a description of this attack).

We present now an attack which succeeds with probability nearly 1. It is a better attack than the birthday attack since its query complexity is only equal to $2 \cdot 2^{3n/8}$, but it has a practical complexity of $\mathcal{O}(n2^{3n/4})$. Moreover, one does not control the size of the two messages generated during the attack and most probably they won't have the same size. Therefore the Merkle-Damgård strengthening allows to thwart the attack. Our analysis of this latter is heuristic and not proved.

However, we have tested the attack for several values of n up to $n = 40$ and it turned out to work well in practice.

Proposition 1. *For $Q \geq 2^{3n/8+2}$ there is a computationally unbounded collision adversary with high success probability.*

Sketch of the attack. In the sequel, first we explain how we make the queries, then we informally evaluate the expected number of messages for which we are able to compute the hash. For a precise algorithm, see the full version of the paper. Note that the following attack is inspired from the way we have proved the collision resistance : we introduce the same tree T and try to make it grow as much as possible, so that it quickly contains $2^{n/2}$ vertices.

Let α_0 and β_0 be two random n-bit strings such that $\alpha_0 \oplus \beta_0 = IV$. Let q be an integer. We generate the sequences $(\alpha_i)_{0 \leq i \leq q}$, and $(\beta_i)_{0 \leq i \leq q}$ such that for all $1 \leq i \leq q$, $\alpha_i = \pi_1(\alpha_{i-1}) \oplus \alpha_{i-1}$, and $\beta_i = \pi_2(\beta_{i-1}) \oplus \beta_{i-1}$. For all $0 \leq i, j \leq q$, let us define $\Delta_{i,j} = \alpha_i \oplus \beta_j$. Note that we make $Q = 2q$ queries to π_1 and π_2, and we generate about $(q+1)^2$ different $\Delta_{i,j}$. We generate the sequences this way, because we have the following interesting property: for all (i,j), $f(\Delta_{i,j}, \alpha_i) = \Delta_{i+1,j+1}$. Therefore, for all $1 \leq \ell \leq q$, the message $\alpha_0 \| \alpha_1 \| \ldots \| \alpha_{\ell-1}$ hashes to $\Delta_{\ell,\ell}$ and if $\Delta_{k,k} = \Delta_{i,j}$, then for all $1 \leq \ell \leq q - \max(i,j)$ the message $M_{k,i,j,\ell} = \alpha_0 \| \alpha_1 \| \ldots \| \alpha_{k-1} \| \alpha_i \| \alpha_{i+1} \| \ldots \| \alpha_{i+\ell-1}$ hashes to $\Delta_{i+\ell,j+\ell}$. Such a triplet (k,i,j) is called a *colliding triplet* and the message $M_{k,i,j,\ell}$ is a preimage of $\Delta_{i+\ell,j+\ell}$ for H.

If there are many different colliding triplets (i,j,k), so we are able to find a preimage for many different values $\Delta_{i+\ell,j+\ell}$. Let us introduce the graph $T = (V,E)$ where:

$$\begin{aligned} V &= \{\Delta_{a,b} | \exists \text{ a colliding triplet } (k,i,j) \; s.t. \; a - i = b - j \geq 0\} \\ &\quad \cup \{\Delta_{a,a}, 0 \leq a \leq q\} \\ E &= \{(\Delta_{a,b}, \Delta_{a+1,b+1}) \; s.t. \; 0 \leq a,b \leq q-1, \Delta_{a,b} \in V\}. \end{aligned}$$

Note that we are able to find a preimage for all $\Delta_{a,b} \in V$ and that T is a tree if and only if there is no collision (otherwise we are able to find a cycle in T and there are two ways to reach some $\Delta_{a,b} \in V$). Therefore, our goal is to make V grow up to a size of about $2^{n/2}$ vertices so that a collision occurs. In the following we explain informally why this happens with high probability for $q = 2^{3n/8+1}$. This analysis considers that the α_i and β_j are uniformly distributed, which is of course not the case. However, we expect that the analysis gives a good intuition of what happens.

Let us evaluate roughly the expected value of T size, denoted t. Note that T contains at least the $q+1$ vertices $\Delta_{a,a}$. If (i,j,k) is a collision triplet with $i \neq j$, then all the $\Delta_{i+\ell,j+\ell}$, with $0 \leq \ell \leq q - \max(i,j)$, are added to T. Thus, if $\mathcal{S}et = \{(i,j,k) \; s.t. \; i \neq j, i \neq k, j \neq k\}$, we have:

$$t \approx 1 + q + \sum_{(i,j,k) \in \mathcal{S}et} \mathbb{1}_{\{\Delta_{k,k} = \Delta_{i,j}\}} (q - \max(i,j)),$$

$$\text{therefore, } \mathbb{E}(t) \approx 1 + q + \sum_{(i,j,k) \in \mathcal{S}et} \Pr\left[\Delta_{k,k} = \Delta_{i,j}\right] (q - \max(i,j)).$$

n	Q	number of experiments	size of T	percentage of success
36	2^{15}	10000	$2^{18} \leq \cdot \leq 2^{19}$	52%
40	$2^{16.5}$	1000	$2^{20} \leq \cdot \leq 2^{21}$	59%

Fig. 3. Experimental results

where $\mathbb{1}$ denotes the characteristic function. If the α_i and β_j were uniformly distributed (which is *not* the case) then we would have that $\Pr[\Delta_{k,k} = \Delta_{i,j}] = 1/2^n$ and therefore that

$$\begin{aligned}\mathbb{E}(t) &\approx 1 + q + \frac{1}{2^n} \sum_{(i,j,k)\in Set} (q - \max(i,j)) \\ &= 1 + q + \frac{q(q+1)(q-1)(q-2)}{3 \cdot 2^n} \approx \frac{q^4}{3 \cdot 2^n}.\end{aligned}$$

We can conclude that for $q = 2^{3n/8}$, we can expect that T contains more than $2^{n/2}$ vertices and, in this case, hope that the birthday paradox applies here so that two of these vertices collide. As already stated, if there is such a collision the attack is finished, we are able to find two messages which collide for H. □

Complexity of the attack and experimental results. The precise algorithm is described in the full version of the paper. The attack requires $\mathcal{O}(2^{3n/8})$ queries to π_1 and π_2, $\mathcal{O}(2^{n/2})$ in space (to store T) and $\mathcal{O}(n2^{3n/4})$ in time (because we have to search for *all* the colliding triplets (i, j, k) with $1 \leq i, j, k \leq 2^{3n/8}$, that is all the triplets (i, j, k) such that $\Delta_{k,k} = \alpha_i \oplus \beta_j$).

We have run several tests for n equals 36 and 40. For that, we have used the blockcipher RC5 with two random keys and with a random IV. The results are summarized in figure 3. It appears that for $Q = 2\sqrt{2} \cdot 2^{3n/8}$, in all experiments the tree T contains between $2^{n/2}$ and $2^{n/2+1}$ vertices and a collision is found at least half the time. This validates our heuristic analysis of the attack.

Note. We have studied some other constructions of a compression function using only two permutations and some "xor". Some lead to hash functions which are trivially breakable, for all the others a variant of this attack could be applied (sometimes this variant is tricky and requires to make oracle queries to π_1^{-1} or π_2^{-1}).

5 Security against Preimage of the Generalized Construction

5.1 Security Proof

In this section we prove that the preimage resistance of our construction is provably in $\mathcal{O}(2^{n/2})$ queries. Note that in the design of the compression function, we have added a feed-forward (more precisely, we xor the chaining value to the output of the two permutations) exclusively in order to prevent trivial preimage attacks against the compression function (removing this feed-forward does not alter the collision resistance). Thus, the compression function is provably collision resistant up to $\mathcal{O}(2^{n/2})$ queries. That is what we show in the following. Since finding a preimage for the whole hash function implies finding a preimage for the compression function, this implies that the whole hash function is provably preimage resistant as long as less than $\mathcal{O}(2^{n/2})$ queries have been made.

Proposition 2. *Let A be a computationally unbounded adversary which makes at most Q queries, its advantage in breaking f preimage resistance is upper bounded by $Q^2/2^n$.*

Proof. Let α and α' be two queries made respectively to π_1 and π_2 and let β and β' be the respective answers (that is we have $\beta = \pi_1(\alpha)$ and $\beta' = \pi_2(\alpha')$). Let x be the value for which a preimage is searched. We have $\Pr[x = \alpha \oplus \beta \oplus \alpha' \oplus \beta'] = \sum_y \Pr[\beta = x \oplus \alpha \oplus \alpha' \oplus y] \Pr[\beta' = y] = 1/2^n$. The result is the same if π^{-1} is queried by β or if π^{-2} is queried by β'. Therefore, if A makes q_1 queries to π_1 or π_1^{-1} and q_2 queries to π_2 or π_1^{-2} (such that $q_1 + q_2 = Q$) and obtains the pairs (α_i, β_i) and (α'_j, β'_j) respectively, the union bound says that the probability that there exists a pair (i, j) such that $x = \alpha_i \oplus \beta_i \oplus \alpha'_j \oplus \beta'_j$ is upper bounded by $q_1 q_2/2^n$, and thus by $Q^2/2^n$. □

5.2 Optimality of the Proof and Attacks

In [12], Rogaway and Steinberger present a generic $\mathcal{O}(n2^{n/2})$ preimage attack against any 2-permutation based hash function. This means, as already stated, that our construction reaches the best security level against preimage that we can expect, namely $\mathcal{O}(2^{n/2})$ queries; in this sense, the construction is optimal.

Besides, the attack of [12] against the whole hash function requires $\mathcal{O}(n2^{n/2})$ queries, but the exact practical complexity is not established in general. However, in our case, its practical complexity seems greater than 2^n. This leads us to wonder what is the attack with the lowest practical complexity. Since finding a preimage for the compression function requires $\mathcal{O}(n2^{n/2})$ in time, the Lai and Massey attack [8] can be used. This attack is an unbalanced meet-in-the-middle attack: we compute $2^{n/4}$ preimages, hash $2^{3n/4}$ messages and meet in the middle using the birthday paradox. This requires to make $\mathcal{O}(2^{3n/4})$ queries and to make $\mathcal{O}(n2^{3n/4})$ computations. This is still greater than $\mathcal{O}(n2^{n/2})$ and it is an open problem to decrease the practical complexity of a preimage attack against the whole hash function.

Acknowledgment. The authors would like to thank Lars Knudsen for his useful comments. This work has been partially supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT, and the French RNRT/ANR SAPHIR Project.

References

1. Bellare, M., Krovetz, T., Rogaway, P.: Luby-rackoff backwards: Increasing security by making block ciphers non-invertible. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 266–280. Springer, Heidelberg (1998)

2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Radiogatùn, a belt-and-mill hash function. In: ECRYPT Hash Workshop (2007)
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
4. Black, J.A., Cochran, M., Shrimpton, T.: On the impossibility of highly-efficient blockcipher-based hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 526–541. Springer, Heidelberg (2005)
5. Black, J.A., Rogaway, P., Shrimpton, T.: Black-box analysis of the block-cipher-based hash-function constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
6. Knudsen, L.R.: SMASH - A cryptographic hash function. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 228–242. Springer, Heidelberg (2005)
7. Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
8. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
9. Lamberger, M., Pramstaller, N., Rechberger, C., Rijmen, V.: Second preimages for SMASH. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 101–111. Springer, Heidelberg (2006)
10. Pramstaller, N., Rechberger, C., Rijmen, V.: Breaking a new hash function design strategy called SMASH. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 233–244. Springer, Heidelberg (2006)
11. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: A synthetic approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
12. Rogaway, P., Steinberger, J.P.: Security/Efficiency tradeoffs for permutation-based hashing. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 220–236. Springer, Heidelberg (2008)
13. Thomsen, S.S.: Cryptographic Hash Functions. PhD thesis, Technical University of Denmark (2005)
14. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, p. 288. Springer, Heidelberg (2002)