Provable Chosen-Target-Forced-Midfix Preimage Resistance

Elena Andreeva and Bart Mennink

Dept. Electrical Engineering, ESAT/COSIC and IBBT
Katholieke Universiteit Leuven, Belgium
{elena.andreeva,bart.mennink}@esat.kuleuven.be

Abstract. This paper deals with definitional aspects of the herding attack of Kelsey and Kohno, and investigates the provable security of several hash functions against herding attacks.

Firstly, we define the notion of chosen-target-forced-midfix (CTFM) as a generalization of the classical herding (chosen-target-forced-prefix) attack to the cases where the challenge message is not only a prefix but may appear at any place in the preimage. Additionally, we identify four variants of the CTFM notion in the setting where salts are explicit input parameters to the hash function. Our results show that including salts without weakening the compression function does not add up to the CTFM security of the hash function.

Our second and main technical result is a proof of CTFM security of the classical Merkle-Damgård construction. The proof demonstrates in the ideal model that the herding attack of Kelsey and Kohno is optimal (asymptotically) and no attack with lower complexity exists. Our security analysis applies to a wide class of narrow-pipe Merkle-Damgård based iterative hash functions, including enveloped Merkle-Damgård, Merkle-Damgård with permutation, HAIFA, zipper hash and hash-twice hash functions. To our knowledge, this is the first positive result in this field.

Finally, having excluded salts from the possible tool set for improving narrow-pipe designs' CTFM resistance, we resort to various message modification techniques. Our findings, however, result in the negative and we demonstrate CTFM attacks with complexity of the same order as the Merkle-Damgård herding attack on a broad class of narrow-pipe schemes with specific message modifications.

Keywords: Herding attack; Chosen-target-forced-midfix; Provable security; Merkle-Damgård.

1 Introduction

Hash functions are an important cryptographic primitive finding numerous applications. Most commonly, hash functions are designed from a fixed input length compression function to accommodate messages of arbitrary length. The most common domain extender is the Merkle-Damgård (MD) iteration [8,16], which has long been believed to be a secure design choice due to its collision security reduction. Recently, however, several results cast doubt on its security with

A. Miri and S. Vaudenay (Eds.): SAC 2011, LNCS 7118, pp. 37–54, 2012.

respect to other properties. The MD design was showed to not preserve either second preimage or preimage properties [3]. Moreover, the indifferentiability attack of Coron et al. [7], the multicollision attack of Joux [12] and the herding attack of Kelsey and Kohno [13] exposed various weaknesses of the MD design.

The herding attack of Kelsey and Kohno, also known as the chosen-target-forced-prefix (CTFP) attack, considers an adversary that commits to a hash value y for a message that is not entirely under his control. The adversary then demonstrates abilities to incorporate an unknown challenge prefix as part of the original preimage corresponding to the committed value y. While for a random oracle the complexity of such an attack is $\Theta(2^n)$ compression function calls for y of length n bits, the herding attack on Merkle-Damgård takes about $\sqrt{n}2^{2n/3}$ compression function executions for a preimage message of length $O(n)$ as demonstrated by Kelsey and Kohno [13]. A more precise attack bound was obtained by Blackburn et al. [6].

Several other hash functions have been analyzed with respect to resistance to herding attacks. In [2], Andreeva et al. showed applications of the herding attack to dither hash functions, and in [1] they generalized the herding attack of [13] to several multi-pass domain extenders, such as the zipper hash and the hash twice design. Gauravaram et al. [9,10] concluded insecurity against herding attacks for the MD designs where an XOR tweak is used in the final message block.

While this topic has generated significant interest, many important questions still remain unanswered. All research so far focused on negative results, being generalized herding attacks on hash function designs, but it is not known whether one can launch herding attacks with further improved complexity against the MD design. The task becomes additionally complicated by the lack of formal security definitions for herding to accommodate the objectives of a proof based approach. Apart from wide-pipe designs, no scheme known yet is secure against herding attacks, nor is it clear how to improve the MD design without enlarging its state size. Some of the possible directions are to either use randomization (salts) or to apply certain message modification techniques.

Our Contributions. In this work we address the aforementioned open problems. Firstly, we develop a formal definition for security against herding attacks. Neven et al. first formalize the notion of chosen-target-forced-prefix security (as the random-prefix preimage problem) in [17]. Our new notion of chosen-target-forced-midfix (CTFM) security extends his notion by enabling challenges of not only prefix type but also as midfixes and suffixes. We achieve this by giving the adversary the power to output an "order"-defining mixing function g together with his response, which fixes the challenge message at a selected by the adversary position in the preimage message.

In addition, we investigate the notion of salted CTFM. Depending on when the salt value is generated and who is in control of it, four variants of the salted notion emerge in the setting where randomness (salt) is used. One of the variants serves no practical purposes because it forces an adversary to commit to a hash

value for an unknown salt, where the salt is defined as an input parameter to the hash function. Although the other three variants are plausible from a practical perspective, we show that they do not attribute to an improved CTFM resistance. This is true for the case where the salt is added in such a way that the cryptographic strength of the compression function is not compromised on some other level, i.e. collision or preimage resistance.

Our main technical contribution is to exhibit a CTFM security proof for the MD domain extender. While until now the research in this area has been focusing on finding herding attacks against hash function designs, we are the first to provide a security upper bound for a hash function design. In more detail, we upper bound the strength of a CTFM attacker in finding in the ideal model a preimage for the MD design, and show that the herding attack described by Kelsey and Kohno is optimal (asymptotically). Using new proof techniques we prove that at least approximately $2^{2n/3}/L^{1/3}$ compression function queries are needed for a CTFM attack, where n is size of the commitment y and L is the maximal allowed length of the preimage in blocks. To the best of our knowledge, there has not been a positive result of this form before. Due to its generic nature, the new security techniques introduced in this work not only apply to the MD design, but directly carry over to a broad spectrum of domain extenders derived from MD, including strengthened MD, MD with a distinct final transformation and HAIFA [5]. Additionally, the bound implies optimality of the attacks on hash twice and the zipper hash function performed by Andreeva et al. [1].

We explore further the question of whether a simple tweak on the narrow-pipe MD construction would allow us to prove optimal CTFM security. Excluding randomness or salting from the set of available tools, we investigate tweaks that modify the message inputs by simple message modification techniques like the XOR operation. These schemes can be viewed as MD type domain extenders with a more sophisticated padding. Our findings, however, result in the negative and we demonstrate CTFM attacks on a class of schemes of this form. The attack particularly applies also to the MD with checksum design, therewith providing a simple and elegant alternative to the attack by Gauravaram et al. [10].

2 Preliminaries

By $x \xleftarrow{\$} \mathcal{X}$ we denote the uniformly random sampling of an element from a set $\mathcal{X}$. By $y \leftarrow \mathsf{A}(x)$ and $y \xleftarrow{\$} \mathsf{A}(x)$, we denote the assignment to y of the output of a deterministic and randomized algorithm A, respectively, when run on input x. For a function f, by $\mathsf{rng}(f)$ we denote its range. By *Func(n+m,n)*, for $m, n \geq 1$, we denote the set of compression functions $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. As of now, we assume $m = \Theta(n)$.

A hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ is a function that maps an arbitrarily long input to a digest of n bits, internally using a compression function $f \in$ *Func(n+m,n)*. In this work we distinguish between a general hash function design $\mathcal{H}$, and the notion of an iterated hash function which is a specific type

of hash function that executes the underlying compression function in an "iterated" way. Let $\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^m)^*$ be an injective padding function. Let $\mathsf{iv} \in \{0,1\}^n$ be a fixed initial value. We define the iterated hash function $\mathcal{IH}$ as follows:

$$\begin{aligned} \mathcal{IH}(M) = h_l, \text{ where: } & (M_1, \ldots, M_l) \leftarrow \mathsf{pad}(M); \\ & h_0 \leftarrow \mathsf{iv}; \\ & h_i \leftarrow f(h_{i-1}, M_i) \text{ for } i = 1, \ldots, l. \end{aligned} \tag{1}$$

One may include a special final transformation, but we note that this would not result in any added value against the herding attack. The general iterative principle of $\mathcal{IH}$ is followed by a wide range of existing modes of operations, including the traditional (strengthened) MD design [8,16], enveloped MD (includes final transformation) [4], MD with permutation (includes final transformation) [11], HAIFA [5] and the zipper hash design [15].

3 Chosen-Target-Forced-Midfix Preimage Resistance

An adversary for a hash function is a probabilistic algorithm with oracle access to underlying compression function f. Queries to f are stored in a query history $\mathcal{Q}$. In the remainder, we assume that $\mathcal{Q}$ always contains the queries required for the attack, and we assume that the adversary does not make trivial queries, i.e. queries to which the adversary already knows the answer in advance. In this work we consider information-theoretic adversaries only. This type of adversary has unbounded computational power, and its complexity is measured by the number of queries made to its oracles.

The goal of an attacker in the original herding attack [13] against a hash function $\mathcal{H}$ is to successfully commit to a hash digest without knowing the prefix of the preimage message in advance. In the first phase of the attack, the adversary makes an arbitrary amount of queries to f. Then, he commits to a hash digest y, and receives challenge prefix P. After a second round of compression function calls, the adversary outputs a response R such that $\mathcal{H}(P\|R) = y$. In this form the power of the adversary is limited significantly. In particular, if the hash function is defined so as to process the message blocks in reverse order, the success probability of the adversary is limited to the preimage advantage. We generalize the original attack by introducing the chosen-target-forced-midfix (CTFM) attack. In the CTFM attack, the challenge P does not necessarily need to be a prefix, but can technically occur at any place in the preimage message. The generalized attack mainly differs from the original one in the fact that the adversary not only outputs a bit string R, but also an "order"-defining function g such that $\mathcal{H}(g(P,R)) = y$. We note that the attack is trivial for some choices of g (e.g. if it is defined by the identity function on its second argument). Below we give a concrete specification of g.

Definition 1. *Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ be a hash function design employing compression function $f \in$ Func(n+m,n). Let p denote the length of the forced*

midfix, and denote by $L \geq 1$ the maximal length of the forged preimage in blocks. Let $\mathcal{A}$ be a chosen-target-forced-midfix (CTFM) finding adversary for this hash function. The advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{ctfm}}_{\mathcal{H}}(\mathcal{A}) = \mathbf{Pr}\Big(f \xleftarrow{\$} \mathit{Func}(n{+}m{,}n),\ (y, st) \leftarrow \mathcal{A}^f,\ P \xleftarrow{\$} \{0,1\}^p, \\ (g, R) \leftarrow \mathcal{A}^f(P, st) \ \ : \ \ \mathcal{H}^f(g(P, R)) = y \ \wedge \ \big|\mathsf{rng}(g)\big| \leq 2^{Lm}\Big).$$

By $\mathbf{Adv}^{\mathsf{ctfm}}_{\mathcal{H}}(q)$ we denote the maximum advantage, taken over all adversaries making q queries to their oracle.

The function g can technically be any function as long as its range is at most 2^{Lm}, but for some choices of g the definition becomes irrelevant. For instance, if the mutual information between P and $g(P, R)$ is 0, the CTFM attack is trivial. More generally, the attack becomes easier if the function g is allowed to split P into parts. However, this type of functions does not correspond to any practically relevant CTFM attacks. Therefore, in the remainder, we restrict g to satisfy that $g(P, R_1 \| R_2) = R_1 \| P \| R_2$, where R_1, R_2 are of arbitrary length.

The chosen-target-forced-prefix attack of Kelsey and Kohno is covered for g restricted to R_1 being the empty string. The variant of the herding attack by Andreeva et al. [1] on the zipper hash function can be seen as a chosen-target-forced-suffix attack, with R_2 being the empty string.

The value p defines the size of the challenge P, and plays an important role in the security results. A smaller value of p allows for higher success probability in guessing P in the first phase of the attack. A larger value of p limits the number of "free" queries of the adversary, the adversary needs to make at least $\lceil p/m \rceil$ compression function queries for incorporating the challenge P, where m is the message block size.

4 Salted-Chosen-Target-Forced-Midfix Preimage Resistance

In this section we discuss the salted variant of CTFM, which we denote by SCTFM. Depending on whether the challenger generates the salt value S or the adversary himself, and at which stage of the game the salt is chosen, four salted variants of CTFM emerge. We view the salt as an explicit input parameter of size $s \geq 1$ to the hash function $\mathcal{H}$, which is not integrated at an internal compression function level, but processed at the higher (iteration) hash function $\mathcal{H}$ level. We define $\mathsf{enc}_i : \{0,1\}^s \times \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^{n+m}$ to take inputs S, chaining value h_{i-1} and message block M_i, where i varies between 1 and L (which is the maximum length of the preimage) to provide a possible function diversification. The function enc_i is a bijection on h_{i-1} and M_i and we denote its inverse function by enc_i^{-1}. More concretely,

$$\mathsf{enc}_i(S, h_{i-1}, M_i) = (h'_{i-1}, M'_i) \ \wedge \ \mathsf{enc}_i^{-1}(S, h'_{i-1}, M'_i) = (h_{i-1}, M_i).$$

We can view the functions enc_i as keyed permutations on the chaining value and message block inputs to the compression function, where the key is the salt S.

Our choice of this encoding function is guided by a simple security objective. Let us define f'_i as $f'_i(S, h_{i-1}, M_i) = f(\mathsf{enc}_i(S, h_{i-1}, M_i))$ for $i = 1, \ldots, L$. We choose enc_i to be a bijection on h_{i-1} and M_i to provide the full set of valid input points for the function f. Any deviation from this would weaken the cryptographic strength of f, i.e. by allowing an adversary to easily launch collision attacks on the encoding function as a way to circumvent collision computations on f. If enc_i is not a bijection on its variable inputs (notice that once chosen the salt is fixed), then the function f would be working only with a restricted set of its domain points.

We provide the definitions of SCTFM security for the four variants indexed by $j = 1, 2, 3, 4$.

Definition 2. *Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ be a salted hash function design and p, L be as in Def. 1. Let s denote the size of the salt and let $\mathsf{enc} = \{\mathsf{enc}_i\}_{i=1}^{L}$ be the family of encoding functions as described above. Let $\mathcal{B}$ be a salted-chosen-target-forced-midfix (SCTFM) finding adversary for this hash function. For $j \in \{1, 2, 3, 4\}$ the advantage of an adversary $\mathcal{B}$ is defined as*

$$\mathbf{Adv}^{\mathsf{sctfm}}_{\mathcal{H}}(\mathcal{B}) = \mathbf{Pr}\Big(\mathsf{E}_j \ : \ \mathcal{H}^{f,\mathsf{enc}}(g(P,R),S) = y \ \wedge \ \big|\mathsf{rng}(g)\big| \leq 2^{Lm} \ \wedge \ |S| = s\Big).$$

By $\mathbf{Adv}^{\mathsf{sctfm}}_{\mathcal{H}}(q)$ we denote the maximum advantage, taken over all adversaries making q queries to their oracle. The events E_j ($j = 1, 2, 3, 4$) are illustrated by the following four game experiments:

j	E_j
1	$f \xleftarrow{\$} \mathit{Func}(n{+}m,n),\ (y, S, st) \leftarrow \mathcal{B}^f,\ P \xleftarrow{\$} \{0,1\}^p,\ (g, R) \leftarrow \mathcal{B}^f(P, st)$
2	$f \xleftarrow{\$} \mathit{Func}(n{+}m,n),\ S \xleftarrow{\$} \{0,1\}^s,\ (y, st) \leftarrow \mathcal{B}^f(S),\ P \xleftarrow{\$} \{0,1\}^p,\ (g, R) \leftarrow \mathcal{B}^f(P, st)$
3	$f \xleftarrow{\$} \mathit{Func}(n{+}m,n),\ (y, st) \leftarrow \mathcal{B}^f,\ P \xleftarrow{\$} \{0,1\}^p,\ S \xleftarrow{\$} \{0,1\}^s,\ (g, R) \leftarrow \mathcal{B}^f(P, S, st)$
4	$f \xleftarrow{\$} \mathit{Func}(n{+}m,n),\ (y, st) \leftarrow \mathcal{B}^f,\ P \xleftarrow{\$} \{0,1\}^p,\ (g, R, S) \leftarrow \mathcal{B}^f(P, st)$

We provide a discussion on the adversarial abilities for the four SCTFM security notions in comparison with the standard CTFM definition and also to the relevance of salted definitions in practice.

Variants 1, 2 and 4. We will compare the strength of a SCTFM adversary $\mathcal{B}$ in variant 1, 2 or 4 with a CTFM adversary $\mathcal{A}$. Notice first that in a (non-salted) CTFM security game, $\mathcal{A}$ can gain advantage in the precomputation phase (the phase before the challenge midfix is known) only by skillfully preparing some computations for f, such that they are consistent with the evaluation of the function $\mathcal{H}$ in order to compute y. Overall, $\mathcal{A}$ can only exploit f. On the other hand, a SCTFM adversary $\mathcal{B}$ (for either of the variants) encounters the additional

problem that he has to handle the encoding functions enc_i which may differ each message block. With respect to the advantage of $\mathcal{A}$, $\mathcal{B}$'s advantage would differ only in the case $\mathcal{B}$ loses control over the outputs of the enc_i function (which are inputs to f), i.e. in the case when he does not know the salt value.

But in each of these three variants the SCTFM adversary $\mathcal{B}$ knows the salt and has control over the inputs to f (as is the case with $\mathcal{A}$) before his commitment to y, and thus his advantage will be the same as the advantage of $\mathcal{A}$. In variant 1, the SCTFM adversary is in full control of the salt value and in variant 2 he knows the salt before committing to y, therefore he can prepare the respective computations for f. Notice that, although in variant 4 the salt value is revealed by the $\mathcal{B}$ in the second stage of the game, he is still in full control of the salt value and his advantage is optimal when he chooses the salt S in the first phase of the attack, does the respective computations for f, and then reveals S only in the second phase.

This analysis comes to show that the SCTFM adversary has the same computational advantage as a CTFM adversary in variants 1, 2 and 4. The conclusion is that salting in these variants does not help build more secure CTFM hash functions $\mathcal{H}$ and one can do as good without additionally increasing the efficiency and complexity of $\mathcal{H}$.

Variant 3. As opposed to the previous variants, here the SCTFM adversary $\mathcal{B}$ does not have control over the salt value before his commitment to y. In this scenario, $\mathcal{B}$ may lose control over the precomputation, because he is forced to use an unknown salt. This is the case for example when a Merkle-Damgård scheme is used where the salt is XORed with each chaining value. The Kelsey and Kohno attack would fail for such a scheme since the precomputed diamond does not contain the correct salt value, unless the adversary guessed it correctly in advance.

Variant 3, however, is not a valid notion because it does not reflect any real world uses of salts with hash functions. More concretely, variant 3 says that the SCTFM adversary first commits to some value y and only then is challenged on the randomness S, which means that he needs to make his precomputations and commitment without knowing or being able to learn the actual input parameter of the hash function. It is clear that such scenario fails not only from the point of view of adversarial abilities, but from a more practical point. The commitment simply does not make sense since one challenges the committer to make his best guess at the salt value. The salt shall be available at the point of hash computation, because it is an explicit input parameter to the hash function. This is the case with examples of salted hashing, such as password salted hashing and randomized hashing.

We exhibited 4 variants of salted hashing, where 3 of them have equivalent complexity as in the non-salted setting and one does not make it for a meaningful salted definition. As opposed to variant 3, the rest of the salted variants attribute to plausible security definitions. However, the clear conclusion we want

to draw here is that salts do not help prevent CTFM attacks and one shall aim at non-salted solutions. We want to elaborate that this is a conclusion drawn for a reasonable encoding function. A different encoding function might lead to weakening the cryptographic strength of the compression function.

5 CTFM Resistance of Merkle-Damgård Design

We consider the resistance of the traditional Merkle-Damgård design against the chosen-target-forced midfix attack. This Merkle-Damgård hash function is an iterated hash function as in (1) with the following padding function:

$$\mathsf{pad}(M) = M \| 10^{-|M|-1 \bmod m}, \tag{2}$$

splitted into blocks of length m. As demonstrated by Kelsey and Kohno [13] and Blackburn et al. [6], one can obtain a CTFM preimage of length $O(n)$ in about $\sqrt{n}2^{2n/3}$ compression function executions. When larger preimages are allowed, the elongated herding attack of [13] results in faster attacks: for $0 \leq r \leq n/2$, one can find a CTFP preimage of length $L = O(n+2^r)$ in approximately $\sqrt{n}2^{(2n-r)/3}$ queries. As we will demonstrate, this is (asymptotically) the best possible result. In Thm. 1 we derive an upper bound on $\mathbf{Adv}^{\mathsf{ctfm}}_{\mathrm{MD}}(q)$ that holds for any q, and we consider the limiting behavior in Cor. 1, in which we show that at least $2^{2n/3}/L^{1/3}$ queries are needed for an attack to succeed. After Cor. 1 we explain why the same or similar analysis applies to a wide class of MD based functions.

Theorem 1. *For any integral threshold $t > 0$, we have*

$$\mathbf{Adv}^{\mathsf{ctfm}}_{MD}(q) \leq \frac{(L-1)tq}{2^n} + \frac{m2^{\lceil p/m \rceil}q}{2^p} + \left(\frac{q^2 e}{t2^n}\right)^t + \frac{q^3}{2^{2n}}.$$

Proof. See Sect. 5.1. □

The bound of Thm. 1 includes a parameter t used to bound multiple events in the security analysis, and the bound holds for any integral t. Notice that for larger value of t, the first factor of the bound becomes large, while for small values the third term becomes large. Therefore, it is of importance to find a balance for this value t. Recall that, as explained in the beginning of this section, an adversary has a higher success probability if larger preimages are allowed. Consequently, the optimum for t partially depends on the allowed length L. In Cor. 1 we analyze the limiting behavior of the bound of Thm. 1.

Notice that the bound of Thm. 1 contains a term that not directly depends on n, the second term. This term essentially represents the "guessing probability" of the attacker: $\mathcal{A}$ may succeed guessing P in advance. If $p = |P|$ is very small, this factor dominates the bound. Therefore, it only makes sense to evaluate this bound for p being "large enough", and we have to put a requirement on p. Although the requirement looks complicated at first sight, it is satisfied by any relevant value of p. In particular, it is satisfied for $p \geq 2n/3$ for $L = O(n)$ preimages and even for lower values of p when L becomes larger.

Corollary 1. *Let* $L = O(2^{n/2})$ *and let* p *be such that* $\frac{2^{\lceil p/m \rceil} 2^{2n/3}}{L^{1/3} 2^p} = O(1)$ *for* $n \to \infty$. *For any* $\varepsilon > 0$, *we obtain* $\lim_{n\to\infty} \mathbf{Adv}^{\mathsf{ctfm}}_{MD}\left(2^{n(2/3-\varepsilon)}/L^{1/3}\right) = 0$.

Proof. The bound of Thm. 1 holds for any $t \geq 1$. As $L = O(2^{n/2})$, there exists a constant c such that $L \leq c2^{n/2}$. We put $t = \frac{2^{n/3}}{(L/c)^{2/3}} \geq 1$. Without loss of generality, t is integral (one can tweak c a little bit to get integral t). From Thm. 1:

$$\mathbf{Adv}^{\mathsf{ctfm}}_{\mathrm{MD}}(q) \leq \frac{L^{1/3}c^{2/3}q}{2^{2n/3}} + \frac{m2^{\lceil p/m \rceil}q}{2^p} + \left(\frac{(L/c)^{2/3}q^2e}{2^{4n/3}}\right)^{\frac{2^{n/3}}{(L/c)^{2/3}}} + \frac{q^3}{2^{2n}}.$$

For any $\varepsilon > 0$, we obtain:

$$\mathbf{Adv}^{\mathsf{ctfm}}_{\mathrm{MD}}\left(\frac{2^{n(2/3-\varepsilon)}}{L^{1/3}}\right) \leq \frac{c^{2/3}}{2^{n\varepsilon}} + \frac{m}{2^{n\varepsilon}}\frac{2^{\lceil p/m \rceil}2^{2n/3}}{L^{1/3}2^p} + \left(\frac{e}{c^{2/3}2^{2n\varepsilon}}\right)^{\frac{2^{n/3}}{(L/c)^{2/3}}} + \frac{1}{L2^{3n\varepsilon}}.$$

All terms approach 0 for $n \to \infty$ (notice that for the second term we have $m = \Theta(n)$, and for the third term its exponent is ≥ 1). □

Although the security analysis of Thm. 1 and Cor. 1 focuses on the original Merkle-Damgård (MD) hash function, a very similar analysis can be directly derived for a broad class of MD based iterative hash functions, including MD with length strengthening [14], enveloped MD [4], MD with permutation [11] and HAIFA [5]. Indeed, a CTFP attack against strengthened MD is provably harder than an attack against plain MD due to the presence of the length encoding at the end, and a similar remark applies to HAIFA. For enveloped MD and MD with permutation, and in general for any MD based function with final transformation, one can use security properties of the final transformation to show the adversary knows only a limited amount of state values y' which propagate to the commitment y through the final transformation, and we can analyze the success probability with respect to each of these possible commitments y'.[1] The security analysis furthermore applies to the hash twice hash function (where the padded message is simply hashed twice) and the zipper hash function (where the padded message is hashed once forward, and once in the opposite direction) [15], therewith demonstrating the (asymptotic) optimality of the attacks deployed by Andreeva et al. [1]. Indeed, a CTFM attack for zipper or hash twice is provably harder than an attack for MD, but the attacks of Andreeva et al. are of similar complexity as the attack of Kelsey and Kohno on MD.

5.1 Proof of Thm. 1

We introduce some definitions for the purpose of security analysis of MD against the CTFM attack. We consider adversaries making $q = q_1 + q_2$ queries to their

[1] In detail for enveloped MD and MD with permutation, the condition on event $\neg\mathsf{E}_2$ in the proof of Thm. 1 guarantees the adversary to know at most 2 such values y'. The final success probability is then at most 2 times as large.

random oracle f. Associated to the queries made in the attack we consider a directed graph (V, A). A compression function execution $f(h_{i-1}, M_i) = h_i$ corresponds to an arc $h_{i-1} \xrightarrow{M_i} h_i$ in the graph. The graph is initialized as $(\{\mathsf{iv}\}, \emptyset)$. We denote by $(V(j), A(j))$ (for $j = -q_1, \ldots, q_2$) the subgraph of (V, A) after the j-th query. Hence, after the first phase of the attack we are left with graph $(V(0), A(0))$.

A path $h_i \xrightarrow{M_{i+1}} h_{i+1} \cdots \xrightarrow{M_l} h_l$ in a graph is called "$\mathsf{v\text{-}tail}$" (for "valid tail") if $M_{i+1} \| \cdots \| M_l$ forms the suffix of a correctly padded message. Formally, it is $\mathsf{v\text{-}tail}$ if there exists $M_i \in (\{0,1\}^m)^*$ such that

$$M_i \| M_{i+1} \| \cdots \| M_l \| M_{l+1} \in \mathsf{rng}(\mathsf{pad}).$$

Intuitively, it means that $h_i \to h_l$ is formed by a valid sequence of message blocks and can possibly occur at the end of a hash function execution. Notice that the $\mathsf{v\text{-}tail}$ property of a path carries over to its sub-suffixes. For two state values $h_i, h_l \in \{0,1\}^n$, we define

$$\mathsf{dist}_{h_i \to h_l}(j) = \left\{ 0 \leq k < \infty \;\middle|\; (V(j), A(j)) \text{ contains } \mathsf{v\text{-}tail}\ h_i \longrightarrow h_l \text{ of length } k \right\}.$$

For a $P \in \{0,1\}^p$, a path $h_i \xrightarrow{M_{i+1}} h_{i+1} \cdots \xrightarrow{M_k} h_k$ is called "P-comprising" if $M_{i+1} \| \cdots \| M_k$ contains P as a substring.

Proof of Thm. 1. Let $\mathcal{A}$ be any CTFM adversary making $q_1 + q_2$ queries to its random function f. In other words, at a high level the attack works as follows: (1) the adversary makes q_1 queries to f, (2) he commits to digest y, (3) he receives a random challenge $P \xleftarrow{\$} \{0,1\}^p$, (4) he makes q_2 queries to f, and (5) he responds with R_1, R_2 such that $\mathcal{H}(R_1 \| P \| R_2) = y$ and $\left| R_1 \| P \| R_2 \right| \leq Lm$. Denote by $\mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(j)\right)$ for $j = -q_1, \ldots, q_2$ the success probability of $\mathcal{A}$ after the j-th query. Obviously, $\mathbf{Adv}^{\mathsf{ctfm}}_{\mathrm{MD}}(\mathcal{A}) = \mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2)\right)$. In the first phase of the attack, the adversary arbitrarily makes q_1 queries to f, and then outputs a commitment y. Consequently, he receives challenge $P \xleftarrow{\$} \{0,1\}^p$. Consider the following event E_0:

$$\mathsf{E}_0 : \ (V(0), A(0)) \text{ contains a } P\text{-comprising path.}$$

E_0 captures the event of $\mathcal{A}$ "guessing" P in the first phase of the attack. We will split the success probability of the attacker, using the fact that he either did or did not guess P in the first phase. In other words, we can write $\mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2)\right) \leq \mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg \mathsf{E}_0\right) + \mathbf{Pr}\left(\mathsf{E}_0\right)$. However, for the purpose of the analysis we introduce two more events $\mathsf{E}_1, \mathsf{E}_2$. Let $t > 0$ be any integral threshold.

$$\mathsf{E}_1 : \ \alpha_1 > t, \text{ where } \alpha_1 = \max_{0 \leq k \leq L} |\{h \in V(q_2) \mid k = \min \mathsf{dist}_{h \to y}(q_2)\}|,$$

$$\mathsf{E}_2 : \ \alpha_2 > 2, \text{ where } \alpha_2 = \max_{h \in V(q_2)} |\{h' \in V(q_2) \mid (h', h) \in A(q_2)\}|.$$

E_1 sorts all nodes in $V(q_2)$ in classes with elements at (minimal) distance $0, 1, 2, \ldots, L$ from y, and considers the class with the maximal number of nodes.

E_2 considers the event that $\mathcal{Q}$ contains a multi-collision of more than two compression function executions. By basic probability theory, we have

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2)\right) &\leq \mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1\right) + \mathbf{Pr}\left(\mathsf{E}_0 \vee \mathsf{E}_1\right), \\ &\leq \mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1\right) + \mathbf{Pr}\left(\mathsf{E}_0 \vee \mathsf{E}_1 \mid \neg\mathsf{E}_2\right) + \mathbf{Pr}\left(\mathsf{E}_2\right), \\ &\leq \mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1\right) + \mathbf{Pr}\left(\mathsf{E}_0 \mid \neg\mathsf{E}_2\right) \\ &\qquad + \mathbf{Pr}\left(\mathsf{E}_1 \mid \neg\mathsf{E}_2\right) + \mathbf{Pr}\left(\mathsf{E}_2\right), \qquad (3)\end{aligned}$$

and we consider the probabilities on the right hand side separately.

- $\mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1\right)$**.** By $\neg\mathsf{E}_0$, P is not contained in $(V(0), A(0))$ yet, but it may be contained partially and hence the adversary will at least need to make 1 compression function execution. It may be the case that the adversary makes calls to the compression function for multiple strings P, and it may be the case that after he queried for P, he knows multiple paths of different length including P, but this does not violate the analysis. In general, the suffix R_2 of the attack covers at most $L-1$ message blocks. At any time in the attack, there are at most
$$|\{h \in V(q_2) \mid \mathsf{dist}_{h \to y}(q_2) \cap \{0, \ldots, L-1\}|$$
possible nodes for which a hit results in a collision. By $\neg\mathsf{E}_1$, this set is upper bounded by $(L-1)t$. As the adversary makes at most $q_2 \leq q$ compression function calls that may result in success, the total probability is upper bounded by $\frac{(L-1)tq}{2^n}$;
- $\mathbf{Pr}\left(\mathsf{E}_0 \mid \neg\mathsf{E}_2\right)$**.** Notice that $\neg\mathsf{E}_2$ implies that all nodes in $(V(q_2), A(q_2))$, as well as all nodes in $(V(0), A(0))$, have at most 2 incoming arcs. We consider the probability that there exists a P-comprising path. The existence of such path implies the existence of an arc that supplies the last bit of P. Consider any arc $h_{j-1} \xrightarrow{M_j} h_j$, and let $M_j^{(i)}$ for $i = 1, \ldots, m$ denote the m-th bit. Now, we can analyze the probability that $P \xleftarrow{\$} \{0,1\}^p$ is included as a substring of a path in $(V(0), A(0))$, with $M_j^{(i)}$ corresponding to the last bit of P. Then, $\mathbf{Pr}\left(\mathsf{E}_0 \mid \neg\mathsf{E}_2\right)$ is upper bounded by this probability summed over all i and the number of arcs. We consider the probability for different values of i:
  - $i \geq p$. P is integrally captured in M_j as $M_j^{(i-p+1)} \| \ldots \| M_j^{(i)} = P$. This happens with probability $1/2^p$ for predetermined M_j and random P;
  - $i < p$. The first i bits of M_j correspond to the last i bits of P, and that the first $p-i$ bits of P are a suffix of any path ending in h_{j-1}. Let $\beta = \lceil (p-i)/m \rceil$. As by $\neg\mathsf{E}_2$ there are at most 2^β paths of length β blocks to h_{j-1}, we can upper bound the probability by $\frac{1}{2^i} \cdot \frac{2^\beta}{2^{p-i}} = \frac{2^\beta}{2^p}$.

  Now, we can sum over all possible values of i and the number of queries q_1. We obtain
$$\mathbf{Pr}\left(\mathsf{E}_0 \mid \neg\mathsf{E}_2\right) \leq \begin{cases} \frac{(m+p-1)q_1}{2^p} & \text{if } p \leq m, \\ \frac{m2^{\lceil p/m \rceil} q_1}{2^p} & \text{if } p > m. \end{cases}$$
  In both cases, we derive upper bound $\frac{m2^{\lceil p/m \rceil} q}{2^p}$, given $q_1 \leq q$;

– $\mathbf{Pr}\left(\mathsf{E}_1 \mid \neg\mathsf{E}_2\right)$**.** Let k^* be minimal such that the maximum is achieved, and let $h_1, \ldots, h_{\alpha_1}$ be all nodes with distance k^* from y. Consider the subgraph $(\overline{V}, \overline{A})$ of $(V(q_2), A(q_2))$ consisting of all[2] paths $h_i \rightarrow y$ of length k^* edges (for $i = 1, \ldots, \alpha_1$). By ways of an elaborate case distinction (see App. A), one can show that for each node h in $(\overline{V}, \overline{A})$, all paths to y are of the same length. This in particular implies that the h_i's $(i = 1, \ldots, \alpha_1)$ have no ingoing edge, and that y has no outgoing edge. Therefore, we can classify the nodes in the subgraph into sets: $\alpha_1^{k^*} = \alpha_1$ at distance k^* from y, $\alpha_1^{k^*-1}$ at distance k^*-1, etc., $\alpha_1^0 = 1$ at distance 0. Notice that $\alpha_1^0, \ldots, \alpha_1^{k^*-1} < \alpha_1^{k^*}$ by definition, but it can be the case that $\alpha_1^{i-1} > \alpha_1^i$ (for $1 < i < k^*$) for technical reasons. By $\neg\mathsf{E}_2$, $\mathcal{Q}$ does not contain any 3-way collisions, but only 2-way collisions. The number of 2-way collisions between the nodes at distances i and $i-1$ equals $\max\{\alpha_1^i - \alpha_1^{i-1}, 0\}$. Consequently, the described subgraph, and hence $(V(q_2), A(q_2))$ itself, contains at least

$$\sum_{i=1}^{k^*} \max\{\alpha_1^i - \alpha_1^{i-1}, 0\} \geq \alpha_1^{k^*} - \alpha_1^0 = \alpha_1 - 1 \geq t$$

2-way collisions. Thus, the probability is upper bounded by $\binom{q}{t}\left(\frac{q}{2^n}\right)^t \leq \left(\frac{q^2 e}{t2^n}\right)^t$, where the inequality holds due to Stirling's approximation ($x! \geq (x/e)^x$ for any x);

– $\mathbf{Pr}\left(\mathsf{E}_2\right)$**.** The occurrence of E_2 implies the presence of a 3-way collision in $\mathcal{Q}$, which exists with probability at most $q^3/2^{2n}$ only [18].

From equation (3) and above upper bounds on the three probabilities, we obtain:

$$\mathbf{Adv}^{\mathsf{ctfm}}_{\mathrm{MD}}(\mathcal{A}) = \mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2)\right) \leq \frac{(L-1)tq}{2^n} + \frac{m2^{\lceil p/m \rceil}q}{2^p} + \left(\frac{q^2 e}{t2^n}\right)^t + \frac{q^3}{2^{2n}}.$$

As this holds for any adversary making q queries, this completes the proof.

6 On Optimally CTFM Resistant Iterated Hash Functions

Knowing that irrespectively of adding a salt or not, the original MD design of Sect. 5 does not withstand the CTFM attack, a natural question arises: is it possible to secure the MD design by ways of a simple tweak? Naturally, wide-pipe designs offer optimal CTFM security, but they require a larger state size, which accounts for an efficiency loss. Note that modes of operation that use the chaining values in an advanced way (e.g. by adding a final compression function call with the checksum of the chaining values) implicitly belong to the set of wide-pipe designs. Another direction may be to tweak the way the message is

[2] In case of multiple paths of the same length starting from a node h_i, one arbitrarily chooses a path.

processed by the mode of operation, which is captured by considering the iterated hash function design of (1) with a more sophisticated padding.

In this section, we launch a CTFM attack against a wide class of iterated hash functions that differ from the original MD design only in the way the message is processed. More detailed, we consider the standard iterated hash function design of (1), with the difference that it employs a sophisticated padding function $\mathsf{s\text{-}pad}$ satisfying some criteria. This padding function $\mathsf{s\text{-}pad}$ may be depending on the standard padding function pad, and generally does. The attack covers a wide spectrum of hash functions, and in particular provides an efficient and elegant alternative to the attacks proposed by Gauravaram et al. [10] on several MD designs using checksums.

We describe the attack on the base of one representative example hash function. A generic version of it can be directly extracted from the attack description. Further, we consider three similar hash functions and a comparison with the attack of [10].

Let pad be the padding function of (2). Let $M \in \{0,1\}^*$ and denote $\mathsf{pad}(M) = M_1 \| \cdots \| M_l$. We define a sophisticated padding function $\mathsf{s\text{-}pad}_1 : \{0,1\}^* \to (\{0,1\}^m)^*$ on M as follows.

$$\mathsf{s\text{-}pad}_1(M) = M_1 \Big\| \bigoplus_{i=1}^{1} M_i \Big\| M_2 \Big\| \bigoplus_{i=1}^{2} M_i \Big\| \cdots \Big\| M_l \Big\| \bigoplus_{i=1}^{l} M_i.$$

For simplicity, denote by N_i for $i = 1, \ldots, 2l$ the i-th block of $\mathsf{s\text{-}pad}_1(M)$. Let $\mathcal{IH}_1$ be defined as an iterated hash function of (1) accommodated with the advanced padding function $\mathsf{s\text{-}pad}_1$. We will describe a CTFM attack against $\mathcal{IH}_1$, but before that we briefly recall the attack of Kelsey and Kohno against the MD hash function. Denote by $\kappa \geq 1$ the size of the diamond we will use.

1. The adversary constructs a diamond of κ levels. He randomly generates 2^κ state values $h_0^{(1)}, \ldots, h_0^{(2^\kappa)}$, and dynamically finds $2^{\kappa-1}$ compression function collisions by varying the message values. The same procedure is applied to the resulting $2^{\kappa-1}$ state values $h_1^{(1)}, \ldots, h_1^{(2^{\kappa-1})}$ until one node $h_\kappa^{(1)}$ is left;
2. The adversary commits to this state value $y := h_\kappa^{(1)}$ and receives challenge P. Without loss of generality P is of length a multiple of m, and he computes the path $\mathsf{iv} \xrightarrow{P} h_{p'}$;
3. The adversary finds a message M_{hit} such that $f(h_{p'}, M_{\text{hit}}) = h_0^{(j)}$ for some $j \in \{1, \ldots, 2^\kappa\}$.

The resulting forgery is formed by $P \| M_{\text{hit}} \| M_{\text{diam}}$, where M_{diam} denotes the message string that labels the path $h_0^{(j)} \to h_\kappa^{(1)}$. Phase 1 of the attack requires about $\sqrt{\kappa} 2^{(n+\kappa)/2}$ work [13,6], the work for phase 2 is negligible and phase 3 takes about $2^{n-\kappa}$ amount of work. The message is of length $\lceil p/m \rceil + 1 + \kappa$ blocks.

The construction of the diamond (phase 1) is correct due to the independent character of the message blocks: given a string $M_i \| \cdots \| M_l$ of message blocks,

one can arbitrarily change one block while still having a valid string of message blocks. Thus, when constructing the diamond one can vary the message blocks independently for obtaining collisions. For the sophisticated padding function $\mathsf{s\text{-}pad}_1$ this is not possible. If for a given padded message one changes N_{2i-1} for $i \in \{1, \ldots, l-1\}$, the values taken by the checksum blocks $N_{2i}, \ldots, N_{2i+2}, \ldots, N_{2l}$ should change as well. At first sight, this makes the construction of the diamond impossible, but by additionally changing N_{2i+1}, one can "stabilize" the values $N_{2i+2}, \ldots, N_{2l}$ and only the blocks $N_{2i-1}, N_{2i}, N_{2i+1}$ get affected (in case $i = l$ only N_{2i-1}, N_{2i} get affected). Based on this observation the attack is defined as follows. Notice, the adversary decides on the length of the forgery in advance: $p' + 2\kappa + 2$.

1. The adversary constructs a diamond of κ levels.
   - He fixes constants $C_0, C_1, \ldots, C_\kappa \in \{0,1\}^m$ in advance. These constants represent

$$C_0 = \bigoplus_{i=1}^{p'+2} M_i, \qquad C_i = M_{p'+2i+1} \oplus M_{p'+2i+2} \text{ for } i = 1, \ldots, \kappa. \tag{4}$$

     The adversary does not know the blocks M_i yet, but will choose them so as to comply with (4);
   - He randomly generates 2^κ state values $h_0^{(1)}, \ldots, h_0^{(2^\kappa)}$, and dynamically finds collisions of the following form for $j = 1, \ldots, 2^{\kappa-1}$

$$\begin{array}{l} h_0^{(2j-1)} \xrightarrow{C_0} \xrightarrow{M_{p'+3}^{(2j-1)}} \xrightarrow{C_0 \oplus M_{p'+3}^{(2j-1)}} \xrightarrow{C_1 \oplus M_{p'+3}^{(2j-1)}} \\ h_0^{(2j)} \xrightarrow{C_0} \xrightarrow{M_{p'+3}^{(2j)}} \xrightarrow{C_0 \oplus M_{p'+3}^{(2j)}} \xrightarrow{C_1 \oplus M_{p'+3}^{(2j)}} \end{array} h_1^{(j)}.$$

     These collisions can indeed by dynamically found, simply by varying the $M_{p'+3}$-blocks (recall that C_0, C_1 are fixed constants). The corresponding blocks $M_{p'+4}$ are computed as $M_{p'+4} = C_1 \oplus M_{p'+3}$ by (4);
   - The same procedure is applied to the resulting $2^{\kappa-1}$ state values $h_1^{(1)}$, $\ldots, h_1^{(2^{\kappa-1})}$, where the arcs are labeled by $C_0 \oplus C_1$, $M_{p'+5}$, $C_1 \oplus M_{p'+5}$ and $C_2 \oplus M_{p'+5}$, respectively. Finally, one node $h_\kappa^{(1)}$ is left;
2. The adversary commits to this state value $y := h_\kappa^{(1)}$ and receives challenge P. Without loss of generality P is of length a multiple of m, and he defines $M_1, \ldots, M_{p'}$ to be the first corresponding blocks. Denote $C_{-1} = \bigoplus_{i=1}^{p'} M_i$. In accordance to the padding function $\mathsf{s\text{-}pad}$ he computes the path $\mathsf{iv} \xrightarrow{M_1} \ldots \xrightarrow{M_{p'}} \xrightarrow{C_{-1}} h_{2p'}$;
3. The adversary finds a message $M_{p'+1}$ such that

$$h_{2p'} \xrightarrow{M_{p'+1}} \xrightarrow{C_{-1} \oplus M_{p'+1}} \xrightarrow{C_{-1} \oplus M_{p'+1} \oplus C_0} h_0^{(j)}$$

   for some $j \in \{1, \ldots, 2^\kappa\}$.

The resulting forgery is formed by $P\|M_{p'+1}\|M_{p'+2}\|M_{\mathrm{diam}}$, where $M_{p'+2} = C_{-1} \oplus M_{p'+1} \oplus C_0$ by (4) and M_{diam} denotes the message string of 2κ blocks that labels the path $h_0^{(j)} \to h_\kappa^{(1)}$. By construction, because the values $C_0, \ldots, C_\kappa$ have been fixed in advance, the path $\mathsf{iv} \longrightarrow h_\kappa^{(1)}$ is in accordance with the padding function $\mathsf{s\text{-}pad}$. Phase 1 of the attack requires about $4 \cdot \sqrt{\kappa} 2^{(n+\kappa)/2}$ work, the work for phase 2 is negligible and phase 3 takes about $3 \cdot 2^{n-\kappa}$ amount of work. The message is of length $\lceil p/m \rceil + 2 + 2\kappa$ blocks.

We notice that the same approach can be applied to the following example hash functions. Consider the following advanced padding functions $\mathsf{s\text{-}pad}_k(M) : \{0,1\}^* \to (\{0,1\}^m)^*$ for $k = 2, 3, 4$:

$$\mathsf{s\text{-}pad}_2(M) = M_1 \Big\| M_2 \Big\| M_1 \oplus M_2 \Big\| M_3 \Big\| M_2 \oplus M_3 \Big\| \cdots \Big\| M_l \Big\| M_{l-1} \oplus M_l,$$

$$\mathsf{s\text{-}pad}_3(M) = \mathsf{rotate}_{m/2}(\mathsf{pad}(M)),$$

$$\mathsf{s\text{-}pad}_4(M) = \mathsf{pad}(M) \Big\| \bigoplus_{j=1}^{l} M_j,$$

where the function $\mathsf{rotate}_{m/2}$ rotates the bit string by $m/2$ places (a half message block). We define by $\mathcal{IH}_k$ for $k = 2, 3, 4$ the standard iterated hash function of (1) accommodated with the advanced padding function $\mathsf{s\text{-}pad}_k$. Notice that for $\mathcal{IH}_4$, any change of M_i can be corrected by M_{i+1} to keep the final checksum invariant. Now, the attacks are described in a similar manner. For $\mathcal{IH}_2$, the complexity is the same as for $\mathcal{IH}_1$. The complexities for $\mathcal{IH}_3, \mathcal{IH}_4$ are half as large. For each of the functions $\mathcal{IH}_k$ the optimum is achieved for $\kappa = n/3$. By tweaking the proof of Thm. 1, asymptotic tightness of this bound can be proven. We notice that Gauravaram et al. [10] describe a generalized herding attack against a class of MD based hash functions using checksums at the end (such as $\mathcal{IH}_4$). The attack described in this section carries over to many of these designs, therewith providing an elegant alternative. These attacks are of the same complexity, although our attack renders shorter messages in case $n/3 < m$. The cause of this difference is the fact that the attack of Gauravaram et al. sets the value of the final checksum at the end while in our attack it is essentially fixed by the adversary in advance.

We leave the existence of narrow-pipe hash functions that achieve optimal security against the CTFM attack as an open problem.

7 Conclusions

We introduced and formalized the notion of a chosen-target-forced-midfix (CTFM) attack as a generalization of the classical herding attack of Kelsey and Kohno [13]. The new notion allows the adversary to include the challenge P at any place in the forged preimage. Hence, it enables arguing the security of hash functions which for example process the message in reverse order and which were otherwise trivially secure against the herding attack. Additionally,

we investigated the CTFM security of salted hash functions showing that adding a salt value without weakening the compression function does not improve the CTFM security of the hash function.

As a main technical contribution of the paper we provided a formal security proof of the MD design against the CTFM attack, and showed that the attack of Kelsey and Kohno [13] is (asymptotically) the best possible. This proof directly applies to a wide class of MD based domain extenders, and implies optimality of other herding attacks, such as those of Andreeva et al. [1] and Gauravaram et al. [10].

In the quest for optimally CTFM secure narrow-pipe MD designs, we analyzed the possibility of message modification as a tool to increase CTFM security. Our result shows however, that such techniques applied to a wide class of narrow-pipe iterated hash function designs do not block CTFM attacks. An open research question that emerges from these observations is to construct a narrow-pipe iterated hash functions that achieves optimal security against the CTFM attacks.

Acknowledgments. This work has been funded in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II, and in part by the Research Council K.U.Leuven: GOA TENSE. The first author is supported by a Ph.D. Fellowship from the Flemish Research Foundation (FWO-Vlaanderen). The second author is supported by a Ph.D. Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

1. Andreeva, E., Bouillaguet, C., Dunkelman, O., Kelsey, J.: Herding, Second Preimage and Trojan Message Attacks Beyond Merkle-Damgård. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 393–414. Springer, Heidelberg (2009)
2. Andreeva, E., Bouillaguet, C., Fouque, P.-A., Hoch, J., Kelsey, J., Shamir, A., Zimmer, S.: Second Preimage Attacks on Dithered Hash Functions. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 270–288. Springer, Heidelberg (2008)
3. Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-Property-Preserving Iterated Hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)
4. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)
5. Biham, E., Dunkelman, O.: A framework for iterative hash functions – HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007)
6. Blackburn, S., Stinson, D., Upadhyay, J.: On the complexity of the herding attack and some related attacks on hash functions. Des. Codes Cryptography (to appear, 2011)

7. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
8. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
9. Gauravaram, P., Kelsey, J.: Linear-XOR and Additive Checksums Don't Protect Damgård-Merkle Hashes from Generic Attacks. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 36–51. Springer, Heidelberg (2008)
10. Gauravaram, P., Kelsey, J., Knudsen, L., Thomsen, S.: On hash functions using checksums. International Journal of Information Security 9(2), 137–151 (2010)
11. Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
12. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
13. Kelsey, J., Kohno, T.: Herding Hash Functions and the Nostradamus Attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
14. 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)
15. Liskov, M.: Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
16. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
17. Neven, G., Smart, N., Warinschi, B.: Hash function requirements for Schnorr signatures. Journal of Mathematical Cryptology 3(1), 69–87 (2009)
18. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday Paradox for Multi-Collisions. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 29–40. Springer, Heidelberg (2006)

A Appendum to Proof of Thm. 1

We show that the graph $(\overline{V}, \overline{A})$ defined in Thm. 1 does not contain a node with two paths of different lengths to y. Recall that $(\overline{V}, \overline{A})$ is constructed in the following manner. k^* is the minimal value for which there are the maximum number of nodes, α_1 with distance k^* to y. For each of the α_1 nodes $h_1, \ldots, h_{\alpha_1}$, we take any path of length k^* to y, and add it to $(\overline{V}, \overline{A})$. By definition, for each $i = 1, \ldots, \alpha_1$, there does not exist a path $h_i \to y$ of length shorter than k^* arcs. We show that for any node $h \in \overline{V}$, all paths to y are of the same length. The proof is done by contradiction: we will show that the existence of an h contradicting aforementioned property implies the existence of a path $h_i \to y$ (for some i) of length strictly shorter than k^* arcs. Notice that this result in particular implies that the h_i's ($i = 1, \ldots, \alpha_1$) have no ingoing edge, and that y has no outgoing edge.

Suppose there exists $h \in \overline{V}$ such that for some $M_1, M_2 \in (\{0,1\}^m)^*$ with $|M_1| < |M_2|$ the paths $h \xrightarrow{M_1} y$ and $h \xrightarrow{M_2} y$ are included in $(\overline{V}, \overline{A})$. If the path

$h \xrightarrow{M_2} y$ is a subpath of any $h_i \to y$ for some i, one can replace this subpath by $h \xrightarrow{M_1} y$ to obtain a path $h_i \to y$ of length strictly shorter than k^* arcs, rendering contradiction. Thus, we assume that $h \xrightarrow{M_2} y$ is not integrally included as a subpath of any $h_i \to y$. We split up the path $h \xrightarrow{M_2} y$ intro three parts. Let $i \in \{1, \ldots, \alpha_1\}$ be such that the first edge of $h \xrightarrow{M_2} y$ is included in the path $h_i \to y$. Let $M_2^{(1)}$ be the maximal prefix of M_2 such that $h \xrightarrow{M_2^{(1)}} h^{(1)}$ (for some $h^{(1)}$) is a subpath of $h_i \to y$. Secondly, identify the edge leaving[3] $h^{(1)}$ in the path $h \xrightarrow{M_2} y$, and let i' be such that this edge is included in the path $h_{i'} \to y$. Let $M_2^{(2)}$ be of maximal length such that $M_2^{(1)} \| M_2^{(2)}$ is a prefix of M_2 and $h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)}$ (for some $h^{(2)}$) is a subpath of $h_{i'} \to y$. Thus, we splitted $h \xrightarrow{M_2} y$ into

$$h \xrightarrow{M_2^{(1)}} h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)} \xrightarrow{M_2^{(3)}} y, \tag{5}$$

where $|M_2^{(1)}|, |M_2^{(2)}| > 0$ and $|M_2^{(3)}| \geq 0$ and

$$h_i \xrightarrow{M_3} h \xrightarrow{M_2^{(1)}} h^{(1)} \xrightarrow{M_4} y, \qquad h_{i'} \xrightarrow{M_5} h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)} \xrightarrow{M_6} y, \tag{6}$$

for some $M_3, M_4, M_5, M_6 \in (\{0,1\}^m)^*$. Here, $M_2^{(1)}$ and $M_2^{(2)}$ are of maximal possible length, i.e. the first arcs of $h^{(1)} \xrightarrow{M_4} y$ and $h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)}$ are different and the first arcs of $h^{(2)} \xrightarrow{M_6} y$ and $h^{(2)} \xrightarrow{M_2^{(3)}} y$ are different.

If $h^{(1)} = h$, the path $h_i \xrightarrow{M_3} h \xrightarrow{M_4} y$ is in $(V(q_2), A(q_2))$ and of length shorter than k^* blocks, rendering contradiction. Similarly, if $h^{(2)} = h^{(1)}$, a shorter path $h_{i'} \to y$ can be found. Hence, we consider the case $h \neq h^{(1)} \neq h^{(2)}$, and make the following case distinction:

1. $|M_4| \neq |M_2^{(2)} M_6|$**.** One can combine the two paths described in (6) to obtain either a path $h_i \to y$ or $h_{i'} \to y$ of length strictly shorter than k^* arcs;
2. $|M_4| = |M_2^{(2)} M_6|$**.** We make the following case distinction:
   a. $|M_6| \geq |M_2^{(3)}|$**.** This means that $|M_4| \geq |M_2^{(2)} M_2^{(3)}|$ and hence $|M_2^{(1)} M_4| \geq |M_2| > |M_1|$. The path $h_i \xrightarrow{M_3} h \xrightarrow{M_1} y$ is thus strictly shorter than k^* arcs;
   b. $|M_6| < |M_2^{(3)}|$**.** One can do the same analysis with paths $h^{(2)} \xrightarrow{M_6} y$ and $h^{(2)} \xrightarrow{M_2^{(3)}} y$. But by construction $|M_2^{(3)}| < |M_2| - 2m$ so one will eventually end up with the same problem with $|M_2^{(3)}| = 0$, in which case one will not arrive in case 2b.

Concluding, there does not exist any node in $(\overline{V}, \overline{A})$ which has two paths of different lengths to y.

[3] This edge exists, as $h \xrightarrow{M_2} y$ is not an integral subpath of any path $h_i \to y$.