On Diamond Structures and Trojan Message Attacks

Tuomas Kortelainen[1] and Juha Kortelainen[2]

[1] Mathematics Division, Department of Electrical Engineering
[2] Department of Information Processing Science
University of Oulu

Abstract. The first part of this paper considers the diamond structures which were first introduced and applied in the herding attack by Kelsey and Kohno [7]. We present a new method for the construction of a diamond structure with 2^d chaining values the message complexity of which is $O(2^{\frac{n+d}{2}})$. Here n is the length of the compression function used. The aforementioned complexity was (with intuitive reasoning) suggested to be true in [7] and later disputed by Blackburn et al. in [3].

In the second part of our paper we give new, efficient variants for the two types of Trojan message attacks against Merkle-Damgård hash functions presented by Andreeva et al. [1] The message complexities of the Collision Trojan Attack and the stronger Herding Trojan Attack in [1] are $O(2^{\frac{n}{2}+r})$ and $O(2^{\frac{2n}{3}} + 2^{\frac{n}{2}+r})$, respectively. Our variants of the above two attack types are the Weak Trojan Attack and the Strong Trojan Attack having the complexities $O(2^{\frac{n+r}{2}})$ and $O(2^{\frac{2n-s}{3}} + 2^{\frac{n+r}{2}})$, respectively. Here 2^r is the cardinality of the prefix set and 2^s is the length of the Trojan message in the Strong Trojan Attack.

1 Introduction

Hash functions are mappings which take as input arbitrary strings over a fixed alphabet (usually assumed to be the binary alphabet $\{0,1\}$) and return a (binary) string of a fixed length as their output. These functions are used in various cryptographic protocols such as message authentication, digital signatures and electronic voting. In order to be useful in cryptographic context, hash functions need to have three traditional properties, *preimage resistance, second preimage resistance* and *collision resistance*.

An ideal hash function $\mathtt{H}$ from the set $\{0,1\}^*$ of all binary strings into the set $\{0,1\}^n$ of all binary strings of length n is a *random oracle*: for each $x \in \{0,1\}^*$, the value $\mathtt{H}(x) \in \{0,1\}^n$ is chosen uniformly at random.

Merkle and Damgård [4,13] devised a method for constructing hash functions from a family of *fixed size collision–free compression functions*. In this method, the message to be hashed is divided into blocks and padded; the hash value is computed by the repeated (iterative) use of the compression function to the message blocks and to the previous value of the computation. The result of the final computation is then defined to be the hash value of the message. Both

K. Sako and P. Sarkar (Eds.) ASIACRYPT 2013, Part II, LNCS 8270, pp. 524–539, 2013.

Merkle and Damgård were able to prove that if the length (in blocks) of the message is appended to the original message and the result is padded and hashed in the previous iterative fashion using a collision resistant compression function, then the resulting hash function is also collision resistant.

The iterative method for constructing hash functions, from a single compression function, has been found quite susceptible to several different types of attacks. Joux [6] demonstrated that, for iterated hash functions (of length n), 2^k–collisions can be found with $O(k \cdot 2^{n/2})$ compression function queries; the respective number of queries for a random oracle hash function is much higher [15]. Multicollision attacks against more generalized hash function structures have been studied in [14,5,9,10], and [11]. A second preimage attack against long messages was first constructed by Kelsey and Schneier [8] while in [7] Kelsey and Kohno presented a new form of attack named the *herding attack*.

The herding attack relies on *diamond structures*, a tree construction where several hash values (leaves of the tree) are herded towards one (fixed) hash value (the root). Diamond structures proved to be very useful in attack construction. They were employed in [1] and [2] to create herding and second preimage attacks against several iterated hash function variants also beyond Merkle-Damgåd. Our special interest, Trojan message attacks [1], can also be based on diamond structures.

Now, in the paper [7], a method to construct diamond structures was also introduced. With intuitive reasoning the authors deduced that to build a diamond structure with 2^d chaining values takes approximately $2^{\frac{n+d}{2}+2}$ compression function queries. Later a more comprehensive study of diamond structures [3] pointed out that the complexity estimation was too optimistic, the true complexity of the method presented being $O(\sqrt{d}\ 2^{\frac{n+d}{2}})$. We shall demonstrate a new (and, unfortunately, also more intricate) construction algorithm with message complexity $O(2^{\frac{n+d}{2}})$ for a diamond structure of 2^d chaining values. Our algorithm is based on recycling previously created hash values and message blocks.

The second goal of our paper is to fortify the two Trojan message attacks developed in [1]. Our variant for the weaker Collision Trojan Attack (possessing the complexity $O(2^{\frac{n+r}{2}})$) is more efficient than the original one (with the complexity $O(2^{\frac{n}{2}+r})$), and moreover, offers the attacker a greater freedom to choose the content of the second preimage message. The attack algorithm makes use of diamond structures. Finally, we are able to significantly reduce the complexity of the Herding Trojan Attack in our version of strong Trojan message attack. Both expandable messages [8] elongated diamond structures [7] are exploited in our construction.

This paper is organized in the following way. In the next section, our new method to generate a diamond structure is presented. Section 3 contains an introduction to Trojan message attacks. We formulate a new security property and study the complexity of creating a Trojan message attack against a random oracle hash function. In the fourth section two new and efficient variants of Trojan message attacks are developed. The final section contains some conclusive remarks.

2 Diamond Structures

From now on, assume that our compression function f is a mapping: $\{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ such that $m > n$. The message hashing is carried out with the *iterative closure* $f^* : \{0,1\}^n \times (\{0,1\}^m)^* \to \{0,1\}^n$ of f which is defined inductively as follows. For the empty word ϵ, let $f^*(h,\epsilon) := h$ for all $h \in \{0,1\}^n$. For each $k \in \mathbb{N}$, words $x_1, x_2, \ldots, x_{k+1} \in \{0,1\}^m$, and $h \in \{0,1\}^n$, let $f^*(h, x_1x_2 \cdots x_{k+1}) := f(f^*(h, x_1x_2 \cdots x_k), x_{k+1})$. Note that for $k = 0$ above, $x_1x_2 \cdots x_k$ is the empty word ϵ and the definition allows us to deduce that $f^*(h, x_1) = f(h, x_1)$. All message lengths are expressed in number of blocks.

2.1 Concepts and Tools

Let $H \subseteq \{0,1\}^n$ be a finite nonempty set of hash values. A *pairing set* of H is any set $B \subseteq H \times \{0,1\}^m$ such that

(i) for each $h \in H$ there exists exactly one $x \in \{0,1\}^m$ such that $(h,x) \in B$; and

(ii) for each $(h_1, x_1) \in B$ there exist $(h_2, x_2) \in B$ such that $h_1 \neq h_2$ and $f(h_1, x_1) = f(h_2, x_2)$.

The following technical result is eventually applied in evaluating the cardinalities of message block sets when building the diamond structure.

Lemma 1. *Let $r \geq 2$ and n be positive integers. Define the integers $s_{r,0}, s_{r,1}, s_{r,2}, \ldots, s_{r,2^{r-2}}$ as follows.*

$$s_{r,0} = \lceil 2^{\frac{n-r}{2}-1} \rceil \qquad s_{r,k+1} = s_{r,k} + \left\lceil \frac{2^{\frac{n-r}{2}+1}}{2^r - 2k} \right\rceil \quad \text{for } k = 0, 1, \ldots, 2^{r-2} - 1$$

Then $s_{r,j} \geq \frac{2^{\frac{n+r}{2}-1}}{2^r - 2j}$ for each $j \in \{0, 1, \ldots, 2^{r-2}\}$.

Proof. Proceed by induction on j. The case $j = 0$ is clear. Suppose that $s_{r,k} \geq \frac{2^{\frac{n+r}{2}-1}}{2^r - 2k}$ where $k \in \{0, 1, \ldots, 2^{r-2} - 1\}$. Then, by definition, the inequality

$$s_{r,k+1} \geq \frac{2^{\frac{n+r}{2}-1} + 2^{\frac{n-r}{2}+1}}{2^r - 2k}$$

holds. It suffices to show that

$$\frac{2^{\frac{n+r}{2}-1} + 2^{\frac{n-r}{2}+1}}{2^r - 2k} \geq \frac{2^{\frac{n+r}{2}-1}}{2^r - 2(k+1)} \quad .$$

But this is obvious since the inequality

$$(2^{\frac{n+r}{2}-1} + 2^{\frac{n-r}{2}+1})[2^r - 2(k+1)] \geq 2^{\frac{n+r}{2}-1}(2^r - 2k)$$

is equivalent with $k \leq 2^{r-2} - 1$. □

A *diamond structure* (with 2^d chaining values, or of breadth 2^d), where $d \in \mathbb{N}_+$, is a both vertex labeled and edge labeled complete binary tree D satisfying the following conditions.

1. The tree D has 2^d leaves, i.e., the height of the tree is d.
2. The vertices of the tree D are labeled by hash values (strings in the set $\{0,1\}^n$) so that the labels of vertices that are on the same distance from the root of D are pairwise disjoint.
3. The edges of the tree D are labeled by message blocks (strings in the set $\{0,1\}^m$).
4. Let v_1, v_2, and v with (hash value) labels h_1, h_2, and h, respectively, be any vertices of the tree D such that v_1 and v_2 are children of v. Suppose furthermore that x_1 and x_2 are (message) labels of the edges connecting v_1 to v and v_2 to v, respectively. Then $f(h_1, x_1) = f(h_2, x_2) = h$.

2.2 Intuitive Description of the Diamond Structure Construction Method

Our method advances in *jumps*, *phases*, and *steps*. In each jump several phases are carried out, every phase consists of numerous steps, and in each step we search two distinct hash value and message block pairs (h_1, x_1), (h_2, x_2) such that $f(h_1, x_1) = f(h_2, x_2)$. By dividing the process in aforementioned manner and recycling hash value and message block sets, we are able to decrease the number of compression function queries. It is quite easy to see that our method is not optimal, but we have to make a compromise between completeness and the simplicity of computations.

Jumps. The construction of a diamond structure D with 2^d chaining values $d \geq 2$ is carried out in d jumps $\mathtt{J}_d$, $\mathtt{J}_{d-1}$, ..., $\mathtt{J}_1$. We proceed from the leaves towards the root of the structure. Let H_d be the set of the 2^d chaining values. In jump $\mathtt{J}_d$, a pairing set B_d of H_d is created. The set B_d is constructed so that the cardinality of the set $H_{d-1} := \{f(h,x) \,|\, (h,x) \in B_d\}$ is 2^{d-1}. In jump $\mathtt{J}_{d-1}$ a pairing set B_{d-1} of H_{d-1} is created so that the cardinality of the set $H_{d-2} := \{f(h,x) \,|\, (h,x) \in B_{d-1}\}$ is 2^{d-2}. We continue like this until in the last jump $\mathtt{J}_1$ a pairing set B_1 of H_1 containing only two hash values is generated. The set $H_0 := \{f(h,x) \,|\, (h,x) \in B_1\}$ contains only one element which is the root of the diamond structure. By each jump the distance to the root of the diamond structure is decreased by one. Obviously we are herding the chaining values towards the final hash value which labels the root of our structure.

Now each jump consists of several *phases*; since the structures of jumps are mutually identical, we give below an intuitive description of the phases (and steps) of the jump $\mathtt{J}_d$ only.

Phases. The jump $\mathtt{J}_d$ consists of d phases $\mathtt{P}_d$, $\mathtt{P}_{d-1}$, ..., $\mathtt{P}_2$, $\mathtt{P}_1$. In the phase $\mathtt{P}_d$ of the jump $\mathtt{J}_d$ we create a pairing set T_{d-1} of a subset $K_{d-1} \subseteq H_d$ of cardinality

2^{d-1}, in the phase $\mathtt{P}_{d-1}$ a pairing set T_{d-2} of a subset $K_{d-2} \subseteq H_d \setminus K_{d-1}$ of cardinality 2^{d-2}, and so on, ..., in the phase $\mathtt{P}_2$ a pairing set T_1 of a subset K_1 of $H_d \setminus (K_{d-1} \cup K_{d-2} \cup \cdots \cup K_2)$ of cardinality 2. There are two hash values (forming the set K_0) still without pairing left in H_d, so in the phase $\mathtt{P}_1$ we search a pairing T_0 of K_0. Then we set $B_d := T_{d-1} \cup T_{d-2} \cup \cdots \cup T_0$. Thus the jump $\mathtt{J}_d$ consists of d phases after which we have created a pairing set B_d of H_d; moreover, it proves to be constructed so that the input set $H_{d-1} := \{f(h,x) \,\big|\, (h,x) \in B_d\}$ of jump $\mathtt{J}_{d-1}$ is of cardinality 2^{d-1}.

Steps. Each phase is made up of several *steps* in the following way. Consider the phase $\mathtt{P}_j$ of jump $\mathtt{J}_d$, where $j \in \{2,3,\ldots,d\}$. As told above, in this phase we create a pairing set for a subset K_{j-1} of $H_d \setminus (K_{d-1} \cup K_{d-2} \cup \cdots \cup K_j)$ of cardinality 2^{j-1}. The phase is divided into 2^{j-2} steps

$$\mathtt{S}(d,j,0),\ \mathtt{S}(d,j,1),\ \ldots\ ,\ \mathtt{S}(d,j,2^{j-2}-1)\quad .$$

In each step we create a pairing for two hash values in $H_d \setminus (K_{d-1} \cup K_{d-2} \cup \cdots \cup K_j)$ so that together the hash values in the pairs form a set K_{j-1} of cardinality 2^{j-1}. A more rigorous description of each step with appropriate input and output follows.

Initialization $\mathtt{I}(d)$
As an input we have a set $A_{d,0} := H_d$ of 2^d hash values. We first create a message block set $M_{d,0} \subseteq \{0,1\}^m$ such that

1. the cardinality of $M_{d,0}$ is $2^{\frac{n-d}{2}-1}$; and
2. the cardinality of the set $f(A_{d,0}, M_{d,0}) = \{f(h,x) \big| h \in A_{s,0}, x \in M_{d,0}\}$ is $2^{\frac{n+d}{2}-1}$.

Let $H_{d,0} = f(A_{d,0}, M_{d,0})$. The complexity to construct such an $H_{d,0}$ is approximately $2^{\frac{n+d}{2}-1}$. Note that our assumption on the cardinality of the set $H_{d,0}$ has an insignificant impact to the complexity; we can easily replace the appropriate message blocks one by one with new ones. The output of the initialization step is: $A_{d,0}$; $M_{d,0}$; $H_{d,0}$.

Let now $j \in \{2,3,\ldots,d\}$ and $k \in \{0,1,2,\ldots,2^{j-2}-1\}$.

Step $\mathtt{S}(d,j,k)$
The step takes as an input $A_{j,k}$, $M_{j,k}$, $H_{j,k}$. Here $A_{j,k}$ is a set of $2^j - 2k$ hash values, $M_{j,k}$ is a set of $s_{j,k}$ message blocks, where $s_{j,k} \geq \frac{2^{\frac{n+j}{2}-1}}{2^j-2k}$, and

$$H_{j,k} \;=\; \{f(x,h) \,\big|\, x \in A_{j,k}\,,\ x \in M_{j,k}\}$$

is a set of hash values such that $|H_{j,k}| = |A_{j,k}| \cdot |M_{j,k}|$. Note that $|H_{j,k}| \geq (2^j - 2k)s_{j,k} \geq 2^{\frac{n+j}{2}-1}$.

A set $M'_{j,k}$ of $\lceil s_{j,k+1} - s_{j,k} \rceil$ new messages is generated so that the cardinality of the set

$$f(A_{j,k}, M'_{j,k}) \;=\; \{f(h,x) \,\big|\, h \in A_{j,k}\,,\ x \in M'_{j,k}\}$$

is at least $2^{\frac{n-j}{2}+1}$. We search for hash values $h_{j_k}, h'_{j_k} \in A_{j,k}$ and message blocks $x_{j_k} \in M_{j,k}$, $x'_{j_k} \in M'_{j,k}$ such that $f(h_{j_k}, x_{j_k}) = f(h'_{j_k}, x'_{j_k})$. Note that since $|H_{i,k} \times f(A_{j,k}, M'_{j,k})| \geq 2^n$, the expected number of hash values h such that $h \in H_{i,k} \cap f(A_{j,k}, M'_{j,k})$ is at least one. Furthermore, for the sake of simplicity of computations, we assume that (h_{j_k}, x_{j_k}) and (h'_{j_k}, x'_{j_k}) are the only colliding pairs in $A_{j,k} \times [M_{j,k} \cup M'_{j,k}]$. Now, what is the (message) complexity of the actions and assumptions above? We may create the message set $M'_{j,k}$ as a statistical experiment and then compute the hash values in the set $f(A_{j,k}, M'_{j,k})$. Since $|H_{i,k} \times f(A_{j,k}, M'_{j,k})| \geq 2^n$, a routine reasoning shows that the probability of finding a colliding pair is greater than 0.5. This means that the expected number of times we have to repeat the experiment is less than two. Thus the message complexity to create the set $M'_{j,k}$, compute the values in $f(A_{j,k}, M'_{j,k})$, and to find the colliding pair is at most $2 \cdot 2^{\frac{n-j}{2}+1}$. Our assumptions on the cardinality of $f(A_{j,k}, M'_{j,k})$ and of the number of colliding pairs do not increase the complexity significantly. This is ensured by either repeating the experiment sufficiently many times or replacing messages in the set $M'_{j,k}$ one by one with new ones.

Let $A_{j,k+1} := A_{j,k} \setminus \{h_{j_k}, h'_{j_k}\}$, $M_{j,k+1} := M_{j,k} \cup M'_{j,k}$, and $H_{j,k+1} := f(A_{j,k+1}, M_{j,k+1})$. Furthermore we set $B_d := B_d \cup \{(h_{j_k}, x_{j_k}), (h'_{j_k}, x'_{j_k})\}$.

As an output of this step, we get $A_{j,k+1}$, $M_{j,k+1}$, $H_{j,k+1}$, and B_d.

The output of a the step $\mathtt{S}(d, j, 2^{j-2} - 1)$ (the last step of the phase $\mathtt{P}_j$) serves as the input to the $\mathtt{S}(d, j-1, 0)$ (the first step of the phase $\mathtt{P}_{j-1}$) for each $j \in \{3, 4, \ldots, d\}$. We thus define $A_{j-1,0} := A_{j,2^{j-2}-1}$, $M_{j-1,0} := M_{j,2^{j-2}-1}$, and $H_{j-1,0} := H_{j,2^{j-2}-1}$.

We carry out our diamond structure construction by running the jumps $\mathtt{J}_d$, $\mathtt{J}_{d-1}$, ..., $\mathtt{J}_2$, $\mathtt{J}_1$ one after another in this order. We describe the inner realization of the jump $\mathtt{J}_d$ more accurately; all the other jumps are carried out completely analogously. The jump $\mathtt{J}_d$ is implemented by running all its phases $\mathtt{I}(d)$, $\mathtt{P}_d$, $\mathtt{P}_{d-1}$, ..., $\mathtt{P}_2$, $\mathtt{P}_1$. The last phase $\mathtt{P}_1$ takes as its input only the set $A_{1,0} := A_{2,1}$ of two (remaining) hash values and the pairing set B_d. It searches a pairing set for $A_{1,0}$ on its own. Each phase $\mathtt{P}_j$, $j \in \{2, 3, \ldots, d\}$, is realized by running all its steps $\mathtt{S}(d, j, 0)$, $\mathtt{S}(d, j, 1)$, ..., $\mathtt{S}(d, j, 2^{j-2} - 1)$ subsequently in this order.

Note that in each phase (step, resp.), the message blocks and hash values generated in the previous phases (steps, resp.) are utilized, recycled, one could say. This means that in our method the excessive growth of the message complexity can be prevented. This is verified in the next subsection.

2.3 Diamond Structure Construction Method: The Pseudocode

1. **Input:** $d \in \mathbb{N}_+$, $(1 < d < \frac{n}{2})$; $H_d \subseteq \{0,1\}^n$, $|H_d| = 2^d$
2. **for** $i = d$ **downto** 2 **do** {*Jumps* $\mathtt{J}(d)$, $\mathtt{J}(d-1)$, ..., $\mathtt{J}(2)$.}
   {*Input to jump* $\mathtt{J}(i)$*: a set* H_i *of* 2^i *distinct hash values.*}
   2.1. $A_{i,0} := H_i$

2.2. Generate a set $M_{i,0} \subseteq \{0,1\}^m$ such that $|M_{i,0}| = 2^{\frac{n-i}{2}-1}$ and $|f(A_{i,0}, M_{i,0})| = 2^{\frac{n+i}{2}-1}$. {*Initialization*}

2.3. $H_{i,0} = f(A_{i,0}, M_{i,0})$; $B_i := \emptyset$

2.4. **for** $j = i$ **downto** 2 **do** {*Phases* $\mathtt{P}(i,i)$, $\mathtt{P}(i,i-1)$, ..., $\mathtt{P}(i,2)$.}
{*Input to phase* $\mathtt{P}(i,j)$*: The sets* B_i, $A_{j,0}$, $M_{j,0}$, *and* $H_{j,0}$}

2.4.1. **for** $k = 0$ **to** $2^{j-2}-1$ **do** {*Steps* $\mathtt{S}(i,j,0)$, $\mathtt{S}(i,j,1)$, ..., $\mathtt{S}(i,j,2^{j-2}-1)$.}
{*Input to* $\mathtt{S}(i,j,k)$*: the sets* $A_{j,k} \subseteq \{0,1\}^n$, $M_{j,k} \subseteq \{0,1\}^m$, $H_{j,k} = f(A_{j,k}, M_{j,k})$, *and* B_i *such that* $|A_{j,k}| = 2^j - 2k$, $|M_{j,k}| = s_{j,k}$, *and* $|H_{j,k}| = |A_{j,k}| \cdot |M_{j,k}|$.}

a. Generate a set $M'_{j,k} \subseteq \{0,1\}^m$ of cardinality $\lceil s_{j,k+1} - s_{j,k} \rceil$ such that $M'_{j,k} \cap M_{j,k} = \emptyset$ and $|f(A_{j,k}, M'_{j,k})| \geq 2^{\frac{n-j}{2}+1}$.

b. Search distinct hash values $h_{j,k}, h'_{j,k} \in A_{j,k}$ and message blocks $x_{j,k} \in M_{j,k}$, $x'_{j,k} \in M'_{j,k}$ such that $f(h_{j,k}, x_{j,k}) = f(h'_{j,k}, x'_{j,k})$.

c. $A_{j,k+1} = A_{j,k} \setminus \{h_{j,k}, h'_{j,k}\}$; $M_{j,k+1} = M_{j,k} \cup M'_{j,k}$; $H_{j,k+1} = f(A_{j,k+1}, M_{j,k+1})$; $B_i = B_i \cup \{(h_{j,k}, x_{j,k}), (h'_{j,k}, x_{j,k})\}$

d. **if** $k = 2^{j-2}-1$ **then**
(i) $A_{j-1,0} := A_{j,2^{j-2}-1}$, $M_{j-1,0} := M_{j,2^{j-2}-1}$; $H_{j-1,0} := H_{j,2^{j-2}-1}$

{*Input to phase* $\mathtt{P}(i,1)$*: the set* $A_{1,0} := \{h_{1,0}, h'_{1,0}\}$ *of two distinct hash values.*}

2.5. Generate a set $M'_{1,0} \subseteq \{0,1\}^m$ of $2^{\frac{n}{2}}$ message blocks such that there exist $x_{1,0}, x'_{1,0} \in M'_{1,0}$ for whicch $f(h_{1,0}, x_{1,0}) = f(h'_{1,0}, x'_{1,0})$. {*Phase* $\mathtt{P}(i,1)$.}

2.6. $B_i := B_i \cup \{(h_{1,0}, x_{1,0}), (h'_{1,0}, x'_{1,0})\}$; $H_{i-1} := \{f(h,x) \mid (h,x) \in B_i\}$

{*Input to jump* $\mathtt{J}(1)$*: the set* $H_1 := \{h_1, h_2\}$ *of two distinct hash values.*}

3. Generate a set $M_1 \subseteq \{0,1\}^m$ of $2^{\frac{n}{2}}$ message blocks such that there exist $x_1, x_2 \in M_1$ for which $f(h_1, x_1) = f(h_2, x_2)$. {*Jump* $\mathtt{J}(1)$.}

4. $B_1 := \{(h_1, x_1), (h_2, x_2)\}$; $H_0 := \{h_0\}$ where $h_0 = f(h_1, x_1) = f(h_2, x_2)$

5. **Output:** $B_d, B_{d-1}, \ldots, B_1$

2.4 The Overall Message Complexity of the Construction

Let us first compute the message complexity of jump $\mathtt{J}_d$; recall that it consists of phases $\mathtt{P}_d$, $\mathtt{P}_{d-1}$, ..., $\mathtt{P}_2$, $\mathtt{P}_1$. Certainly the complexity of jump $\mathtt{J}_d$ is the sum of the expected number of compression function enquieries in $\mathtt{I}(d)$ and the phases $\mathtt{P}_d$, $\mathtt{P}_{d-1}$, ..., $\mathtt{P}_2$, $\mathtt{P}_1$. Applying Lemma 1 and induction on j and k that $s_{j,k} \geq \frac{2^{\frac{n+j}{2}-1}}{2^j - 2k}$ holds for each $j \in \{2,3,\ldots,d\}$ and $k \in \{0,1,\ldots,2^{j-2}-1\}$. This means that the complexity analysis given in the description of step $\mathtt{S}(d,j,k)$ holds. This implies, that given $j \in \{2,3,\ldots,d\}$, the expected number of compression function queries to carry out phase $\mathtt{P}_j$ is at most (a small multiple of) $2^{j-2} \cdot 2^{\frac{n-j}{2}+1}$; here 2^{j-2} naturally refers to the number of steps in the phase. The complexity of $\mathtt{I}(d)$ is approximately $2^{\frac{n+d}{2}-1}$ and of $\mathtt{P}_1$ approximately $2 \cdot 2^{\frac{n}{2}}$.

The total complexity of jump $\mathtt{J}_d$ is thus

$$\mathrm{comp}(\mathtt{J}_d) \leq a \cdot \{2^{\frac{n+d}{2}-1} + \sum_{i=2}^{d} [2^{j-2} \cdot 2^{\frac{n-j}{2}+1}] + 2 \cdot 2^{\frac{n}{2}}\} \leq 2a \cdot 2^{\frac{n+d}{2}}$$

where a is a positive rational smaller than 2 and certainly independent of both n and d.

Remark 1. As noted above, the probability to find a colliding pair in each step $\mathtt{S}(d, j, k)$is greater than 0.5. It can be shown that we can choose the constant a to be approximately $\frac{e}{e-1}$ when 2^n is sufficiently large.

By the cosiderations above, we can deduce that the complexity of jump $\mathtt{J}_i$ is at most $2a \cdot 2^{\frac{n+i}{2}}$ for $i = 2, 3, \ldots, d$. Since running the jump $\mathtt{J}_1$ takes approximately $2 \cdot 2^{\frac{n}{2}}$ compression function queries, the overall message complexity of our diamond structure construction is not more than

$$2a \cdot \big[\sum_{j=2}^{d} 2^{\frac{n+j}{2}} + 2 \cdot 2^{\frac{n}{2}}\big] \leq 8\, a \cdot 2^{\frac{n+d}{2}} \quad .$$

2.5 Reducing the Complexity

It is quite easy to slightly reduce the complexity of the first pairing (i.e., $\mathtt{J}_1$) if we can choose the chaining values freely. One can choose an arbitrary hash value set A such that $|A| = 2^{\frac{n+d}{2}}$. After this we can fix a single message block x and compute the value $f(h, x)$ for all $h \in A$. Thus we have $2^{\frac{n+d}{2}}$ hash values and the number of possibly colliding pairs is

$$\binom{2^{\frac{n+d}{2}}}{2} = 2^{n+d-1} - 2^{\frac{n+d}{2}-1} \approx 2^{n+d-1}.$$

Since the codomain of f consists of 2^n elements, there should be approximately 2^{d-1} pairs $h, h' \in A$ such that $f(h, x) = f(h', x)$. We have now found 2^{d-1} colliding pairs with the approximate complexity $2^{\frac{n+d}{2}}$ (instead of $2\,a \cdot 2^{\frac{n+d}{2}}$).

As stated before the method presented in this section does not give us optimal complexity. A more effacious approach would be to create new message blocks one by one, to compute the respective hash values, and to search for colliding pairs after each new message block. However, we certainly still need to apply the compression function at least $2^{\frac{n+d-i}{2}}$ times to create 2^{d-i} pairs and so the total message complexity of our diamond structure construction will thus not drop below $\mathrm{O}(2^{\frac{n+d}{2}})$.

3 Trojan Message Attacks on Merkle-Damgård Structure

As mentioned before, the Trojan message attack was first presented in [1]. A Trojan message is a nonempty string t produced offline by the attacker and

given to the victim. The victim then chooses some word x from a fixed set P of prefixes (also known to the attacker) and forms the word $x\,t$. The attacker's task is to find a second preimage for $x\,t$. Extra constraints may be imposed to the structure of the preimage depending on the type of the Trojan attack.

In practise a Trojan message attack could happen for example in the following situation. Two parties $\mathcal{A}$ (the attacker) and $\mathcal{B}$ (the victim) are forming a contract and $\mathcal{B}$ is satisfied when choosing the first part of the contract from some set of precreated messages; then $\mathcal{A}$ is free to create the rest of the contract to be a Trojan message t. A situation like this could occur, for instance, when $\mathcal{A}$ does not know the exact day when the contract will be signed, but is allowed to otherwise formalize its details.

Inspired by the results in [1], we launch the following security property:

Trojan message resistance. Given any finite message set P, where $|P| > 1$, it is computationally infeasible to find a message t and a message set M such that $|M| = |P|$ and for each $p \in P$ there exists $m \in M$ such that $H(p\,t) = H(m)$.

Assume for a moment that $\mathtt{H} : \{0,1\}^* \to \{0,1\}^n$ is a random oracle hash function and P is a set of messages with cardinality $k \in \mathbb{N}_+$. Suppose that M is another set of messages such that $|M| = 2^s$ for some $s \in \mathbb{N}_+$. The probability that a random message t satisfies the property: for each $p \in P$ there exists $m \in M$ satisfying $\mathtt{H}(p\,t) = \mathtt{H}(m)$, is approximately $\left(\frac{2^s}{2^n}\right)^k$. Assume now that we create a new message set T where $|T| = 2^j$, $j \in \mathbb{N}_+$. The expected number of messages $t \in T$ such that for each $x \in P$ there exists $y \in M$ satisfying $\mathtt{H}(xt) = \mathtt{H}(y)$ is $2^j \cdot \left(\frac{2^s}{2^n}\right)^k = 2^{j+k\,s-k\,n}$. In order to succesfully complete the attack we should be able to create at least one Trojan message satisfying the given conditions, so the above expected number of messages should certainly be at least one. This means that $j + k\,s - k\,n \geq 0$.

The number of hash function queries needed is certainly in $\Omega(2^j + 2^s)$. We can minimize the complexity by setting $j = s = \frac{k\,n}{k+1}$. So the number of hash function queries needed is in $\Omega(2^{\frac{k}{k+1}\cdot n})$. It is interesting to see, that this is almost equal to the number of hash function queries needed to create a $k-$collision [15].

From now on we consider Trojan message attacks on Merkle-Damgård hash functions; recall that $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ is our compression function and $f^* : \{0,1\}^n \times (\{0,1\}^m)^* \to \{0,1\}^n$ its iterative extension. Assume furthermore that $h_0 \in \{0,1\}^n$ the initial hash value and $P = \{p_1, p_2, \cdots, p_{2^r}\}$ is the set of prefixes, $r \in \mathbb{N}_+$. Moreover, denote $h_{0,i} := f^*(h_0, p_i)$ for $i = 1, 2, \cdots, 2^r$. For the sake of simplicity we will assume that all the prefixes in P are of equal length k, $k \in \mathbb{N}_+$.

As mentioned before, the Andreeva et al [1] offered two variants of Trojan message attacks against Merkle-Damgård structure: the Collision Trojan Attack (abbr. ColTrA) and Herding Trojan Attack (abbr. HerTrA). Both attacks are comprised of three general phases. It is assumed that both the attacker and victim are familiar with the compression function f, the initial hash value h_0 and the prefix set P.

1. The attacker, Trudy, creates a Trojan message t. The complexity of this phase is the *offline complexity* of the attack.
2. The victim, Alice, chooses a prefix message p from the prefix set P, where $|P| = 2^r$.
3. Trudy creates a second preimage for pt. The complexity of this phase is the *online complexity* of the attack.

3.1 The Collision Trojan Attack

The first phase of ColTrA consists of 2^r step. In the first step the Trudy creates a message block pair x_1, y_1 such that $f(h_{0,1}, x_1) = f(h_{0,1}, y_1)$, $x_1 \neq y_1$. In the step i of the attack, where $i \in \{2, 3, \cdots, 2^r\}$, the attacker computes the value $h_{i-1} = f(h_{0,i}, x_1 x_2 \cdots x_{i-1})$ and creates a message block pair x_i, y_i such that $f(h_{i-1}, x_i) = f(h_{i-1}, y_i)$ and $x_i \neq y_i$. The attacker chooses then the word $t = x_1 x_2 \cdots x_{2^r}$ for the Trojan message and has thus completed the offline phase of the attack.

Assume now that in the second phase Alice chooses a prefix p_j and forms the word $p_j\, t$. The word $p_j\, t$ is passed to Trudy.

In the third phase the attacker first sets $t' := x_1 x_2 \cdots x_{j-1} y_j x_{j+1} \cdots x_{2^r}$ and then offers the word $p_j\, t'$ for a second preimage to $p_j\, t$. The attack is successful, since obviously $f(h_0, p_j\, t) = f(h_0, p_j\, t'))$. The offline complexity of this attack is $\mathrm{O}(2^{\frac{n}{2}+r})$ while the online complexity is negligible.

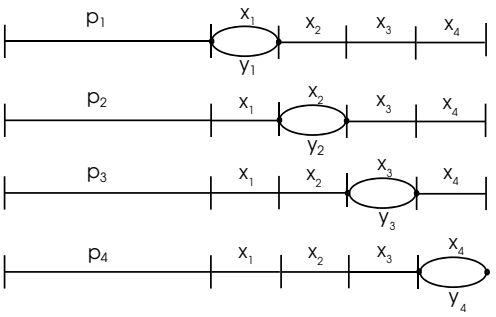

Fig. 1. Example of the Collision Trojan Attack when $r = 2$

3.2 The Herding Trojan Attack (HerTrA)

In first phase the attacker, Trudy, creates a diamond structure, with 2^d chaining values. The complexity of this operation is $\mathrm{O}(2^{\frac{n+d}{2}})$. Assume now that the final value of the structure is h'. Now Trudy creates a message x_0 such that $|x_0| = d$. Next she searches for message block pair x_1, y_1 such that $f(h_{0,1}, x_0 x_1) = f(h', y_1)$, and then sets $h_1 := f(h_{0,1}, x_0 x_1)$.

In the step i of the first phase, where $i \in \{2, 3, \cdots, 2^r\}$, Trudy computes the value $h_{i-1,i} := f(h_{0,i}, x_0 x_1 \cdots x_{i-1})$ and creates a message block pair x_i, y_i such that $f(h_{i-1,i}, x_i) = f(h_{i-1}, y_i)$. Then she simply sets $h_i := f(h_{i-1,i}, x_i)$ and is ready to proceed to next step. Finally Trudy creates the Trojan message $t = x_0 x_1 \cdots x_{2^r}$ and has finished the second phase.

Assume now that the attacker is challenged both with a prefix p_j, $j \in \{1, 2, \ldots, 2^r\}$ and a second prefix w such that the length $|w|$ of w is smaller than k. Trudy now searches for a connection message z such that $|wz| = k$ and $f(h_0, wz)$ is equal to some for chaining value of the created diamond structure. Assume now that message u is the path from this chaining value to the root hash value h' of the diamond structure, i.e. $f(h_0, wzu) = h'$.

Now we have $f(h_0, wzuy_1y_2 \cdots y_j) = h_j = f(h_0, p_j x_0 x_1 \cdots x_j)$ so clearly $wzuy_1y_2 \cdots y_j x_{j+1} x_{j+2} \cdots x_{2^r}$ is a second preimage for the word $p_j t$.

The complexity of creating a diamond structure is $\mathrm{O}(2^{\frac{n+d}{2}})$ so the complexity of the offline phase is $\mathrm{O}(2^{\frac{n}{2}+r} + 2^{\frac{n+d}{2}})$ while the complexity of finding z is 2^{n-d} which means that the complexity of the online phase is also 2^{n-d}. If we want to minimize the total complexity, we can set $d = \frac{n}{3}$ and get the total complexity of $\mathrm{O}(2^{\frac{n}{2}+r} + 2^{\frac{2n}{3}})$.

It is easy to see that the complexity of this kind of attack is in $\mathrm{O}(2^{\frac{2n}{3}})$, as long as the number of possible preimages is at most $2^{\frac{n}{6}}$, while the length of the created message is $k + d + 2^r$. If the number of possible preimages is larger than $2^{\frac{n}{6}}$ the complexity exceeds $2^{\frac{2n}{3}}$.

In comparison the second preimage attack presented in [8] and second preimage attack based on diamond structure presented in [2] against message with length $2^{\frac{n}{6}}$ would have the complexity $\mathrm{O}(2^{\frac{5n}{6}})$.

4 New Versions of the Trojan Message Attacks

4.1 The Weak Trojan Attack (WeaTrA)

We shall now present a new variant of the Collision Trojan Attack. The complexity of our construction is lower than that of the original one, while it gives the attacker more freedom to choose the content of the created second preimage. To ensure this we will assume that the attacker is, in addition to the prefix choice p of the victim from the set P, challenged with another prefix v from a set V such that $|V| \leq 2^r$ in the second phase of the attack. The attacker, Trudy, now has to find suffix s such that $f(h_0, vs) = f(h_0, pt)$, where t is the Trojan message created by Trudy in the first phase.

In the first, offline phase Trudy creates a diamond structure with chaining values $h_{0,1}, h_{0,2}, \cdots, h_{0,2^r}$. The complexity is certainly $\mathrm{O}(2^{\frac{n+r}{2}})$. Assume now that the final, root hash value of the diamond structure is h'. Next the attacker creates an expandable message, starting from the hash value h', with minimum block length $r + 1$ and maximum block length $2^{r+1} + r$. The complexity of this effort is $\mathrm{O}((r + 1) \cdot 2^{\frac{n}{2}})$ [8]. Assume that the final hash value of expandable message is h''.

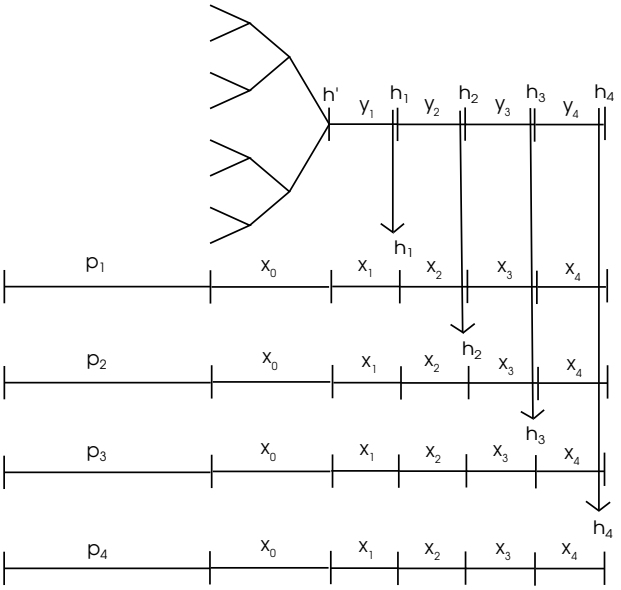

Fig. 2. Example of offline phase in Herding Trojan Attack when $r = 2,\ d = 3$

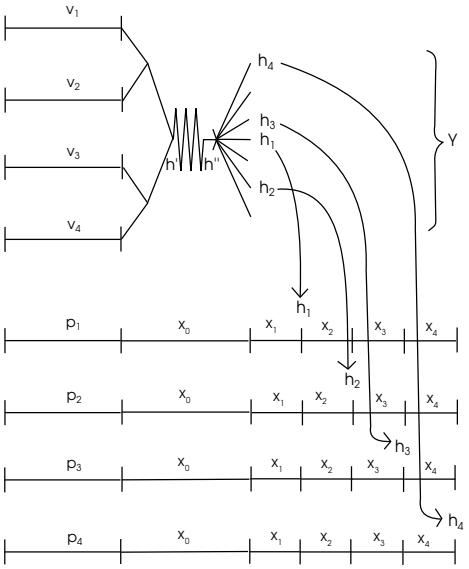

Fig. 3. Weak Trojan Message Attack example when $r = 2$

The attacker now creates a set Y consisting of $2^{\frac{n+r}{2}}$ random message blocks and computes the respective hash values $f(h'', y)$ for each $y \in Y$. This requires approximately $2^{\frac{n+r}{2}}$ compression function queries. In addition, the attacker now chooses any message x_0 such that the length of x_0 is $2r + 1$.

Now Trudy searches for a message block x_1 such that $f(h_{0,1}, x_0x_1) = f(h'', y)$ for some $y \in Y$. Denote $h_1 := f(h_{0,1}, x_0x_1)$. The complexity of finding such an x_1 is approximately $2^{\frac{n-r}{2}}$. The attacker sets $y_1 := y$ and is now ready for the second step of the first phase.

Consider the step $i \in \{2, 3, \cdots, 2^r\}$ of the first phase of the attack. Trudy computes $h'_{i-1} := f(h_{0,i}, x_0x_1x_2 \cdots x_{i-1})$ and searches for a message block x_i such that $f(h'_{i-1}, x_i) = f(h'', y)$ for some $y \in Y$. Once again the complexity of finding x_i is $2^{\frac{n-r}{2}}$. Denote $h_i := f(h'_{i-1}, x_i)$ and $y_i := y$. Once the attacker has completed the 2^r steps, the offline phase is done. The attacker now forms the Trojan message $t = x_0x_1x_2 \cdots x_{2^r}$.

In the second phase the victim picks from P the prefix p_j, where $j \in \{1, 2, \ldots, 2^r\}$. The attacker is also challenged with a second prefix $v \in V$. Assume that z is the expandable message with length $j+l$ and y is the path from $f^*(h_0, v)$ to h', i.e., $f^*(f^*(h_0, v), y) = h'$. Obviously $f^*(h_0, p_jx_0x_1 \cdots x_{2^r}) = f^*(h_0, vyzy_jx_{j+1}x_{j+2} \cdots x_{2^r})$, so clearly $vyzy_jx_{j+1}x_{j+2} \cdots x_{2^r}$ is a second preimage for p_jt.

The messages created in this way have the length $k + 2r + 1 + 2^r$. The offline complexity of this attack is $O(2^{\frac{n+r}{2}})$ while the online complexity is negligible. Since ColTrA has the complexity in $O(2^{\frac{n}{2}+r})$, the advantage of the WeaTrA is obvious.

4.2 The Strong Trojan Attack (StrTrA)

We shall use both *expandable messages* [8] and *elongated diamond structures* [7] to reduce the complexity of the original HerTrA.

The attacker, Trudy, begins the first phase of the attack by creating a random message $z = z_1z_2 \cdots z_{2^s}$, where $s \geq r$ and $z_1, z_2, \ldots, z_{2^s}$ are message blocks. Then she chooses random hash values $b_1, b_2, \cdots b_{2^d}$ (where $d \in \mathbb{N}_+, d \leq s$), computes $a_i := f^*(b_i, z)$ for $i = 1, 2, \ldots 2^d$, and creates a diamond structure with chaining values $a_1, a_2, \ldots, a_{2^d}$. The number of compression function queries needed is $O(2^{\frac{n+d}{2}})$. Assume that the final root hash value of the structure is h'. Trudy then continues by constructing an expandable message, starting from the hash value h', with minimun length $s + 1$ and maximum length $s + 1 + 2^{s+1}$. The complexity of the construction is $O((s + 1) \cdot 2^{\frac{n}{2}})$. Suppose that the final hash value of the expandable message is h''.

Trudy now creates a set Y containing $2^{\frac{n+r}{2}}$ random message blocks and computes all the hash values $f(h'', y)$, $y \in Y$. She also chooses an arbitrary message x_0 of length $2^s + d + s + 1$, and searches a message block x_1 such that $f^*(h_{0,1}, x_0x_1) = f(h'', y)$ for some $y \in Y$. The complexity of finding such x_1 and y is $O(2^{\frac{n-r}{2}})$. Denote $h_1 := f^*(h_{0,1}, x_0x_1)$ and $y_1 := y$.

Let $i \in \{2, 3, \cdots, 2^r\}$. In the step i of the first phase of the attack, Trudy computes $h'_{i-1} := f^*(h_{0,i}, x_0x_1x_2 \cdots x_{i-1})$ and searches for a message block

x_i such that $f(h'_{i-1}, x_i) = f(h'', y)$ for some $y \in Y$. To find such x_i and y takes approximately $2^{\frac{n-r}{2}}$ compression function queries. Finally Trudy sets $h_i := f(h'_{i-1}, x_i) = f(h'', y)$ and $y_i := y$.

After the 2^r steps our attacker chooses $t := x_0 x_1 x_2 \cdots x_{2^r}$ for the Trojan message and has completed the first (offline) phase of the attack. Since there were alltogether 2^r steps above, the complexity of completing them all is $O(2^{\frac{n+r}{2}})$. This means that the total complexity of the offline phase is $O(2^{\frac{n+d}{2}} + 2^{\frac{n+r}{2}})$.

Assume now that the attacker is challenged with a prefix $p_j \in P$ (chosen by the victim, Alice) and another (arbitrary) prefix p with length smaller than k. The attacker now searches for a connection message x such that length of px is k, and $f(h_0, px) = l$ for some hash value l that satisfies the condition $l = f(b_i, z_1 z_2 \cdots z_k)$ for some $i \in \{1, 2, \cdots d\}$ and $k \in \{1, 2, \cdots 2^s\}$. Assume now that message y is the path from l to the hash value h' in the diamond structure, i.e., $f(h_0, pxy) = h'$; the length of y is clearly $2^s + d - k$. Assume furthermore, that w is the expandable message chosen so that the total length of the message $pxywy_j x_{j+1} x_{j+2} \cdots x_{2^r}$ is $2^s + 2^r + k + d + s + 1$.

Now $f(h_0, pxywy_j) = f(h_0, p_j x_0 x_1 \cdots x_j) = h_j$ so $pxywy_j x_{j+1} x_{j+2} \cdots x_{2^r}$ is a second preimage for the message $p_j t$. The length of both messages is $2^s + 2^r + k + d + s + 1$.

The complexity of finding x is $O(2^{n-d-s})$ which means that the complexity of the online phase is also in $O(2^{n-d-s})$. If we want to minimize the total complexity we can choose $d = \frac{n-2s}{3}$ which means that the total complexity of the attack is $O(2^{\frac{2n-s}{3}} + 2^{\frac{n+r}{2}})$.

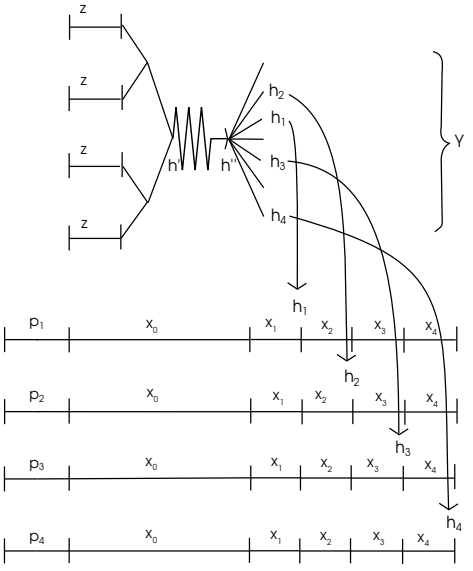

Fig. 4. Offline phase for Strong Trojan Message Attack example when $r = 2$

This means of course, that if we are able to create longer messages we can reduce the total complexity of the attack. Ideally we could choose $s = \frac{n-3r}{2}$ giving us the total complexity of $O(2^{\frac{n+r}{2}})$ i.e. the same as in the weak version of the attack. For exampla in SHA-1 the maximum lenght of the message is 2^{54} message blocks while $n = 160$. This implies that if, for example $r = 20$ we will have $2^{\frac{n+r}{2}} = 2^{90}$ if we can choose the length of the message to be 2^{50} message blocks. Creating a basic second preimage attacks, presented in [8] and [2], against messages with that length would have complexity greater than 2^{110}, while the complexity of previous version of strong Trojan message attack woud be approximately 2^{107}.

In practice messages are of course far shorter. However if we are able to choose, for example $s = \frac{n}{5}$, we would have the total complexity of $O(2^{\frac{3n}{5}})$ in comparison to $O(2^{\frac{4n}{5}})$ offered by ordinary second preimages against messages with length $2^{\frac{n}{5}}$, while $s = \frac{n}{11}$ would give us complexity $O(2^{\frac{7n}{11}})$ in comparison to $O(2^{\frac{10n}{11}})$.

5 Conclusion

In this paper we have presented a better and more efficient versions of Trojan message attacks. By using expandable messages and elongated diamond structures we have been able to reduce the complexity needed to create Trojan message attack significantly. We have also proven that for random oracle hash function the Trojan message complexity should be at least in $\Omega(2^{\frac{r}{r+1} \cdot n})$. Furhter study is needed to show if it is possible to create even more efficient Trojan message attacks or implement them in practice.

Diamond Structure Creation Method	Message Complexity
Blackburn & all	$O(\sqrt{d}2^{\frac{n+d}{2}})$
New Method	$O(2^{\frac{n+d}{2}})$

Trojan Message Attack Type	Message Complexity	AttCon
Second preimage attack	$O(2^{n-s})$	Any prefix
ColTrA	$O(2^{\frac{n}{2}+r})$	-
HerTrA	$O(2^{\frac{2n}{3}} + 2^{\frac{n}{2}+r})$	Any prefix
WeaTrA	$O(2^{\frac{n+r}{2}})$	Restricted prefix
StrTrA	$O(2^{\frac{2n-s}{3}} + 2^{\frac{n+r}{2}})$	Any prefix

Message complexities for Trojan message attacks when the length of the second preimage is in $O(2^s)$ and the size of the prefix set is 2^r, where $r < s$. Second preimage attack means the attack presented by Kelsey and Schneier [8] AttCon refers to the controll attacker has over created second preimage in online phase. Restricted prefix means that the attacker can choose the prefix of the second preimage from the pregenerated set with 2^r prefixes. Any prefix means that only the length of the prefix is restricted.

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.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. Blackburn, S., Stinson, D., Upadhyay, J.: On the Complexity of the Herding Attack and Some Related Attacks on Hash Functions. Cryptology ePrint Archive, Report 2010/030 (2010), `http://eprint.iacr.org/2010/030`
4. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
5. Hoch, J., Shamir, A.: Breaking the ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 179–194. Springer, Heidelberg (2006)
6. 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)
7. 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)
8. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
9. Kortelainen, J., Halunen, K., Kortelainen, T.: Multicollision attacks and generalized iterated hash functions. Journal of Mathematical Cryptology 4, 239–270 (2010)
10. Kortelainen, J., Kortelainen, T., Vesanen, A.: Unavoidable Regularities in Long Words with Bounded Number of Symbol Occurrences. In: Fu, B., Du, D.-Z. (eds.) COCOON 2011. LNCS, vol. 6842, pp. 519–530. Springer, Heidelberg (2011)
11. Kortelainen, T., Vesanen, A., Kortelainen, J.: Generalized Iterated Hash Functions Revisited: New Complexity Bounds for Multicollision Attacks. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 172–190. Springer, Heidelberg (2012)
12. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A. (eds.): Handbook of Applied Cryptology, pp. 321–376 (1996)
13. Merkle, R.: A Certified Digital Signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
14. Nandi, M., Stinson, D.: Multicollision attacks on some generalized sequential hash functions. IEEE Transactions on Information Theory 53(2), 759–767 (2007)
15. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday paradox for multicollisions. IEICE Transactions 91A(1), 39–45 (2008)