A Synthetic Indifferentiability Analysis of Some Block-Cipher-Based Hash Functions*

Zheng Gong, Xuejia Lai and Kefei Chen
Department of Computer Science and Engineering
Shanghai Jiaotong University, China
neoyan@sjtu.edu.cn,{lai-xj,chen-kf}@cs.sjtu.edu.cn

Abstract

At *ASIACRYPT'06*, Chang *et al.* analyzed the indifferentiability of some popular hash functions based on block-cipher, namely the twenty collision resistant PGV, MDC2 and PBGV hash functions etc. In particular, two indifferentiable attacks were presented on the four of the twenty collision resistant PGV and the PBGV hash functions with the prefix-free padding. In this article, a synthetic indifferentiability analysis of some block-cipher-based hash functions is considered. First, a more precise definition is proposed on the indifferentiability adversary in block-cipher-based hash functions. Next, the advantage of indifferentiability is extended by considering whether the hash function is keyed or not. Finally, a limitation is observed in Chang et al.'s indifferentiable attacks on the four PGV and the PBGV hash functions. The formal proofs show the fact that those hash functions are indifferentiable from random oracle in the ideal cipher model with the prefix-free padding, the NMAC/HMAC and the chop construction.

1 Introduction

Block-Cipher-Based Hash Function. A cryptographic hash function $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ maps an infinite set of inputs $\mathcal{M}$ to the finite set of n-bit outputs $\mathcal{Y}$. It is one of the most important primitives in cryptography

*This paper is supported by NSFC under the grants 60573032, 90604036 and National 863 Projects 2006AA01Z422

to provide a unique "fingerprint" on a certain information. The design of today's cryptographic hash functions still follows the Merkle-Damgard(MD) structure[18, 9], by iterating a round function on the input message. The hash function will be collision resistant if the round function is.

In practice, there are two main approaches for designing the round function in the MD structure. First, the most in-use hash functions, for example the MD5 and SHA-1, were constructed by iterating a *dedicated compression function*. There also exists a second setting for hash function design and analysis, in which one makes the round function out of a *block cipher*. A well-known advantage of the block-cipher-based approach is to minimize the effort in the design of a secure compression function. In the past decades, many hash functions were designed from block cipher, i.e., the PGV hash functions[22], the PBGV[21] and the MDC2[4]. A negative side of the block-cipher-based hash function is a decrease in speed. Still, the efficiency of the AES and the recent advances in collision finding[25, 26] motivate renewed interest in finding good ways to turn a block cipher into a cryptographic hash function. Instructive examples can be found in [16, 13].

Random Oracle Model. Random oracle model has been first introduced by Bellare and Rogaway as a "paradigm for design efficient protocols" [1]. It assumes that all parties, including the adversary, have access to a public, truly random oracle. This model becomes extremely useful since the schemes design under such model would be more simple and highly practical while compare with the standard model ones. In most applications, random oracle is an oracle that anybody can query but no one has control over it. This is according to a completely valid application of the random oracle (as explained in [1]). Then in some proofs, random oracle is considered to be under control of a simulator. The simulator can listen to any query made to the oracle, so he knows what queries were asked. Yet he has no control over the output, so the oracle still remains a truly random oracle[11]. Finally in some proofs, random oracle is considered to be under complete control of a simulator. The simulator can actually manipulate the answers the oracle gives, as long as the result is indistinguishable from a random oracle[3].

Since random oracle performs quite like cryptographic hash function, people is suggested to replace random oracle in the scheme with a "secure" dedicated hash function (e.g., SHA-1, SHA-256, etc) to preserve the security in the standard model. One has to be careful with the selection of the hash function, for example a specific vulnerability will be found when instantiates the random oracle with a "bad" hash function[5, 20]. Research on how to instantiate a random oracle with a certain hash function has been red hot in recent years. Many valuable references on this problem could not be indicated at a specific location: [6, 10, 12, 19]. All these researches take the hash function in the scheme as a *black-box*, which means the internal structure of the hash function is ignored. Since one can prove such scheme is secure in the random oracle model, solving those problems back to the efforts that design a cryptographic hash function to instantiate such a random oracle.

Indifferentiability Methodology. In [17], Maurer *et al.* first introduced a term "indifferentiability" and a formal model to "distinguish" whether a given construction has any difference from a heuristic random oracle. The indifferentiability has been focussed on the question: what conditions should be imposed on the round function $\mathcal{F}$ to make sure that the hash function $\mathcal{C}^{\mathcal{F}}$ satisfies the certain conditions of the random oracle. This approach is based on the fact that one of the problems in assessing the security of a hash function is caused by arbitrary length of input. It is clear that the weakness of $\mathcal{F}$ will generally result in the weakness of $\mathcal{C}^{\mathcal{F}}$, but the converse does not hold in general. The main problem is to derive such sufficient conditions. The indifferentiability between a hash function and a random oracle is a more rigorous *white-box* analysis which needs to expose the internal structure of the hash function, while the indistinguishability just requires a *black-box* analysis.

Recently, Coron *et al.*[8] first implemented the notion of indifferentiability for analysis of some classical MD variants. They analyzed that plain MD hash function can be differentiable from random oracle, then proved MD structure hash functions will be indifferentiable with the prefix-free, the HMAC/NMAC and the chop construction. In [7], Chang *et al.* continued this initial suggestion and analyzed the indifferentiability of some popular block-cipher-based hash functions with the prefix-free padding. In particular, a formal proof of indifferentiability were given on the twenty collision resistant PGV[2] and the PBGV[21] hash functions. On those indifferentiability results, they claimed that there are sixteen collision-resistant PGV hash functions are indifferentiable from random oracle in the ideal cipher model, while the remains four PGV hash functions are not. They also gave an indifferentiable attack on the PBGV hash function, and said by using the same idea one can find indifferentiable attacks on MDC2[4], QG-I, and LOKI-DBH[14] etc.

Our Contributions In this paper, a synthetic indifferentiability analysis of some block-cipher-based hash functions is considered. First, we propose a more precise definition of indifferentiability adversary in block-cipher-based hash functions. Next, we analyze the advantage of indifferentiability in keyed and unkeyed modes. The authors of [7] only focused on the collision event, not all of the indifferentiable events on hash function, e.g., preimage attack, second preimage attack etc. Moreover, they only analyzed the situation in unkeyed mode. Since keyed hash functions are receiving more and more attention, after the genius attacks were found in widely-used dedicated-key hash functions, such as MD4, MD5 and SHA-1[25, 26]. The indifferentiability analysis of keyed hash function will be necessary in both of theory and practice. Prior to the current work, we are unaware of any analysis on the advantage of indifferentiability for keyed hash function based on any block-cipher. Finally, we observe a limitation in Chang *et al.*'s indifferentiable attacks on the four PGV and the PBGV hash functions, which makes these attacks are not possible if one limits the message space to messages of at least two blocks. In particular, we formally prove the four PGV and the PBGV hash functions are indifferentiable from random oracle with the prefix-free padding, the HMAC/NMAC and the chop construction.

Organization. The remainder of this paper is organized as follows. In Section 2, we review the definitions and describe a more precise definition of the indifferentiability adversary in block-cipher-based hash functions. In Section 3, first, we analyze the advantage of indifferentiability in keyed and unkeyed modes. Then, we show a limitation in Chang *et al.*'s attacks on the four PGV and the PBGV hash functions. Finally, we give our indifferentiability analysis of the four PGV hash functions and the PBGV hash function. Section 4 gives a conclusion.

2 Preliminaries

Here we review the notation and definitions that will be used throughout the paper. Let the symbol $\oplus$ be the bitwise exclusive OR. For binary sequences a and b, $a||b$ denotes their concatenation. The i-th block of a message M is m_i and so $M = m_1||m_2||\cdots||m_{|M|/n}$, where n is the block length. Let IV be the initial value. The same terminology and abbreviations in different definitions are the same meaning, except there are special claims in the context.

2.1 Ideal Cipher Model

Let κ, n, ℓ be numbers. A *block cipher* is a keyed function $E : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$. For each $k \in \{0,1\}^\kappa$, the function $E_k(\cdot) = E(k, \cdot)$ is a permutation on $\{0,1\}^n$. If E is a block cipher then E^{-1} denotes its inverse, where $E_k^{-1}(y) = x$ such that $E_k(x) = y$. Let $\texttt{Bloc}(\kappa, n)$ be the family of all block ciphers $E : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$. A *block-cipher-based* hash function is a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ and $E \in \texttt{Bloc}(\kappa, n)$ is the block cipher used in the round function of H. If $\ell = n$, then H is called a single block length(SBL) hash function, i.e., the PGV hash functions[22]. If $\ell = 2n$, then H is called a double block length(DBL) hash function, i.e., MDC2[4], QG-I, and LOKI-DBH[14].

Ideal cipher model is the formal model for the security analysis of block-cipher-based hash functions, which is dating back to Shannon [24] and widely used, i.e., in [2, 15, 22]. By choosing a block cipher $E \in \texttt{Bloc}(\kappa, n)$, an adversary is given access to two oracles E and E^{-1}. Thus, the i-th query-response r_i is a four-tuple that

$$r_i = (\sigma_i, k_i, x_i, y_i)$$

where $\sigma_i \in \{1, -1\}, k_i \in \{0,1\}^\kappa, x_i, y_i \in \{0,1\}^n$. If $\sigma_i = 1$ then the adversary queries (k_i, x_i) and gets response $y_i = E_{k_i}(x_i)$, otherwise he queries (k_i, y_i) and gets response $x_i = E_{k_i}^{-1}(y_i)$. Since $E_k(\cdot)$ is a permutation on $\{0,1\}^n$, it holds

$$Pr[E_{k_i}(x_i) = y_i] = Pr[E_{k_i}^{-1}(y_i) = x_i] = \frac{1}{n}.$$

In the ideal cipher model, the complexity of an attack is measured by the total number of the optimal adversary's queries to the two oracles E and E^{-1}.

2.2 Indifferentiability

Here we recall the definitions for the indifferentiability analysis [17].

Definition 2.1 *A Turing machine $\mathcal{C}$ with oracle access to an ideal primitive $\mathcal{F}$ is said to be (t_D, t_S, q, ϵ)-indifferentiable from an ideal primitive Rand if there exists a simulator $\mathcal{S}$, such that for any distinguisher $\mathcal{D}$ it holds the advantage of indifferentiability that:*

$$Adv(D) = |Pr[\mathcal{D}^{\mathcal{C},\mathcal{F}} = 1] - Pr[\mathcal{D}^{Rand,\mathcal{S}} = 1]| < \epsilon,$$

where $\mathcal{S}$ has oracle access to Rand and runs in polynomial time at most t_S, and $\mathcal{D}$ runs in polynomial time at most t_D and makes at most q queries. $\mathcal{C}^{\mathcal{F}}$ is said to be (computationally) indifferentiable from Rand if ϵ is a negligible function of the security parameter k (in polynomial time t_D and t_S).

It is proved in [17] that if $\mathcal{C}^{\mathcal{F}}$ is indifferentiable from *Rand*, then $\mathcal{C}^{\mathcal{F}}$ can replace *Rand* in any cryptosystem, and the resulting cryptosystem is at least as secure in the $\mathcal{F}$ model as in the *Rand* model. For example, if a block-cipher-based hash function $\mathcal{C}^{\mathcal{F}}$ is indifferentiable from a random oracle *Rand* in the ideal cipher model, then $\mathcal{C}^{\mathcal{F}}$ can replace *Rand* in any cryptosystem, while keeps the resulting system (with $\mathcal{C}^{\mathcal{F}}$) remaining secure in the ideal cipher model if the original system (with *Rand*) is secure in the random oracle model.

In this paper, hash function H denotes the Turing machine $\mathcal{C}^{\mathcal{F}}$ where the ideal primitive $\mathcal{F}$ is the round function of $\mathcal{C}$. Let E denote the block cipher used in the round function $\mathcal{F}$ and E^{-1} is its inverse. Since we focus on block-cipher-based hash functions in case of the ideal cipher model, $\mathcal{S}$ has to simulate the oracles E and E^{-1}. Therefore, any distinguisher $\mathcal{D}$ obtains the following rules: either the block-cipher E, E^{-1} is chosen at random and the hash function H is constructed from it, or the hash function H is chosen at random (*Rand*) and the simulated encryption and decryption oracles $\mathcal{S}, \mathcal{S}^{-1}$ are implemented by a simulator $\mathcal{S}$ with oracle access to *Rand*. Those two ways to build up a cryptographic hash function should be indifferentiable.

2.3 Indifferentiability Adversary in Block-Cipher-Based Hash Functions

For indifferentiability analysis of block-cipher-based hash functions, one needs to formally define an adaptive adversary's activities in those hash functions. In [7], Chang *et al.* just defined the adversary in hash functions based on dedicated compression function in the random oracle model. A more precise definition of the indifferentiability adversary in block-cipher-based hash functions is defined as follows.

Let $\mathcal{D}$ be a distinguisher and $\mathcal{S}$ be a simulator for indifferentiability analysis. By following Definition 2.1, the goal of $\mathcal{D}$ is to distinguish two cryptosystems $\mathcal{O}_1$ and $\mathcal{O}_2$, such that $\mathcal{O}_1 = (H, E, E^{-1})$ and $\mathcal{O}_2 = (Rand, \mathcal{S}, \mathcal{S}^{-1})$. $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ denotes a hash function constructed from block-cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K} \in \{0,1\}^\kappa, \mathcal{M} \in \{0,1\}^*$ and $\mathcal{Y} \in \{0,1\}^\ell$. *Rand* is a random oracle which has the same domain and range with H. h_i denotes the hash value of the i-th query. The function $Pad(\cdot)$ denotes the padding rule of the hash function H. Let $r_i \leftarrow (h_{i-1} \xrightarrow{m_i} h_i)$ be i-th query-response to the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$ where $m_i \in \{0,1\}^n$. $\mathcal{R}_i = (r_1, \cdots, r_i)$ denotes the query-response set on the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$ after i-th query. Let $r'_i \leftarrow (IV \xrightarrow{M} h_i)$ be i-th query-response to the oracles $(H, Rand)$ where $M \in \mathcal{M}$. $\mathcal{R}'_i = (r'_1, \cdots, r'_i)$ denotes the query-response set on the oracles $(H, Rand)$ after i-th query. A *functional closure* $\mathcal{R}^*$ on $\mathcal{R}$ is the set with the following conditions.

1. If $h_{i-1} \xrightarrow{m_i} h_i, h_i \xrightarrow{m_{i+1}} h_{i+1} \in \mathcal{R}_{i+1}$, then $h_{i-1} \xrightarrow{m_i||m_{i+1}} h_{i+1} \in \mathcal{R}^*_{i+1}$.
2. If $h_{i-1} \xrightarrow{m_i} h_i, h_{i-1} \xrightarrow{m_i||m_{i+1}} h_{i+1} \in \mathcal{R}_{i+1}$, then $h_i \xrightarrow{m_{i+1}} h_{i+1} \in \mathcal{R}^*_{i+1}$.

The $(H, Rand)$-query inputs an arbitrary length message and outputs a fixed length hash value, while the $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$-query inputs a fixed length plain-text or cipher-text and outputs the corresponding cipher-text or plain-text, respectively. The details of the two categories of queries are described below.

- **Query on** $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$:
  - For i-th query on $(E, \mathcal{S})$, distinguisher $\mathcal{D}$ queries $(1, h_{i-1}, m_i)$ and the response is $y_i = E_{h_{i-1}}(m_i)$ or $\mathcal{S}(h_{i-1}, m_i)$, where $y_i, m_i \in \{0,1\}^n$. By computing the hash value h_i from the tuple (y_i, h_{i-1}, m_i), the i-th query-response set $\mathcal{R}_i = \mathcal{R}_{i-1} \cup (h_{i-1} \xrightarrow{m_i} h_i)$.
  - For i-th query on $(E^{-1}, \mathcal{S}^{-1})$, distinguisher $\mathcal{D}$ queries $(-1, h_{i-1}, y_i)$ and the response is $m_i = E^{-1}_{h_{i-1}}(y_i)$ or $\mathcal{S}^{-1}(h_{i-1}, y_i)$, where $y_i, m_i \in \{0,1\}^n$. By computing the hash value h_i from the tuple (y_i, h_{i-1}, m_i), the i-th query-response set $\mathcal{R}_i = \mathcal{R}_{i-1} \cup (h_{i-1} \xrightarrow{m_i} h_i)$.
  - Let $\mathcal{R}_q$ be the query-response set of the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$ after the maximum q queries. According to the transitive and substitute properties of $\mathcal{R}_q$, the functional closure set $\mathcal{R}^*_q$ is the complete view of distinguisher $\mathcal{D}$ on the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$.

- **Query on** $(H, Rand)$:
  - For i-th query on $(H, Rand)$, distinguisher $\mathcal{D}$ selects an arbitrary length message $M_i \in \mathcal{M}$ and a key $k_i \in \mathcal{K}$. Thus, the query on hash functions will be (k_i, M_i). In particular, k_i equals a fixed value IV in unkeyed hash functions. The response of $H, Rand$ is $h_i = H(k_i, Pad(M_i))$ or $S(k_i, Pad(M_i)), h_i \in \mathcal{Y}$.
  - Let $\mathcal{R}'_i = \mathcal{R}'_{i-1} \cup (k_i \xrightarrow{M_i} h_i)$ be the query-response set on the oracles $(H, Rand)$ after i-th query. The query-response set $\mathcal{R}'_q$ is the complete view of distinguisher $\mathcal{D}$ on the oracles $(H, Rand)$ after the maximum q queries.

In indifferentiability analysis, all the repetition query will be ignored, i.e., $R_i = R_j$ or $R'_i = R'_j$ for any $i \neq j$. For simplicity, one can assume there is no such trivial query since it does not help anything as the view of distinguisher.

3 Indifferentiability Analysis of Some Block-Cipher-Based Hash Functions

In this section, a synthetic indifferentiability analysis of block-cipher-based hash functions is considered. First, we give a definition of the advantage of indifferentiability in block-cipher-based hash functions of keyed and unkeyed modes. Next, we observe a limitation in Chang *et al.*'s indifferentiable attacks on the four PGV and the PBGV hash functions, then formally prove the fact that those hash functions are indifferentiable from random oracle with the prefix-free padding, the HMAC/NMAC and the chop construction.

3.1 Advantage of Indifferentiability

The fact that the original advantage of indifferentiability presented by Chang *et al.* [7] is incomplete because it just covered the collision event, where there are some other indifferentiable events need to be totally considered. For an exact bound of the advantage, one has to carefully consider all the security events that will affect the advantage of indifferentiability. Based on the original analysis in [7] and the extended definition of adversary in block-cipher-based hash functions in Section 2.3, a more precise advantage of indifferentiability is analyzed as follows.

Let $Bad_i, i = 1, 2$ be the set of the indifferentiable events on the two cryptosystems $\mathcal{O}_1 = (H, E, E^{-1})$ and $\mathcal{O}_2 = (Rand, S, S^{-1})$, respectively. The oracles (H, E, E^{-1}) and $(Rand, S, S^{-1})$ are identically distributed conditioned on the past view of the distinguisher and Bad_i does not occur. If $\mathcal{D}$ is a distinguisher then we write $Adv(\mathcal{D})$ as a measure of the maximal differentiable advantage overall distinguishers $\mathcal{D}$. For brevity, D_1 denotes the event $\mathcal{D}^{H,E,E^{-1}} = 1$ and D_2 denotes the event $\mathcal{D}^{Rand,S,S^{-1}} = 1$. The function $Max()$ returns the biggest value of inputs. The advantage of indifferentiability of the two cryptosystems $\mathcal{O}_1 = (H, E, E^{-1})$ with $\mathcal{O}_2(Rand, S, S^{-1})$ is at most

$$
\begin{aligned}
Adv(\mathcal{D}) &= |Pr[\mathcal{D}^{H,E,E^{-1}} = 1] - Pr[\mathcal{D}^{Rand,S,S^{-1}} = 1]| \\
&= |(Pr[D_1 \cap Bad_1] + Pr[D_1 \cap \neg Bad_1]) \\
&\quad - (Pr[D_2 \cap Bad_2] + Pr[D_2 \cap \neg Bad_2])| \\
&= |(Pr[D_1|Bad_1] \times Pr[Bad_1] - Pr[D_2|Bad_2] \times Pr[Bad_2]) \\
&\quad + (Pr[D_1|\neg Bad_1] \times Pr[\neg Bad_1] - Pr[D_2|\neg Bad_2] \times Pr[\neg Bad_2])| \\
&\leq Max(Pr[Bad_1], Pr[Bad_2]]) \times |Pr[D_1|Bad_1] - Pr[D_2|Bad_2]| \\
&\quad + |Pr[D_1|\neg Bad_1] \times Pr[\neg Bad_1] - Pr[D_2|\neg Bad_2] \times Pr[Bad_2]| \\
&\leq Max(Pr[Bad_1], Pr[Bad_2]]) \times (1 + Pr[D_1|\neg Bad_1]) \\
&\leq 2 \times Max(Pr[Bad_1], Pr[Bad_2]]).
\end{aligned}
$$

Then we analyze the set of the indifferentiable events Bad_i in block-cipher-based hash functions. For unkeyed hash function, the events include the collision($Coll$), the second preimage(Sec) and the preimage(Pre). Because collision resistance implies second-preimage resistance, while separates from preimage resistance, then the set of the differentiable events in unkeyed hash functions is

$$Bad_i = \{Coll_i, Pre_i\}, i = 1, 2.$$

For keyed hash functions, there are more indifferentiable events need to be considered. Depends on the key and the challenge are fixed or random, one thus has seven sensible notions, which are named Pre, $ePre$, $aPre$, Sec, $eSec$, $aSec$, and $Coll$. The leading "a" in the name of a notion is meant to suggest always: if a hash function is secure for any fixed key, then it is *always* secure. The leading "e" in the name of a notion is meant to suggest everywhere: if a hash function is secure for any fixed challenge, then it is *everywhere* secure. According to the implications and separations of the seven security notions[23], collision resistance implies (always) second-preimage resistance and always/everywhere preimage resistance implies preimage resistance, the set of the indifferentiable events in keyed hash functions is

$$Bad_i^{key} = \{Coll_i, eSec_i, aPre_i, ePre_i\}, i = 1, 2.$$

For brevity, we ignore the description of those security notions and the proof of the implications and separations here. See [18, 23] for more details.

3.2 Indifferentiability of The Four PGV Hash Functions

In [7], Chang *et al.* first proved there are sixteen out of the 20 collision resistant PGV hash functions[2] which are indifferentiable from random oracle in the ideal cipher model with the prefix-padding. Then they designed two indifferentiable attacks on the four PGV hash functions and the PBGV, respectively. The authors of [7] claimed that the two attacks are not only possible with one-block message, but also more than one block. Furthermore, they said by using the same idea one can find indifferentiable attacks on some of the double block length hash functions, i.e., MDC2, QG-I, and LOKI-DBH etc. Here we show a limitation in their attacks, which exposed their attacks are rather artificial and only possible in the one-block padded message. Then we construct the simulations to prove that the four PGV and the PBGV hash functions are indifferentiable from random oracle in the ideal cipher model with the prefix-free padding, the NMAC/HMAC and the chop construction. First we give the analysis of the four PGV hash functions.

The four PGV hash functions are $E_{h_{i-1}}(m_i) \oplus m_i$ (PGV-17), $E_{h_{i-1}}(m_i \oplus h_{i-1}) \oplus m_i \oplus h_{i-1}$(PGV-18), $E_{h_{i-1}}(m_i) \oplus m_i \oplus h_{i-1}$(PGV-19), $E_{h_{i-1}}(m_i \oplus h_{i-1}) \oplus m_i$(PGV-20). Let $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a hash function constructed from block-cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ where $\mathcal{K} \in \{0,1\}^\kappa, \mathcal{M} \in \{0,1\}^*$ and $\mathcal{Y} \in \{0,1\}^\ell$. *Rand* is a random oracle which has the same domain and range with H. h_i denotes the hash value of the i-th query. The function $Pad(\cdot)$ denotes the prefix-free padding. Let $r_i \leftarrow (h_{i-1} \xrightarrow{m_i} h_i)$ be the i-th query-reponse to the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$ where $m_i \in \{0,1\}^n$. $\mathcal{R}_i =$

$(r_1, \cdots, r_i)$ denotes the query-response set on the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$ after i-th query and $\mathcal{R}^*$ is its functional closure. Let $r'_i \leftarrow (IV \xrightarrow{M} h_i)$ be the i-th query-response to the oracles $(H, Rand)$ where $M \in \mathcal{M}$. $\mathcal{R}'_i = (r'_1, \cdots, r'_i)$ denotes the query-response set on the oracles $(H, Rand)$ after i-th query. Let IV be the initial value. Chang *et al.*'s indifferentiable attack on PGV-17 is recalled in Fig 3.1.

> Distinguisher $\mathcal{D}$ can access to oracles $(\mathcal{O}_1, \mathcal{O}_2)$ where $\mathcal{O}_1 = (H, E, E^{-1})$ and $\mathcal{O}_2 = (Rand, \mathcal{S}, \mathcal{S}^{-1})$.
>
> 1. $\mathcal{D}$ selects a message M such that $Pad(M) = m$ and $|m| = n$, then he makes the query M to H and receives $H(M) = h_i$.
> 2. $\mathcal{D}$ makes an inverse query $(-1, h_{i-1}, h_i \oplus m)$ to S^{-1} and receives m^*, where $h_{i-1} = h_0 = IV$.
> 3. if $m = m^*$ output 1, otherwise output 0.

Fig 3.1 Chang *et al.*'s indifferentiable attack on PGV-17.

It is obvious that simulator $\mathcal{S}$ can return $m^* = m$ only with probability 2^{-n}, thus PGV-17 is differentiable from random oracle in the ideal cipher model. But their attack needs the key (h_{i-1}) in the first iteration is fixed (IV) or assumed known by the distinguisher. If one limits the message space to messages of at least two blocks, i.e., the prefix-free padding $Pad(M)$ returns $1||m_i$ if m_i is the last block, else returns $0||m_i$. Because distinguisher $\mathcal{D}$ only queried the hash value $h_i = H(M)$ from $(H, Rand)$, $\mathcal{D}$ cannot make an inverse query $(-1, h_{i-1}, h_i \oplus m)$ since the internal hash value h_{i-1} is unknown, and $\mathcal{D}$ only knows $(IV \xrightarrow{M} h_i) \in \mathcal{R}'_i$. If $\mathcal{D}$ queried the internal value h_{i-1} before, then $\mathcal{S}$ can track it since $h_{i-1} \in \mathcal{R}^*_i$. Therefore, Chang *et al.*'s indifferentiable attack on the four PGV hash functions is rather artificial and only possible with one-block message. In practice, the attack can be avoided by using some well-known MD variants which were proposed in [8], namely the prefix-free padding, the NMAC/HMAC and the chop construction, described in Fig 3.2.

Now we give a simulation to prove the fact that PGV-17 is indifferentiable from random oracle in the ideal cipher model with the prefix-free padding. To avoid some trivial attacks, the last block contains the length of input. Let q be the maximum times of oracle access and l is the maximum

Prefix-free MD(IV, M) $M = m_1\|\|\cdots\|\|m_i, h_0 = IV$ For $i = 1$ to i do $h_i = F(Pad(m_i), h_{i-1})$ Return h_i	NMAC Construction (IV, M) $M = m_1\|\|\cdots\|\|m_i, h_0 = IV$ For $i = 1$ to i do $h_i = F(m_i, h_{i-1})$ Return $Perm(h_i)$
HMAC Construction (IV, M) $M = m_1\|\|\cdots\|\|m_i, h_0 = IV$ For $i = 1$ to i do $h_i = F(m_i, h_{i-1})$ Return $h_{i+1} = F(h_i, IV)$	Chop Construction (IV, M) $M = m_1\|\|\cdots\|\|m_i, h_0 = IV$ For $i = 1$ to i do $h_i = F(m_i, h_{i-1})$ Return $Chop(h_i)$

Fig 3.2 Definitions of the four MD variants proposed in [8]. $Pad(m_i)$ is prefix-free padding, returns $1||m_i$ if m_i is the last block, else returns $0||m_i$. $Perm(x), x \in \{0,1\}^\ell$ is a random permutation in $\{0,1\}^\ell$. $Chop(x), x \in \{0,1\}^\ell$ returns first s-bit of x.

length of a query made by $\mathcal{D}$. Based on the definition in Section 2.3, the simulation is as follows.

- $Rand$**-Query.** For i-th $Rand$-query $M_i \in \mathcal{M}$, if M_i is a repetition query, retrieve $r'_j \leftarrow (IV \xrightarrow{M_i} h_j)$ where $r'_j \in \mathcal{R}'_{i-1}, j \leq i-1$, then return $Rand(M_i) = h_j$. Else randomly select a hash value $h_i \in \mathcal{Y}$ and update $\mathcal{R}'_i = \mathcal{R}'_{i-1} \cup \{IV \xrightarrow{M_i} h_i\}$, then return $Rand(M_i) = h_i$.
- $(\mathcal{S}, \mathcal{S}^{-1})$**-Query.** To answer distinguisher $\mathcal{D}$'s encryption and decryption queries, the simulator $\mathcal{S}$ responses as follows.
  1. On $\mathcal{S}$ i-th query $(1, h_{i-1}, m_i)$,
     (a) If $\exists IV \xrightarrow{M} h_{i-1} \in \mathcal{R}'_{i-1}$ and $Pad(M) = m_i$, run $Rand(M)$ and obtain the response h_i, update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_{i-1} \xrightarrow{m_i} h_i\}$, then return $h_i \oplus m_i$.
     (b) Else select a random value $h_i \in \mathcal{Y}$, $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_{i-1} \xrightarrow{m_i} h_i\}$, then return $h_i \oplus m_i$.
  2. On $\mathcal{S}^{-1}$ i-th query $(-1, h_{i-1}, c_i)$,
     (a) If $\exists IV \xrightarrow{M} h_{i-1} \in \mathcal{R}'_{i-1}$, run $Rand(M)$ and obtain the response h_i. Then if $c_i = h_i \oplus Pad(M)$, update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_{i-1} \xrightarrow{Pad(M)} h_i\}$ and return $m_i = Pad(M)$.
     (b) Else randomly select a message $m_i \in \mathcal{M}$, update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_{i-1} \xrightarrow{m_i} c_i \oplus m_i\}$ and return m_i.

Before stating the main result of the four PGV hash functions, the probability of the security events $Bad_i, i = 1, 2$ is analyzed for the two cryptosystems $\mathcal{O}_1$ and $\mathcal{O}_2$.

Lemma 1 *In PGV-17 hash function with the prefix-free padding,* $Pr[Bad_1] = 2^{-n+1} \cdot O(q^2)$ *and* $Pr[Bad_2] = 2^{-n+1} \cdot l^2 \cdot O(q^2)$*, where l is the maximum number of length in a hash query.*

Proof. For i-th query $(-1, h_{i-1}, c_i)$ on $\mathcal{S}$, it is possible that distinguisher $\mathcal{D}$'s query c_i is a valid cipher-text such that $c_i = E_{h_{i-1}}(m_i)$ where h_{i-1} was never queried before. Since q is the maximum times of oracle access and l is the maximum length of a query made by $\mathcal{D}$, thus the probability that the above event occurs is $Pr[Pre_1] = O(\frac{q}{2^n})$ or $Pr[Pre_2] = l \cdot O(\frac{q}{2^n})$. In the worst case, the simulator $\mathcal{S}$ has to track at most $l \times O(q)$ times to check if $\exists IV \xrightarrow{M} h_{i-1} \in \mathcal{R}'_i$. Thus, in case of $\mathcal{O}_1$, the probability of the security events Bad_1 is

$$Pr[Bad_1] = 2 \times Max(Pr[Coll_1], Pr[Pre_1]) = 2 \times Pr[Coll_1] = 2^{-n+1} \cdot O(q^2).$$

In case of $\mathcal{O}_2$, the total number of choices is $l \cdot q$, where l is the maximum number of length in a hash query. Similarly, the probability of the security events Bad_2 is

$$Pr[Bad_2] = 2 \times Max(Pr[Coll_2], Pr[Pre_2]) = 2 \times Pr[Coll_2] = 2^{-n+1} \cdot l^2 \cdot O(q^2).$$

By implementing the advantage of indifferentiability in keyed hash function, similar results can be easily deduced in keyed mode. □

Conventionally, the running time should be the worst case running time of $\mathcal{D}$. According to Lemma 1, we have the following theorem.

Theorem 1 *PGV-17 hash function is* (t_D, t_S, q, ϵ)*-indifferentiable from a random oracle in ideal cipher model with the prefix-free padding, for any distinguisher $\mathcal{D}$ in polynomial time bound t_d, with $t_s = l \cdot O(q)$ and the advantage $\epsilon = 2^{-n+1} \cdot l^2 \cdot O(q^2)$, where l is the maximum length of a query made by $\mathcal{D}$.*

Proof. The results are obvious from the proof of Lemma 1, so we omit the proof here. □

By using the similar method one can find PGV-18, PGV-19, PGV-20 are also indifferentiable from random oracle with the prefix-free padding in the ideal cipher model. It is easy to extend the same results with the NMAC/HMAC and the chop construction. Thus we obtain the following main theorem of this section.

Theorem 2 *The four PGV hash functions are (t_D, t_S, q, ϵ)-indifferentiable from a random oracle in the ideal cipher model with the prefix-free padding, the HMAC/NMAC, and the chop construction, for any distinguisher $\mathcal{D}$ in polynomial time bound t_D, with $t_S = l \cdot O(q)$ and the advantage $\epsilon = 2^{-n+1} \cdot l^2 \cdot O(q^2)$, where l is the maximum length of a query made by $\mathcal{D}$.*

3.3 Indifferentiability of The PBGV Hash Function

Similar to the four PGV hash functions, Chang *et al.*'s indifferentiable attack on the PBGV hash function is only possible with one-block message. In this section, we give an indifferentiability analysis on the PBGV hash function.

Let $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{Y}$ be the PBGV hash function constructed from block-cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$ where $\kappa = n$, $\mathcal{K} \in \{0,1\}^\kappa, \mathcal{M} \in \{0,1\}^*$ and $\mathcal{Y} \in \{0,1\}^{2n}$. *Rand* is a random oracle which has the same domain and range with H. (h_i, g_i) denotes the hash value of the i-th query. The function $Pad(\cdot)$ denotes the prefix-free padding. Let $IV = (h_0, g_0)$ be the initial value. The PBGV hash function takes $l \cdot 2n$-bit message $M = (m_1, m_2, \cdots, m_l)$ (where $m_i = m_{i,1} || m_{i,2}, |m_{i,1}| = |m_{i,2}| = n$) and IV as inputs. For $i = 1$ to l, the PBGV hash function $H : H(M) = (h_l, g_l)$ is iterated as follows.

$$\begin{aligned} h_i &= E_{m_{i,1} \oplus m_{i,2}}(h_{i-1} \oplus g_{i-1}) \oplus m_{i,1} \oplus h_{i-1} \oplus g_{i-1} \\ g_i &= E_{m_{i,1} \oplus h_{i-1}}(m_{i,2} \oplus g_{i-1}) \oplus m_{i,2} \oplus h_{i-1} \oplus g_{i-1} \end{aligned}$$

Chang et al's indifferentiable attack on the PBGV hash function is recalled in Fig 3.3. By the same reason, this attack is also impossible if the message space to messages of at least two blocks.

Distinguisher $\mathcal{D}$ can access to oracles $(\mathcal{O}_1, \mathcal{O}_2)$ where $\mathcal{O}_1 = (H, E, E^{-1})$ and $\mathcal{O}_2 = (Rand, \mathcal{S}, \mathcal{S}^{-1})$.

1. $\mathcal{D}$ selects a message M such that $Pad(M) = m_1 = m_{1,1}||m_{1,2}$ and $|m_1| = 2n$, then he makes the query M to H and receives $H(M) = (h_1, g_1)$.
2. $\mathcal{D}$ makes an inverse query $(-1, m_{1,2} \oplus h_0 \oplus g_0 \oplus g_1, m_{1,1} \oplus h_0)$ to S^{-1} and receives out.
3. if $out = m_{1,2} \oplus g_0$ output 1, otherwise 0.

Fig 3.3 Chang *et al.*'s indifferentiable attack on PBGV.

Now we give a simulation to prove the PBGV hash function with prefix-free padding is also indifferentiable from random oracle. Let distinguisher $\mathcal{D}$ can access to oracles $(\mathcal{O}_1, \mathcal{O}_2)$ where $\mathcal{O}_1 = (H, E, E^{-1})$ and $\mathcal{O}_2 = (Rand, \mathcal{S}, \mathcal{S}^{-1})$. Let $r_i \leftarrow ((h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i))$ be the i-th query-response to the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$ where $m_i \in \{0,1\}^{2n}$. $\mathcal{R}_i = (r_1, \cdots, r_i)$ denotes the query-response set on the oracles $(E, E^{-1}, \mathcal{S}, \mathcal{S}^{-1})$ after i-th query and $\mathcal{R}^*$ is its functional closure. Let $r'_i \leftarrow (IV \xrightarrow{M} (h_i, g_i))$ be the i-th query-response to the oracles $(H, Rand)$ where $M \in \mathcal{M}$. $\mathcal{R}'_i = (r'_1, \cdots, r'_i)$ denotes the query-response set on the oracles $(H, Rand)$ after i-th query.

- ***Rand*-Query.** For i-th $Rand$-query $M_i \in \mathcal{M}$, if M_i is a repetition query, retrieve $r'_j \leftarrow (IV \xrightarrow{M_i} (h_j, g_j))$ where $r_j \in \mathcal{R}'_{i-1}, j \leq i-1$, then return $Rand(M_i) = (h_j, g_j)$. Else randomly select a hash value $(h_i, g_i) \in \mathcal{Y}$ and update $\mathcal{R}'_i = \mathcal{R}'_{i-1} \cup \{IV \xrightarrow{M_i} (h_i, g_i)\}$, then return $Rand(M_i) = (h_i, g_i)$.
- **$(\mathcal{S}, \mathcal{S}^{-1})$-Query.** To answer the distinguisher $\mathcal{D}$'s encryption and decryption queries, the simulator $\mathcal{S}$ responses as follows.
  1. On $\mathcal{S}$ query $(1, x_i, y_i)$,
     (a) If $\exists IV \xrightarrow{M} (h_{i-1}, g_{i-1}) \in \mathcal{R}'_{i-1}$, compute $Pad(M) = m_i = m_{i,1}||m_{i,2}$. Then

i. if $x_i = m_{i,1} \oplus m_{i,2}$ and $y_i = h_{i-1} \oplus g_{i-1}$, run $Rand(M)$ and obtain the response (h_i, g_i), and update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$, then return $h_i \oplus m_{i,1} \oplus y_i$.

ii. if $x_i = m_{i,1} \oplus h_{i-1}$ and $y_i = m_{i,2} \oplus g_{i-1}$, run $Rand(M)$ and obtain the response (h_i, g_i), and update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$, then return $g_i \oplus h_{i-1} \oplus y_i$.

(b) Else randomly select $(h_i, g_i, h_{i-1}, m_{i,1})$, compute $m_{i,2} = x_i \oplus m_{i,1}$ and $g_{i-1} = y_i \oplus h_{i-1}$, and update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$, then return $h_i \oplus m_{i,1} \oplus y_i$.

2. On $\mathcal{S}^{-1}$ query $(-1, x_i, y_i)$,

(a) If $\exists IV \xrightarrow{M} (h_{i-1}, g_{i-1}) \in \mathcal{R}'_{i-1}$, compute $Pad(M) = m_i = m_{i,1}||m_{i,2}$. Then

i. if $x_i = m_{i,1} \oplus m_{i,2}$, run $Rand(M)$ and obtain the response (h_i, g_i). Check if $y_i = h_i \oplus m_{i,1} \oplus h_{i-1} \oplus g_{i-1}$, then update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$ and return $h_{i-1} \oplus g_{i-1}$.

ii. if $x_i = m_{i,1} \oplus h_{i-1}$, run $Rand(M)$ and obtain the response (h_i, g_i). Check if $y_i = g_i \oplus m_{i,2} \oplus h_{i-1} \oplus g_{i-1}$, then update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$ and return $m_{i,2} \oplus g_{i-1}$.

(b) Else randomly select $(h_{i-1}, g_{i-1}, m_{i,1}, g_i)$, compute $h_i = y_i \oplus m_{i,1} \oplus h_{i-1} \oplus g_{i-1}$ and $m_{i,2} = x_i \oplus m_{i,1}$, update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$. Then return $h_{i-1} \oplus g_{i-1}$.

Before stating the main result of the PBGV hash function, a simple lemma is proved.

Lemma 2 *In PBGV double block length hash functions with the prefix-free padding,* $Pr[Bad_1] = 2^{-n-3} \cdot O(q^2)$ *and* $Pr[Bad_2] = 2^{-n-3} \cdot l^2 \cdot O(q^2)$*, where* l *is the maximum number of length in a hash query.*

Proof. In [14], it is proved that the upper bound of the collision attack on the PBGV hash function is $4 \times O(2^{n/2})$.

$Pr[Bad_1] = 2 \times Max(Pr[Coll_1], Pr[Pre_1]) = 2 \times Pr[Coll_1] = 2^{-n-3} \cdot O(q^2)$.

In case of $\mathcal{O}_2$, the total number of choices is $l \cdot q$, where l is the maximum number of length in a hash query. Similarly, the probability of the security events Bad_2 is

$$Pr[Bad_2] = 2 \times Max(Pr[Coll_2], Pr[Pre_2]) = 2 \times Pr[Coll_2] = 2^{-n-3} \cdot l^2 \cdot O(q^2).$$

By implementing the advantage of indifferentiability in keyed hash function, similar results can be easily deduced in keyed mode. □

Similar to the four PGV hash functions, one can easily obtain the following result from the above analysis.

Theorem 3 *The PBGV hash function is (t_D, t_S, q, ϵ)-indifferentiable from a random oracle in the ideal cipher model with the prefix-free padding, the NMAC/HMAC, and the chop construction, for any distinguisher $\mathcal{D}$ in polynomial time bound t_D, with $t_S = 2l \cdot O(q)$ and the advantage $\epsilon = 2^{-n-3} \cdot l^2 \cdot O(q^2)$, where l is the maximum length of a query made by $\mathcal{D}$.*

Although Knudsen *et al.*[14] proved that the PBGV hash function and some of fast DBL hash functions are not optimally secure against the preimage and the collision attacks, our indifferentiable result of the PBGV hash function is not conflict with theirs. By assuming the block cipher used in the hash function is *ideal*, the advantage of indifferentiability will be reduced to a negligible value. Actually, the advantage of indifferentiability is based on computational complexity, not on the measurement of the unconditional security at all. Similar results on others double block length hash functions, i.e., MDC2, QG-I, and LOKI-DBH etc, can be extended from the proof of the PBGV hash function.

4 Conclusion

Since hash functions play a fundamental primitive in nearly all of the cryptosystems, investigating how to design a better hash function is important. In this paper, a synthetic indifferentiability analysis of some block-cipher-based hash functions is described. The results show the fact that all of the 20 collision resistant PGV hash functions and the PBGV hash function are indifferentiable from random oracle with the prefix-free padding, the HMAC/NMAC and the chop construction. The analysis can be extended to MDC2, QG-I, and LOKI-DBH etc. As the notion of indifferentiability is a critical methodology to find the gap between hash function and random oracle in a white-box investigation, there are still many hash functions and MD variants are open in the view of indifferentiability security analysis.

Acknowledgments. We would like to thank the anonymous reviewers for helpful comments that improved the presentation of this paper.

References

[1] M. Bellare and P. Rogaway. Random oracle are practical: a paradigm for designing efficient protocols. In *ACM CCS'93*, ACM, pp. 62-73. 1993.

[2] J. Black, P. Rogaway and T. Shrimpton. Black-Box Analysis of the Black-Cipher-Based Hash-Function Constructions from PGV. *Advances in Cryptology - CRYPTO'02*. LNCS 2442, pp. 320-335. 2002.

[3] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. of Computing*, Vol. 32, No. 3, pp. 586-615. 2003.

[4] B.O. Brachtl, D. Coppersmith, M.M. Hyden, S.M. Matyas, C.H. Meyer, J. Oseas, S. Pilpel and M. Schilling. *Data Authentication Using Modification Detection Codes Based on a Publick One Way Encryption Function*. U.S. Patent Number 4,908,861, March 13, 1990.

[5] D. Brown. Generic Groups, Collision Resistance, and ECDSA. In *http://eprint.iacr.org/2002/026*. 2002.

[6] R. Canetti, O. Goldreich and S. Halevi. The randoom oracle methodology, revisited. In *Proceedings of 30th ACM Symposium on the Theory of Computing*, ACM Press, pp. 209-218. 1998.

[7] D. H. Chang, S. J. Lee, M. Nandi and M. Yung. Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. X. Lai and K. Chen(Eds): *ASIACRYPT 2006*, LNCS 4284, pp. 283-298. 2006.

[8] J. S. Coron, Y. Dodis, C. Malinaud and P. Puniya. Merkle-Damgard Revisited: How to Construct a Hash Function. *Advances in Cryptology - CRYPTO'05*, LNCS 3621, pp. 21-39. 2005.

[9] I. Damgard. A Design Principle for Hash Functions, *Advances in Cryptology, Cyrpto'89*, LNCS 435, pp. 416-427. 1989.

[10] A. Dent. Adapting the weakness of the random oracle to the generic model. In *ASIACRYPT 2002*, LNCS 2501, pp. 101-109. 2002.

[11] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *CRYPTO'99*, LNCS 1666, pp. 537-554. 1999.

[12] S. Goldwasser and Y. Tauman. On the (In)security of the Fiat-Shamir Paradigm. In *FOCS 2003*, IEEE Computer Society, pp. 102-122. 2003.

[13] S. Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In *FSE 2006*, LNCS 4047, pp. 210-225. 2006.

[14] L.R. Knudsen, X. Lai and B. Preneel. Attacks on Fast Double Block Length Hash Functions. *Journal of Cryptology*(1998) 11: 59-72.

[15] X. Lai and J. L. Massey. Hash Functions Based on Block Ciphers. In *Advances in Cryptology-Eurocrypt'92*, LNCS 658, pp. 55-70. 1993.

[16] S. Lucks. A Failure-Friendly Design Principle for Hash Functions. In *ASIACRYPT 2005*, LNCS 3788, pp. 474-494. 2005.

[17] U. Maurer, R. Renner, and C. Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. *Theory of Cryptography - TCC 2004*, LNCS 2951, pp. 21-39. 2004.

[18] R.C. Merkle. One way hash functions and DES, *Advances in Cryptology, Crypto'89*, LNCS 435, pp. 428-446. 1989.

[19] J.B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In *CRYPTO'98*, LNCS 2442, pp. 111-126. 2002.

[20] P. Paillier and D. Vergnaud. Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log. In *ASIACRYPT 2005*, LNCS 3788, pp. 1-20. 2005.

[21] B. Preneel, A, Bosselaers, R. Govaerts and J. Vandewalle. Collision-free Hash-functions Based on Blockcipher Algorithms. In *Proceeding of 1989 International Carnahan Conference on Security Technology*, pp. 203-210. 1989.

[22] B. Preneel, R. Govaerts and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *Advances in Cryptology - CRYPTO'93*, LNCS 773, pp. 368-378. 1994.

[23] P. Rogaway and T. Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance and Collision Resistance. In *FSE 2004*, LNCS 3017, pp. 371-388. 2004.

[24] C. Shannon. Communication theory of secrecy systems. Bell Systems Techincal Journal, 28(4): pages 656-715. 1949.

[25] X. Wang, Y. Yin and H. Yu. Finding Collision in the Full SHA-1. In *CRYPTO'05*, LNCS 3621, pp. 17-36. 2005.

[26] X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In *EUROCRYPT'05*, LNCS 3494, pp. 19-35. 2005.