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

Nowadays, investigating what construction is better to be a cryptographic hash function is red hot. In [13], Maurer et al. first introduced the notion of indifferentiability as a generalization of the concept of the indistinguishability of two cryptosystems. At *ASIACRYPT'06*, Chang et al. [6] analyzed the indifferentiability security of some popular block-cipher-based hash functions, such as PGV constructions and MDC-2. In this paper, we investigate Chang et al.'s analysis of PGV constructions and the PBGV double block length constructions. In particular, we point out a more precise adversarial advantage of indifferentiability, by considering the two situations that whether the hash function is either keyed or not. Furthermore, Chang et al.[6] designed attacks on 4 PGV hash functions and PBGV hash function to prove they are differentiable from random oracle with prefix-free padding. We find a limitation in their differentiable attacks and construct our simulations to obtain the controversy results that those schemes are indifferentiable from random oracle with prefix-free padding and some other popular constructions.

1 Introduction

Block-Cipher-Based Hash Function Hash functions are a cryptographic primitive in the design of schemes to provide a unique "fingerprint" on a certain information. If hash functions receive some additional properties, they

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

can make schemes more secure and efficient. In practice, *one-wayness* property and *collision-resistance* property are two fundamental conditions for a hash function can be implemented. The cryptoanalysis of a hash function construction is used to basing on the statistical analysis to check whether it preserves the secure conditions. Many articles have discussed those general aspects of how to construct hash functions. Instructive examples can be found in [8, 14].

Cryptoanalysis on hash functions has been focussed on the question: what conditions should be imposed on f to guarantee that h satisfies certain properties? It is obvious that weakness of f will affect the security of h, but the converse does not hold in general. An accomplished assessing standard is checking whether a CRHF or OWHF can be derived from an ideal fixed size compression function or an ideal block cipher.

In practice, most of hash functions are either explicitly based on block-cipher for [24, 12] or implicitly as in SHA-1[16]. Preneel et al.[19] proposed 64 kinds of constructions to build up a hash function $H : \{0,1\}^* \to \{0,1\}^n$ from a compression function $f : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ by using a block cipher $E : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^l$. They regards 12 out of 64 schemes (We denote these 12 schemes by PGV-Group 1) as secure, besides the remaining 52 schemes were shown to be not collision resistant or preimage resistant. Recently, Black, Rogaway and Shrimption [3] proved that in a black box model, the 12 schemes are really secure. Furthermore, they showed that an additional 8 of the 52 remaining schemes are collision resistant as the first 12 schemes (We will call these 8 schemes as PGV-Group 2), which are classified as *backward-attackable* (potential but not serious) in [19].

Random Oracle Model Random oracle methodology was proposed by Bellare and Rogaway in [1], which is quickly wide-used because the schemes design under such model would be more efficient and practical while compare to the standard model ones. In most applications, random oracle is an oracle that anybody can query but no one has control over. 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[10]. 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[4].

Since random oracle performs quite like hash function, people is suggested to replace random oracle in their schemes with a "secure" dedicated hash function, such as SHA-1, SHA-256, etc. In variant applications, the requirements impose on the hash function are also different. E.g., in authority implementations, hash functions are always used to store manipulation detective code (MDC) corresponding to the password, instead of the password itself, which implies that *one-wayness* is sufficient. But in digital signature schemes, hash functions must be *collision-resistance*, since two distinct messages have the same hash value allows forgery and repudiation. Formal construction of hash functions totally focus on the statistical cryptanalysis to make the proof of the security. One has to be careful with the selection of the hash function, because a specific vulnerability will be found in between the digital signature scheme and the hash function[5, 17]. Since we can prove such scheme is secure in random oracle model, this back to the efforts that how can we design an *ideal* hash function *same as* random oracle.

Indifferentiability Methodology Research on how to instantiate the random oracle with a certain hash function has been a hot argumentation in recent years. Many valuable references on this problem could not be indicated at a specific location: [9, 11, 15]. The problem has been focussed on the question: what conditions should be imposed on the round function $\mathcal{F}$ to make sure that the transform $\mathcal{C}^{\mathcal{F}}$ 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 the arbitrary size of input. It is clear that the weakness of $\mathcal{F}$ will generally result in weakness of $\mathcal{C}^{\mathcal{F}}$, but the converse does not hold in general. The main problem is to derive such that sufficient conditions. In [13], Maurer et al. first provide a term "indifferentiability" and a formal model to "distinguish" whether a given construction has any different from a heuristic random oracle. Informally, indifferentiability methodology is a *white-box* analysis that be restrictive to investigate all of the internal interfaces in the construction, while indistinguishability methodology is just a *black-box* analysis which ignores the internal.

Recently, Coron et al.[7] first implemented the notion of indifferentiability for analysis of some classical hash constructions. They proved that plain Merkle-Damgard hash function is differentiable with random oracle, and show MD hash functions will be indifferentiable under the prefix-free, HMAC/NMAC and Chop constructions. Following Coron et al.'s initial work, Chang et al.[6] continued this suggestion and analyzed the indifferentiability in some block-cipher-based hash functions with prefix-free padding, especially in PGV hash functions. They claimed that there are 16 out of

20 collision-resistant PGV hash functions are indifferentiable from random oracle in the ideal cipher model, while the remains 4 schemes are not. And they also gave an differentiable attack on the PBGV double block length hash function[18].

Some Missing Results The authors of [6] only focused on collision-free event, not all of the security events on hash function, e.g., preimage attack, second preimage attack, etc. Sepcifically, they only analyzed the situation in unkeyed mode of hash functions. Since keyed hash functions are receiving more and more attention, after the genius attacks were found by Wang et al. in dedicated-key hash functions, such as MD4, MD5 and SHA-1[22, 23]. The indifferentiability security analysis of keyed hash function becomes both practically and theoretically significant. Prior to the current work, we are unaware of any indifferentiability advantage analysis for the keyed hash functions based on any block-cipher. We begin by giving a more suitable definition of adversary in block-cipher-based hash functions and then proposing the advantage of differentiable attackers in either keyed or unkeyed mode to strengthen the result. Moreover, Chang et al.[6] also showed a differentiable attack on 4 out of 20 PGV hash functions and PBGV hash function. They said the attacks is not only valid in one-block padded message, but also similar to multi-block message. But we find a flaw in their attacks, which limits their attacks only works in the one-block mode. As the main contribution of this paper, we give our simulations to prove that those schemes are indifferentiable from random oracle with prefix-free padding and some other popular constructions.

Organization The remainder of this paper is organized as follows. In Section 2, we provide some necessary definitions on block-cipher-based hash functions and indifferentiability methodology for our security analysis. Then in Section 3, first we propose an more exact analysis of adversary's advantage in indifferentiability. After that, we present our security analysis on the four PGV hash functions and PBGV double block length hash function. Section 4 give a conclusion.

2 Preliminary

Here we provide those main notions and definitions that will be used throughout the paper. 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

Ideal cipher model is the formal model for the security analysis of block-cipher-based hash functions, which is dating back to Shannon [21] and widely used, e.g., in [12, 19]. Let $\texttt{Bloc}(K, X) = E : K \times X \rightarrow X$ be a family of trapdoor permutation, where $E(k_i, \cdot), k_i \in K$ ($E_{k_i}(\cdot)$ for short) denotes an instance of the family. An adversary can query two oracles: E and its inversion E^{-1}. Thus, we define the i-th query-response q_i is a four-tuple

$$q_i = (\sigma_i, k_i, x_i, y_i).$$

If $\sigma_i = 1$ then adversary inputs (k_i, x_i) and gets response $y_i = E_{k_i}(x_i)$, otherwise inputs (k_i, y_i) and gets answer $x_i = E^{-1}_{k_i}(y_i)$. In generally, $E_k(\cdot)$ is a trapdoor permutation that $Pr[E_k(x) = y] = \frac{1}{/}|X|$ and for different k, e.g., $E_{k_1}(\cdot), \cdots, E_{k_i}(\cdot)$ have independently uniform distributions. See [3] for more details about ideal cipher model.

2.2 Definitions for Indifferentiability Security Analysis

We now recall the definitions for indifferentiability security analysis [13].

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 showed in [13] 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 iterative 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 keep 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 the rest of the paper, the Turing Machine $\mathcal{C}$ will denote the construction of an iterative hash function. The ideal primitive $\mathcal{F}$ will represent the underlying iterative function. E denotes the block cipher used in the iterative function and E^{-1} denotes the corresponding inverse operation. Since we focus on block-cipher-based hash functions in case of the ideal cipher model, $\mathcal{S}$ has to simulate both E and E^{-1}.

Therefore, every distinguisher $\mathcal{D}$ obtain 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 and the block-cipher E, E^{-1} is implemented by a simulator $\mathcal{S}$ with oracle access to H. Those two ways to build up a hash function should be indistinguishable.

2.3 Adversary in Block-cipher-based Hash Functions

For indifferentiability security analysis of block-cipher-based hash functions, we need to formally define the adversary in the indifferentiability model. Chang et al. [6] proposed a definition of the adversary in the random oracle model (attack hash functions based on one-way compression function). We propose our modifications to make it more suitable for block-cipher-based hash functions.

Let $\mathcal{D}$ be a distinguisher in the indifferentiable attack. $\mathcal{D}$ can access two oracles. One is $\mathcal{O}_1 = (H, E, E^{-1})$, the other is $\mathcal{O}_2 = (Rand, S, S^{-1})$. H denotes iterative hash function based on block-cipher E. Let $r_i \leftarrow (IV \xrightarrow{M} h_i)$ be the i-th query to $\mathcal{O}_1$, where $M \in \mathcal{M}$. Let $r_j \leftarrow (h_i \xrightarrow{m} h_j)$ be the j-th to $\mathcal{O}_2$. Let $\mathcal{R}_i = (r_1, \cdots, r_i)$ be the query-response set after i^{th} query. $Pad(\cdot)$ denotes scheme's padding rule. Let $M = m_0||m_1||\cdots|||m_i$, where $||$ denotes the concatenation operation. In fact, $IV \xrightarrow{M} h_i$ can be represented by $h_0 \xrightarrow{m_0} h_1 \cdots \xrightarrow{m_i} h_i$

In [6], Chang et al. only defined one kind of distinguisher's query. It is not sufficient for block-cipher-based hash functions since the query for $H, Rand$ is quite different from the query for E, E^{-1}, S, S^{-1}. The $H, Rand$ query inputs an arbitrary length message and the response is a fixed length hash value, while the E, E^{-1}, S, S^{-1} query is a plain-text or cipher-text in fixed block length and the output is the corresponding cipher-text or plain text, respectively. Here we give a complete definition for the both cases.

- **Query on** (E, E^{-1}, S, S^{-1}):

  1. For i-th query on (E, S), distinguisher $\mathcal{D}$ inputs $Q_i = (1, h_{i-1}, m_i)$ and the response is $h_i = E_{h_{i-1}}(m_i)$ or $S(h_{i-1}, m_i)$. Here $m_i \in \mathcal{M}$ is fixed one block length, where $\mathcal{M}$ denotes message space.
  2. For j-th query on E^{-1}, S^{-1}, adversary's query is $Q_i = (-1, h_{i-1}, c_i)$ and the response is $m_i = E^{-1}_{h_{i-1}}(c_i)$ or $S^{-1}(h_{i-1}, c_i)$. Let $R_i = R_{i-1} \cup (h_{i-1} \xrightarrow{m_i} h_i)$. We denote $\mathcal{R} = (R_1, R_2, \cdots, R_q)$ be the complete view after the maximum q queries. According to the transitive and substitute properties of Q_i, the functional closure set $\mathcal{R}^* = (R^*_1, \cdots, R^*_q)$ will be the complete view of distinguisher $\mathcal{D}$.

- **Query on** $(H, Rand)$: For i-th query on $H, Rand$, distinguisher $\mathcal{D}$ can select an arbitrary length message $Q'_i = M_i \in \mathcal{M}$ as input. Thus the query on keyed hash functions will be $Q'_i = (k_i, M_i)$. The response of $H, Rand$ is $h_i = H(M_i)$ or $S(M_i)h_i \in Y$, where Y is the range of the oracles. Let Q'_i be the i-th query. For brevity, $R'_i = R'_{i-1} \cup (IV \xrightarrow{M_i} h_i)$ denotes the functional closure set after i-th query. Let $\mathcal{R}' = (R'_1, R'_2, \cdots, R'_q)$ be the complete view after the maximum q queries.

In a simulation game, we will ignore all the repetition query i.e. $R_i = R_j$ or $R'_i = R'_j$ for some $j < i$. For simplicity, we can assume there is no such trivial query since it does not help distinguisher as the view of indifferentiability.

3 Security analysis on some popular hash functions

In this section, we point out our indifferentiability advantage analysis on block-cipher-based hash functions. And then we give our indifferentiability security analysis on 4 out of 20 PGV hash functions and PBGV double block length hash function.

3.1 Advantage of Indifferentiability

The original analysis given by Chang et al. [6] just covered the situation of *collision event*. By considering the high-level classification that whether the hash function is either keyed or not, we take all the differentiable events in consideration to achieve a more precise adversarial advantage of indifferentiability.

First, we describe the situation in unkeyed hash functions. To give a exact probability analysis, we must carefully consider all the events that will affect the advantage of distinguisher D. Let **Bad** be the indifferentiable event of the distinguisher D for iterative hash functions. We assume (H, E, E^{-1}) and $(Rand, S, S^{-1})$ are identically distributed conditioned on the past view of the distinguisher and **Bad** does not occur. Since iterative hash function can easily resist extension attack by length padding technique, we will ignore this event by implicitly using the padding in our discussion. If a hash construction already used a padding rule, then we will combine length padding technique with the given rule without special description. For brevity, we denote the event $\mathcal{D}^{H,E,E^{-1}} = 1$ by D_1 and the event $\mathcal{D}^{Rand,S,S^{-1}} = 1$ by D_2. Let $\mathbf{Bad}_i, i \in \{1, 2\}$ denotes the indifferentiable event for $\mathcal{O}_1 = (H, E, E^{-1})$ and $\mathcal{O}_2 = (Rand, S, S^{-1})$, respectively. The function $Max(\cdot, \cdot)$ returns the biggest value of inputs. If $\mathcal{D}$ is a distinguisher then we write $Adv(\mathcal{D})$ as a measure of the maximal differentiable advantage overall distinguishers $\mathcal{D}$. The advantage of the indifferentiability of (H, E, E^{-1}) with $(Rand, S, S^{-1})$ is as follows

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

Then we analyze the differentiable event **Bad** in hash functions. For unkeyed hash function, the security properties include collision resistance, secondary preimage and preimage. Because collision resistance(CR for short) implies second-preimage resistance(Sec for short), while separates from preimage resistance(Pre for short), then the differentiable event

$$\mathbf{Bad} = \{CR, Pre\}.$$

For keyed hash functions, there are more security properties need to be considered. Besides the standard three of CR, Pre and Sec, there are four *always* and *everywhere* variants (aSec, eSec, aPre and ePre) for keyed hash functions. According to the conclusion of implications and separations for keyed hash functions, the differentiable event of keyed hash functions will be

$$\mathbf{Bad^{key}} = \{CR, eSec, aPre, ePre\}.$$

For brevity, We ignore the description of those security definitions and the proof of the implications and separations here, see [14, 20] for more details.

3.2 Different Result on Chang et al.'s Attack on Some Block-Cipher-Based Hash Functions

In [6], Chang et al. showed a differentiable attack on four PGV hash functions and PBGV hash function. They said such attack is not only works with one-block padded message, but also more than one block. We find a limitation in their attack, which exposed their attack only works in the one-block mode. Then we construct our simulations to prove that those schemes are indifferentiable from random oracle with prefix-free padding and some other popular constructions.

3.3 The Four PGV Hash Functions

To prove their result, Chang et al. gave an indifferentiable attack on the four PGV hash functions. The four schemes are $E_{h_{i-1}}(m_i) \oplus m_i$ $(PGV-17)$, $E_{h_{i-1}}(m_i \oplus h_{i-1}) \oplus m_i \oplus h_{i-1}(PGV-18)$, $E_{h_{i-1}}(m_i) \oplus m_i \oplus h_{i-1}(PGV-19)$, $E_{h_{i-1}}(m_i \oplus h_{i-1}) \oplus m_i(PGV-20)$.They claimed that the attack is based on one-block padded message, which is easily extended to more than one block.

We give our analysis that their attack is not feasible in the multi-block mode. And then we prove the four PGV hash functions are indifferentiable from random oracle with prefix-free padding. First we recall Chang et al.'s attack on $PGV-17$ 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. Select a message M such that $g(M) = m$ and $|m| = n$, then make the query M to H and receive z.
2. make an inverse query $(-1, h_{i-1}, z \oplus m)$ to S^{-1} and receive m^*, where $h_{i-1} = h_0 = IV$
3. if $m = m^*$ output 1, otherwise output 0.

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

Since any simulator $\mathcal{S}$ can return $m^* = m$ only with probability 2^{-n}, it is obviously differentiable from random oracle. We see the attack can only works when $h_{i-1} = h_0 = IV$, consequently the result is correct only in one block mode. The situation will be quite different while works with padding rules, such as prefix-free or NMAC, etc. In prefix-free mode, $z \oplus m$ will equal to $E_{h_{i-1}}(m_i)$ and there are no messages shorter than two block in $(H, Rand)$-query. Because distinguisher $\mathcal{D}$ only knows $H(M) = z$ (interior value h_{i-1} is unknown), $\mathcal{D}$ cannot make inverse query $(-1, h_{i-1}, z \oplus m)$ if $h_{i-1} \notin \mathcal{R}^*$. So the attack cannot work in multi-block message mode, which means Chang et al.'s differentiable attack is only feasible in one-block message mode. It is easily to verify the differentiable attack on PBGV hash function fails in multi-block message similarly. If $\mathcal{D}$ has asked the internal value h_{i-1} before, then $\mathcal{S}$ can track it from the relation closure $\mathcal{R}$.

Now we will give our simulation to show the above 4 PGV hash functions are indifferentiable from random oracle in the prefix-free mode. Let $Pad(\cdot)$ denote the padding algorithm. We can assume distinguisher $\mathcal{D}$ never make a repetition query since it does not help anything.

- *Rand*-**Query.** For *Rand*-query Q'_i, If Q'_i is a repetition query, then retrieves h_j where $Q'_i \in R'_j$. Else *Rand* returns $h_i = Rand(M_i)$ and updates $R'_i = R'_{i-1} \cup \{IV \xrightarrow{M_i} h_i\}$.

- $(\mathcal{S}, \mathcal{S}^{-1})$**-Query.** Like the previous simulator, our simulator $\mathcal{S}$ also keeps the relations $\mathcal{R} = (R_1, \cdots, R_{i-1})$. Initially, $R_0 = \emptyset$. To answer distinguisher $\mathcal{D}$'s encrypt or decrypt query, the response of $\mathcal{S}$ is as follow:

  1. On $\mathcal{S}$ query $(1, h_{i-1}, m_i)$,
     (a) If $\exists IV \xrightarrow{M} h_{i-1} \in \mathcal{R}_{i-1}$ and $Pad(M) = m_i$, then 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, $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{h_{i-1} \xrightarrow{m_i} h_i\}$, then return $h_i \oplus m_i$.
  2. On $\mathcal{S}^{-1}$ query $(-1, h_{i-1}, c_i)$,
     (a) If $\exists IV \xrightarrow{M} h_{i-1} \in \mathcal{R}_{i-1}$, then run $Rand(M)$ and obtain the response h_i. Check if $c_i = h_i \oplus Pad(M)$, then 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.

We notice that on $\mathcal{S}^{-1}$ query, there is a probability that distinguisher $\mathcal{D}$'s query c_i is a valid cipher-text on the key h_{i-1} while h_{i-1} has been never queried. Because q is the maximum times of oracle access and l is the maximum length of a query made by $\mathcal{D}$, the probability that the above event occurs is is equal to $Pr[Pre_2] = l \cdot O(\frac{q}{2^n})$. In the worst case, simulator $\mathcal{S}$ has to track at most $l \times O(q)$ to check if $\exists IV \xrightarrow{M} h_{i-1}$. By convention, the running time is actual worst case running time of $\mathcal{D}$.

Lemma 1 *In PGV-17 hash functions with 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. Assume there are no repetition query. Thus, in case of $\mathcal{O}_1$, there are q queries and the probability is

$$Pr[Bad_1] = 2 \times max\{Pr[CR_1], Pr[Pre_1]\} = 2 \times Pr[CR_1] = 2^{-n+1} \cdot O(q^2).$$

Prefix-free MD(IV, M)	NMAC Construction (IV, M)
$M = m_1 \| \cdots \| m_i, h_0 = IV$	$M = m_1 \| \cdots \| m_i, h_0 = IV$
For $i = 1$ to i do $h_i = F(PFPad(m_i), h_{i-1})$	For $i = 1$ to i do $h_i = F(m_i, h_{i-1})$
Return h_i	Return $g(h_i)$
$Pad(\cdot)$ is a Prefix-free padding function	$g(\cdot)$ is a random permutation
HMAC Construction (IV, M)	Chop Construction (IV, M)
$M = m_1 \| \cdots \| m_i, h_0 = IV$	$M = m_1 \| \cdots \| m_i, h_0 = IV$
For $i = 1$ to i do $h_i = F(m_i, h_{i-1})$	For $i = 1$ to i do $h_i = F(m_i, h_{i-1})$
Return $h_{i+1} = F(h_i, IV)$	Return $Chop(h_i)$

Fig 3.2 **Definitions of the four MD variants proposed in [7].** We notice that string IV is fixed initialization vector. M is arbitrary message in the space $\mathcal{M}$. $PFPad(m_j)$ returns $1||m_j$ if m_j is the last block, else returns $0||m_j$. $g(x), x \in \{0,1\}^n$ is a random permutation in $\{0,1\}^n$.

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

$$Pr[Bad_2] = 2 \times max\{Pr[CR_2], Pr[Pre_2]\} = 2 \times Pr[CR_2] = 2^{-n+1} \cdot l^2 \cdot O(q^2).$$

According to previous lemma, we have the following theorem of our result.

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

Proof. The results are obvious from the above analysis, so we skip the proof here.$\square$

By using the similar method we can find $PGV-18$, $PGV-19$, $PGV-20$ are also indifferentiable from random oracle in prefix-free mode. It is easily to extend the same results in NMAC, HMAC and Chop constructions.

Theorem 2 *Prefix-free **Group-2** hash functions in ideal cipher model is (t_D, t_S, q, ϵ)-indifferentiable form a random oracle. For any distinguisher $\mathcal{D}$ in polynomial time bound t_D and $t_S = l \cdot O(q)$, the advantage will be $\epsilon = 2^{-n+1} \cdot l^2 \cdot O(q^2)$, where l is the maximum length of a query made by $\mathcal{D}$.*

Theorem 3 *HMAC/NMAC **Group-2** hash functions in ideal cipher model is (t_D, t_S, q, ϵ)-indifferentiable form a random oracle. For any distinguisher $\mathcal{D}$ in polynomial time bound t_D and $t_S = l \cdot O(q)$, the advantage will be $\epsilon = 2^{-n+1} \cdot l^2 \cdot O(q^2)$, where l is the maximum length of a query made by $\mathcal{D}$.*

Theorem 4 *Chop **Group-2** hash functions in ideal cipher model is (t_D, t_S, q, ϵ)-indifferentiable form a random oracle. For any distinguisher $\mathcal{D}$ in polynomial time bound t_D and $t_S = l \cdot O(q)$, the advantage will be $\epsilon = 2^{-n+1} \cdot l^2 \cdot O(q^2)$, where l is the maximum length of a query made by $\mathcal{D}$.*

3.4 The PBGV Hash Function

Similar to the (Group-2) PGV hash fucntions, Chang et al.'s differentiable attack on the PBGV hash fucntion is also infeasible in multi-block message mode. Here We give our indifferentiability analysis on the PBGV hash function. The PBGV scheme is a double block length hash function proposed in [18]. Let $IV = h_0||g_0$ be initialization vetors. E denotes a block cipher with $\{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$. The PBGV hash function takes $l \cdot 2k$-bit message $M = (m_1, m_2, \cdots, m_l)$ (where $m_i = m_{i,1}||m_{i,2}, |m_{i,1}| = |m_{i,2}| = k$) and IV as inputs. For $i = 1$ to l, the PBGV hash fuction $H : H(M) = h_l||g_l$ is defined as follows.

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

Now we give our 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 $\mathcal{O}_2 = (Rand, \mathcal{S}, \mathcal{S}^{-1})$

- *Rand*-**Query.** For *Rand*-query Q'_i, If Q'_i is a repetition query, then *Rand* retrieves h_j where $Q'_i \in R'_j$. Else *Rand* returns $(h_i, g_i) \leftarrow Rand(M_i)$ and updates $R'_i = R'_{i-1} \cup \{IV \xrightarrow{M_i} (h_i, g_i)\}$.
- $(\mathcal{S}, \mathcal{S}^{-1})$**-Query.** Like the previous simulator, our simulator $\mathcal{S}$ also keeps the relations $\mathcal{R} = (R_1, \cdots, R_{i-1})$. Initially, $R_0 = \emptyset$. To answer distinguisher D's query, the response of $\mathcal{S}$ is as follow:

1. On S query $(1, x_i, y_i)$,
   (a) If $\exists IV \xrightarrow{M} (h_{i-1}, g_{i-1}) \in \mathcal{R}_{i-1}$, first compute $Pad(M) = m_i = m_{i,1}||m_{i,2}$, then:
      i. If $x_i = m_{i,1} \oplus m_{i,2}$ and $y_i = h_{i-1} \oplus g_{i-1}$, then run $Rand(M)$ and obtain the response (h_i, g_i), update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$ and return $h_i \oplus m_{i,1} \oplus y_i$.
      ii. Else If $x_i = m_{i,1} \oplus h_{i-1}$ and $y_i = m_{i,2} \oplus g_{i-1}$, then run $Rand(M)$ and obtain the response (h_i, g_i), update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$ and return $g_i \oplus h_{i-1} \oplus y_i$.
   (b) Else select $(h_i, g_i, h_{i-1}, m_{i,1})$ uniformly and randomly, compute $m_{i,2} = x_i \oplus m_{i,1}$ and $g_{i-1} = y_i \oplus h_{i-1}$, update $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \{(h_{i-1}, g_{i-1}) \xrightarrow{m_i} (h_i, g_i)\}$. Return $h_i \oplus m_{i,1} \oplus y_i$.
2. On $\mathcal{S}^{-1}$ query $(-1, x_i, y_i)$,
   (a) If $\exists IV \xrightarrow{M} (h_{i-1}, g_{i-1}) \in \mathcal{R}_{i-1}$, first compute $Pad(M) = m_i = m_{i,1}||m_{i,2}$, then:
      i. If $x_i = m_{i,1} \oplus m_{i,2}$, then 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 $x_i = m_{i,1} \oplus h_{i-1}$, then 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 choose $(h_{i-1}, g_{i-1}, m_{i,1}, g_i)$ uniformly and randomly, 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 $h_{i-1} \oplus g_{i-1}$.

Similarly, we can obtain the following results by implementing the simulation.

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

Theorem 5 *Prefix-free* ***PBGV*** *double block length hash functions in ideal cipher model is (t_D, t_S, q, ϵ)-indifferentiable form a random oracle. For any* *distinguisher $\mathcal{D}$ in polynomial time bound t_D and $t_S = 2l \cdot O(q)$, the advantage will be $\epsilon = 2^{-2n+1} \cdot l^2 \cdot O(q^2)$, where l is the maximum length of a query made by $\mathcal{D}$.*

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, first we point out a more precise advantage of indifferentiability, by considering the high level that whether the hash function is keyed or not. Then we show a flaw in Chang et al.'s differentiable attack on 4 out of 20 PGV hash functions and PBGV hash function, and give our simulations to prove those schemes with prefix-free are actually indifferentiable from random oracle. The result shows all of 20 PGV hash functions that Black et al. proved secure in [3] are indifferentiable from random oracle if we choose Prefix-free, HMAC/NMAC and Chop constructions, and the same goes to the PBGV double block length hash function. 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 kinds of hash functions and padding rules are open in the view of indifferentiability security analysis and our synthetic advantage boundary of distinguisher.

References

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

[2] M. Bellare and T. Ristenpart. Multi-Property-Preserving Hash Domain Extension and the EMD Transform. X. Lai and K. Chen eds. *Advances in Cryptology - ASIACRYPT 2006*, LNCS 4284, pp. 299-314. 2006.

[3] 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.

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

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

[6] 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.

[7] 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.

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

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

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

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

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

[13] 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.

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

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

[16] National Institute of Standards and Technology. FIPS PUB 180-1: Secure Hash Standard. (1995) Supersedes FIPS FUB 180 1993 May 11.

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

[18] 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.

[19] 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.

[20] 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.

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

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

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

[24] R.S. Winternitz. A secure one-way hash function built from DES. *Proc. IEEE Symposium on Information Security and Privacy 1984*, 1984, pp.88-90.