On the Security of Hash Functions Employing Blockcipher Postprocessing

Donghoon Chang[1], Mridul Nandi[2], and Moti Yung[3]

[1] National Institute of Standards and Technology, USA
pointchang@gmail.com
[2] C R Rao AIMSCS Institute, Hyderabad, India
mridul.nandi@gmail.com
[3] Google Inc. and Department of Computer Science, Columbia University, New York, USA
my123@columbia.edu

Abstract. Analyzing desired generic properties of hash functions is an important current area in cryptography. For example, in Eurocrypt 2009, Dodis, Ristenpart and Shrimpton [7] introduced the elegant notion of "Preimage Awareness" (PrA) of a hash function H^P, and they showed that a PrA hash function followed by an output transformation modeled to be a FIL (fixed input length) random oracle is PRO (pseudorandom oracle) i.e. indifferentiable from a VIL (variable input length) random oracle. We observe that for recent practices in designing hash function (e.g. SHA-3 candidates) most output transformations are based on permutation(s) or blockcipher(s), which are not PRO. Thus, a natural question is how the notion of PrA can be employed directly with these types of more prevalent output transformations? We consider the Davies-Meyer's type output transformation $OT(x) := E(x) \oplus x$ where E is an ideal permutation. We prove that $OT(H^P(\cdot))$ *is PRO if* H^P *is PrA, preimage resistant and computable message aware* (a related but not redundant notion, needed in the analysis that we introduce in the paper). The similar result is also obtained for 12 PGV output transformations. We also observe that some popular double block length output transformations can not be employed as output transformation.

Keywords: Preimage Awareness, PRO, Random Permutation, Computable Message Awareness.

1 Introduction

Understanding what construction strategy has a chance to be a good hash function is extremely challenging. Further, nowadays it is becoming more important due to the current SHA3 competition which is intended to make a new standard for hash functions. In TCC'04, Maurer *et al.* [16] introduced the notion of indifferentiability as a generalization of the concept of the indistinguishability of two systems [15]. Indifferentiable from a VIL (variable input length) random oracle (also known as PRO or pseudorandom oracle) is the appropriate notion of random oracle for a hash-design. Recently, Dodis, Ristenpart and Shrimpton [7] introduced a generic method to show indifferentiable or PRO security proof of a hash function, whose final output function $\mathcal{R}$ is a FIL (fixed input length) random oracle. More precisely, they defined a new security notion of hash function, called **preimage awareness** (PrA), and showed that $F(M) = \mathcal{R}(H^P(M))$ is PRO provided H^P is preimage aware (supposed to be a weaker assumption). The result is applied to prove the indifferentiable security of the Skein hash algorithm [2], a second round SHA3 candidate. Informally, a hash function H^P is called PrA if the following is true for any adversary A having access to P: For any y committed by A, if a preimage of y is not efficiently "computable" (by an algorithm called *extractor*) from the tuple of all query-responses of P (called *advise string*) then A should not be able to compute it even after making additional P queries. This new notion seems to be quite powerful whenever we have composition of a VIL hash function and a FIL output transformation.

Our Result. We start with a preliminary discussion about the different notions and interrelationship among them. We note that there are hash functions whose final output transformation *cannot be viewed as a random oracle* e.g. some SHA3 second round candidates. So one needs to extended results beyond that of Dodis *et al.* to cover the cases of hash functions with various output transformations which are

in use and this becomes our major objective, since it is important to assure good behavior of these hash functions as well. As a good example of a prevalent transform for construction of hash functions, we choose Davies-Meyer [20] $OT(x) = E(x) \oplus x$ where E is a random permutation and study it in Section 3. We observe that the preimage awareness of H^P is not sufficient for the PRO security of F. In addition to PrA, if H^P is also preimage resistant (PI) and computable message aware (as we define in Section 3.1), then $F^{P,E}$ is PRO (proved in Theorem 1). Informally speaking, a hash function H^P is called **computable message aware** (or CMA) if there exists an efficient extractor (called computable message extractor) which can list the set of all computable messages whose H^P outputs are implied with high probability given the advise string of P. The main difference with PrA is that here, no adversary is involved and the extractor does not get any specific target (see Definition 2). We show that both preimage resistant and CMA are not implied by PrA and hence these properties can not be ignored. Our result can then be employed to prove that a close variant of Grøstl is PRO (see Section 4)[1]. We continue our research in finding other good output transformations. We found 12 out of 20 PGVs can be employed as output transformation OT and we require similar properties of H^P, i.e. PrA, PI and CMA, to have PRO property of $OT(H^P)$ (see Section 5). However these three properties are not sufficient for some DBL post processors. In section 6 we show PRO attacks when some popular double block length post processors are employed. It would be an interesting future research work to characterize the properties of the inner hash function H^P and the output transformation OT such that $OT(H^P)$ become PRO. In the appendix we review the results of [7, 8].

2 Preliminaries

A game is a tuple of probabilistic stateful oracles $G = (\mathcal{O}_1, \ldots, \mathcal{O}_r)$ where states can be shared by the oracles. The oracles can have access to primitives (e.g. random oracle) via black-box modes. It is reasonable to assume that all random sources of games come from the primitives. A probabilistic oracle algorithm A (e.g. an adversary) executes with an input x, its oracle queries being answered by the corresponding oracles of G. Finally it returns $y := A^G(x)$. An adversary A may be limited by different resources such as its runtime, number of queries to different oracles, size of its inputs or outputs, etc. If θ is a tuple of parameters describing the available resources of A then we say that A is a *θ-adversary*. In this paper H^P is an n-bit hash function defined over a message space $\mathcal{M}$ based on a primitive P which can be only accessed via a black-box.

Indifferentiability. The security notion of indifferentiability or PRO was introduced by Maurer *et al.* in TCC'04 [16]. In Crypto'05, Coron *et al.* adopted it as a security notion for hash functions [5]. Let F be a hash function based on ideal primitives $P = (P_1, ..., P_j)$ and $\mathcal{F}$ be a VIL random oracle, let $S^{\mathcal{F}} = (S_1^{\mathcal{F}}, ..., S_j^{\mathcal{F}})$ be a simulator (aimed to simulate $P = (P_1, \ldots, P_j)$) with access to $\mathcal{F}$, where S_i's can communicate to each other. Then, for any adversary A, the indifferentiability- or PRO-advantage of A is defined by $\mathbf{Adv}^{\text{pro}}_{F^P, S^{\mathcal{F}}}(A) = |\Pr[A^{F,P} = 1] - \Pr[A^{\mathcal{F},S} = 1]|$. When the value of the above advantage is negligible, we say that the hash function F is indifferentiable or PRO. Maurer *et al.* [16] also proved that if F is indifferentiable, then $\mathcal{F}$ (a VIL random oracle) used in any secure cryptosystem can be replaced by $F^{P_1,...,P_j}$ with a negligible loss of security. In other words, F can be used as a VIL PRO.

Preimage-Awareness or PrA. Dodis, Ristenpart and Shrimpton defined a new security notion called *Preimage-Awareness* (or PrA) for hash functions [7, 8] which plays an important role in analyzing indifferentiability of a hash function [2]. Given a game G^P (can be P itself), a tuple $\alpha = ((x_1, w_1), \ldots, (x_s, w_s))$ is called an *advise string* at some point of time in the execution of A^G, if w_i's are responses of all P-queries x_i's until that point of time. A PrA (q, e, t)-adversary A (making q queries and running in time t) commits $y_1, \ldots, y_e$ during the execution of A^P and finally returns M. We write

[1] The indifferentiable security analysis of Grøstl has been studied and can be found in [1]

$(y_1, \ldots, y_e) \leftarrow A^P_{\text{guess}}$, $M \leftarrow A^P$ and denote the advise string at the time A^P_{guess} commits y_i by α_i. The guesses and P-queries can be made in any order.

Definition 1. *The* PrA-advantage *of H^P with an extractor $\mathcal{E}$ is defined as* $\mathbf{Adv}^{\text{pra}}_{H,P,\mathcal{E}}(q,e,t) = \max_A \mathbf{Adv}^{\text{pra}}_{H,P,\mathcal{E}}(A$ *where maximum is taken over all (q,e,t)-adversaries A and the PrA advantage of A is defined as*

$$\mathbf{Adv}^{\text{pra}}_{H,P,\mathcal{E}}(A) = \Pr[\exists i,\ H^P(M) = y_i,\ M \neq \mathcal{E}(y_i, \alpha_i) : M \leftarrow A^P; (y_1, \ldots, y_e) \leftarrow A^P_{\text{guess}}]. \quad (1)$$

H^P is called (q, e, t, t_e, ϵ)-PrA if $\mathbf{Adv}^{\text{pra}}_{H,P,\mathcal{E}}(q,e,t) \leq \epsilon$ for some extractor $\mathcal{E}$ with runtime t_e. In short, we say that a hash function is PrA or preimage-aware if there exists an "efficient" extractor such that for all "reasonable" adversaries A the PrA advantage is "small".

Relationship among Collision Resistant, Preimage resistant (PI) and PrA.
If a collision attacker B^P returns a collision pair (M, M'), then a PrA attacker makes all necessary P queries to compute $H^P(M) = H^P(M') = y$ and finally returns M if $\mathcal{E}(y, \alpha) = M'$, o.w. returns M'. So a PrA hash function must be collision resistant. In [7] the authors consider a weaker version of PrA (called weak-PrA) where an extractor can return a set of messages (possibly empty) whose output is y. A PrA adversary wins this new weak game if it can find a preimage of y different from those given by the extractor. They also have shown that PrA is equivalent to collision resistant and weak-PrA. One can modify a definition of a preimage-resistant hash function by introducing only one collision pair. It still remains preimage resistant as the randomly chosen target in the particular collision value has negligible probability. However, it is not preimage-aware since a collision is known. On the other hand $H^P(x) = P^{-1}(x)$ or $H^P(x) = x$ are not preimage resistant but PrA.

3 Hash Function with Output Transformation $OT(x) := E(x) \oplus x$

In [7], hash functions have been analyzed for which the output transformation can be modeled as a FIL random oracle. Generally, we can consider various kinds of output transformations such as Davis-Meyer, PGV compression functions [19] or some DBL (Double Block Length) compression functions [18, 11, 12, 14] in the ideal cipher model. Traditionally, the most popular known design of hash function uses one of the above post-processors. It is well-known that all such compression functions are not indifferentiably secure [5]. So, we need a separate analysis from [7]. In this section, we consider Davis-Meyer transformation $OT(x) = E(x) \oplus x$, where E is a permutation modeled as a "random permutation." A simple example of H^P (e.g. the identity function) tells us that the preimage awareness is not a sufficient condition to have PRO after employing Davis-Meyer post-processor. This suggests that we need something stronger than PrA. We first observe that the preimage attack on identity function can be exploited to have the PRO attack. So preimage resistant can be a necessary condition. We define a variant of preimage resistant, called multipoint-preimage (or mPI), which is actually equivalent to PI. The multipoint-preimage (or mPI) advantage of a (q, t, s)-adversary A (i.e., adversary which makes q queries, runs in time t and has s targets) for H^P is defined as

$$\mathbf{Adv}^{\text{mPI}}_{H^P}(A) = \Pr[\exists i, H^P(M) = h_i : M \leftarrow A^P(h_1, \ldots, h_s);\ h_1, \ldots, h_s \xleftarrow{\$} \{0,1\}^n.] \quad (2)$$

When $s = 1$, it corresponds to the classical preimage advantage $\mathbf{Adv}^{\text{PI}}_{H^P}(A)$. Conversely, mPI advantage can be bounded above by preimage advantage as described in the following. For any (q, t, s)-adversary A with multipoint-preimage advantage ϵ against H^P, there is a $(q, t + O(s))$-adversary A' with preimage-advantage ϵ/s. The adversary A' generates $(s-1)$ targets randomly and embeds his target among these in a random position. So whenever an mPI adversary A finds a multipoint preimage of these s targets, it is the preimage of A's target with probability $1/s$ (since there is no way for A to know the position of the target for A'). W.l.o.g. one can assume that the targets are distinct and chosen at random. Otherwise we remove all repeated h_i's and replace them by some other random distinct targets. So we have the following result.

Lemma 1. *Let $h_1, \ldots h_s$ be distinct elements chosen at random (i.e. outputs of a random permutation for distinct inputs). Then, any (q,t)-adversary A^P can find one of the preimages of h_i's with probability at most $s \times \mathbf{Adv}^{\mathrm{PI}}_{H^P}(q,t)$.*

3.1 Computability

Next we show that preimage resistant and PrA are not sufficient to prove the PRO property. Consider the following example based on an n-bit one-way permutation f and random oracle P.

Example 1. $H^P(m) = P(f(m)) \oplus m$. Given $\alpha = (f(m), w)$ it is hard to find m and hence there is no efficient extractor to find the message m even though an adversary A knows m and its H^P-output. An adversary can compute $z = \mathcal{F}(m)$ and makes $E^{-1}(z \oplus w \oplus m)$ query. No feasible simulator can find message m from it with non-negligible probability and hence cannot return $w \oplus m$. However $w \oplus m$ is the response when A interacts with the real situation $(F^{P,E}, P, E, E^{-1})$. So A can have a PRO attack to F. It is easy to see that H^P is preimage resistant and PrA (given the advise string $\alpha = ((x_1, w_1), \ldots, (x_q, w_q))$ and the target x (that helps to find m back) the extractor finds i for which $f(w_i \oplus x) = x_i$ and then returns $w_i \oplus x$).

The above example motivates us to define a computable message given an advise string. A message M is called computable from α if there exists y such that $\Pr[H^P(M) = y|\alpha] = 1$. In other words, the computation of $H^P(M) = y$ can be made without making any further P-queries. We require the existance of an efficient extractor $\mathcal{E}_{\mathrm{comp}}$, called *computable message extractor*, which can list all computable messages given the advise string. We note that this is not same as weak-PrA as the extractor has to find all messages whose outputs can be computed to a value (unlike PrA, no such fixed target is given here). This notion does not involve any adversary.

Definition 2. *A pair $(H^P, \mathcal{E}_{\mathrm{comp}})$ is called (q, q_H, ϵ)-computable message aware or CMA if for any advise string α with q pairs, the number of computable messages is at most q_H and $\mathcal{E}_{\mathrm{comp}}(\alpha)$ outputs all these. Moreover, for any non-computable messages M, $\Pr[H^P(M) = y|\alpha] \leq \epsilon$, $\forall y$.*

A hash function H^P is called (q, q_H, ϵ, t_c)-computable message aware or CMA if there is $\mathcal{E}_{\mathrm{comp}}$ with run time t_c such that $(H^P, \mathcal{E}_{\mathrm{comp}})$ is (q, q_H, ϵ)-CMA. In short we say that H^P is CMA if it is (q, q_H, ϵ, t_c)-computable message aware where for any feasible q, q_H and t_c are feasible and ϵ is negligible. We reconsider the above example $H^P(m) = P(f(m)) \oplus m$ for an one-way permutation f. We have seen that it is both PI and PrA. However, there is no efficient extractor that can not find all computable messages given the advise string say $(f(m), w)$. In fact, m is computable but there is no way to know it by the extractor only from the advise string (extractor can know if the target $f(m) \oplus w = H^P(m)$ is given).

To be a computable message aware, *the list of computable message has to be small or feasible so that an efficient computable message extractor can exist.* For example, the identity function has a huge set of computable messages given any advise string which cannot be listed by any efficient algorithm even though we theoretically know all these messages.

3.2 PRO Analysis of a Hash Function with Output Transformation $E(x) \oplus x$

In this section we prove that $OT(H^P())$ is PRO whenever H^P is PrA, PI and CMA. We give an informal idea how the proof works. Note that for E-query, $E(x) \oplus x$ almost behaves like a random oracle and hence PrA property of H^P takes care the simulation. This would be similar to the random oracle case except that we have to deal with the fact there is no collision on E. The simulation of responses of P-queries will be same as P. The non-trivial part is to response E^{-1} query. If the E^{-1}-query y is actually obtained by $y = \mathcal{F}(M) \oplus H^P(M)$ then simulator has to find the M to give a correct response.

Since simulator has no idea about the $\mathcal{F}(M)$ as he can not see $\mathcal{F}$-queries, the query y is completely random to him. However, he can list all computable messages and try to compute $H^P(M)$ and $\mathcal{F}(M)$. This is why we need CMA property. The simulator should be able to list all computable messages only from the P query-responses. If he finds no such messages then he can response randomly. The simulator would be safe as long as there is no preimage attack to the random output. No we provide a more formal proof.

Let $F^{P,E}(M) = E(H^P(M)) \oplus H^P(M)$ and A be a PRO adversary making at most (q_0, q_1, q_2, q_3) queries to its four oracles with bit-size l_{max} for the longest $\mathcal{O}_0$-query. We assume that $H^P(\cdot)$ is preimage resistant and (q^*, q_H, ϵ)-computable message aware for an efficient computable message extractor $\mathcal{E}_{\text{comp}}$ where $q^* = q_1 + q_2\mathsf{NQ}[l_{max}]$. Let $q = q_H + q'$ and $q' = q_0 + q_1 + q_2 + q_3$. For any given PrA-extractor $\mathcal{E}$, we can construct a simulator $S^{\mathcal{F}} = (S_1, S_2, S_3)$ (defined in the oracles of C_A in Fig. 3) that runs in time $t^* = O(q^2 + q_3\mathsf{Time}(\mathcal{E}_{\text{comp}}))$. Given any indifferentiability adversary A making at most (q_0, q_1, q_2, q_3) queries to its four oracles with bit-size l_{max} for the longest $\mathcal{O}_0$-query, there exists a PrA $(q, q_2 + 1, t)$-adversary C_A with runtime $t = \mathsf{Time}(A) + O(q_2 \cdot \mathsf{Time}(\mathcal{E}) + q_0 + q_1 + (q_2 + q_0)\mathsf{NQ}[l_{max}])$. Now we state two lemmas which are useful to prove the main theorem of the section. Proof ideas of these lemmas are very similar to that of Lemma 9 and Lemma 11. The games $G4$ and $G5$ are defined in the Fig. 2. We use a simulation oracle simE which works given a runtime database E. A random element from the set $\{0,1\}^n \setminus \mathit{Range}(\mathsf{E})$ is returned for $\mathsf{simE}[1, y]$ whenever y is not in the domain of E. Similarly a random element from the set $\{0,1\}^n \setminus \mathit{Domain}(\mathsf{E})$ is returned in $\mathsf{simE}[-1, c]$ whenever c is not in the range of E. Whenever y or c are defined before the simulator oracle just returns the previous returned value. We use three such simulation oracles for E_0 (which keeps the input output behavior of E due to $\mathcal{O}_0$ queries only), E_1 (which keeps the input output behavior of E due to $\mathcal{O}_2$ and $\mathcal{O}_3$ queries) and $\overline{\mathsf{E}}$ (which keeps the input output behavior of E for all queries, i.e. it is the union of the previous two unions).

Lemma 2. *$G4 \equiv (F^{P,E}, P, E, E^{-1})$, $G5 \equiv (\mathcal{F}, S_1, S_2, S_3)$ and $G4, G5$ are identical-until-Bad.*

Proof. It is easy to see from the pseudocode that $G4, G5$ are identical-until-Bad. The games $G5$ and the oracles simulated by C_A (same as $(\mathcal{F}, S_1, S_2, S_3)$) are actually identical. They have common random sources which are namely $\mathcal{F}$, P and the simulated oracle simE_0 (we can ignore the dead conditional statements which have boxed statements which are not actually executed in game $G5$). Now it remains to show that $G5$ is equivalent to a real game for a PRO attacker. Note that oracles $\mathcal{O}_2$ and $\mathcal{O}_3$ are statistically equivalent to a random permutation $\overline{\mathsf{E}}$ and its inverse which are simulated runtime. Moreover $\mathcal{O}_0$ returns $\mathcal{F}(M)$ if $\overline{\mathsf{E}}[y], \overline{\mathsf{E}}^{-1}[c]$ are undefined or $\overline{\mathsf{E}}[y] = c$ where $y = H^P(M)$ and $c = \mathcal{F}(M) \oplus y$. In all other cases $\mathcal{O}_0(M)$ either computes or simulates $c' = \overline{\mathsf{E}}[y]$ and returns $c' \oplus y$. So $\mathcal{O}_0(M)$ is statistically equivalent to the oracle $\overline{\mathsf{E}}(H^P()) \oplus H^P()$. Hence $G4$ is statistically equivalent to $(F^{P,E}, P, E, E^{-1})$. ∎

The following result follows immediately from the fact that $\mathcal{F}$ and H^P are statistically independent and $\mathcal{F}$ is a random oracle.

Lemma 3. *For any adversary $C^{P,\mathcal{F}}$ making q queries to the n-bit random oracle $\mathcal{F}$ we have* $\Pr[\mathcal{F}(M) \oplus H^P(M) = \mathcal{F}(M') \oplus H^P(M'), M \neq M' : (M, M') \leftarrow C] \leq q(q-1)/2^{n+1}$.

Lemma 4. *Whenever A^{G5} sets Bad true, C_A also sets one of the Bad events true. Moreover,*

$$\Pr[C_A \text{ sets } \mathsf{Bad} \text{ true }] \leq \mathbf{Adv}^{\text{pra}}_{H^P,P,\mathcal{E}}(C_A) + q_3 \times \mathbf{Adv}^{\text{PI}}_{H^P}(q,t) + q_0 q_3 \epsilon + \frac{2q_0q_3 + q_2q_0}{2^n - q_0 - q_2 - q_3} + \frac{(q_H + q_2 + q_0)^2}{2^{n+1}}.$$

Proof. The first part of the lemma is straightforward and needs to be verified case by case. We leave the details for readers to verify. It is easy to see that whenever Bad_{pra} sets true C is successful in a PrA attack. Now we estimate the probability of the other bad events, from which the theorem follows.

Game $\boxed{G4}$ and $G5$

Initialize : $\mathsf{E} = \mathsf{E}_0 = \mathsf{E}_1 = \phi$; $\mathsf{H} = \mathsf{H}' = \beta = \phi$, Bad =F;

300 On $\mathcal{O}_3$ - query c
301 $S = \{X_1, \ldots, X_r\} = \mathcal{E}_{\text{comp}}(\beta)$;
302 For all $1 \leq i \leq r$ do
303 $y_i = H^{\mathcal{O}_1}(X_i) := \mathsf{H}[X_i]$; $\mathcal{F}(X_i) = z_i$; $c_i = y_i \oplus z_i$;
304 If $\exists$ unique i s.t. $c_i = c$,
305 If $\mathsf{E}_1[y_i] = \perp$ then $y = y_i$;
306 Else if $\mathsf{E}_1[y_i] = c' \neq \perp$ then $\mathsf{Bad} = T$; $y = \mathsf{sim}\mathsf{E}_1[-1, c]$;
307 Else if no i s.t $c_i = c$, then $y = \mathsf{sim}\mathsf{E}_1[-1, c]$;
308 Else Bad =T; $y = \mathsf{sim}\mathsf{E}_1[-1, c]$;
309 If $\mathsf{E}_0^{-1}[c] \neq \perp$ and $\mathsf{E}_0^{-1}[c] \neq y_i$ then $\mathsf{Bad} = T$; $\boxed{y = \mathsf{E}_0^{-1}[c];}$
310 If y is E_1-simulated and $\mathsf{E}_0[y] \neq \perp$ then $\mathsf{Bad} = T$; $\boxed{y = \mathsf{sim}\overline{\mathsf{E}}[-1, c];}$
311 $\mathsf{E}_1[y] := c$; $\bar{\mathsf{E}}[y] := c$; return y;

200 On $\mathcal{O}_2$ - query y
201 $X = \mathcal{E}(y, \beta)$; $\mathsf{Ext} \xleftarrow{\cup} (y, X)$;
202 $y' = H^{\mathcal{O}_1}(X)$, $\mathsf{H} \xleftarrow{\cup} (X, y')$; $z = \mathcal{F}(X)$; $c = z \oplus y'$;
203 If $y' \neq y$ then
204 $c' = \mathsf{sim}\mathsf{E}_1[1, y]$;
205 If $\mathsf{E}_0[y] \neq \perp$ then Bad =T; $\boxed{c' = \mathsf{E}_0[y];}$
206 Else if $c' \in \mathsf{Range}(\mathsf{E}_0)$ then Bad =T; $\boxed{c' = \mathsf{sim}\bar{\mathsf{E}}[1, y];}$
207 $\mathsf{E}_1[y] := c'$; $\bar{\mathsf{E}}[y] := c'$; return c';
208 If $y' = y$ and $c \in \mathsf{Range}(\mathsf{E}_1)$ then
209 $c' = \mathsf{sim}\mathsf{E}_1[1, y]$;
210 If $\mathsf{E}_0[y] \neq \perp$, then Bad =T; $\boxed{c' = \mathsf{E}_0[y];}$
211 Else if $c' \in \mathsf{Range}(\mathsf{E}_0)$ then Bad =T; $\boxed{c' = \mathsf{sim}\bar{\mathsf{E}}[1, y];}$
212 $\mathsf{E}_1[y] := c'$; $\bar{\mathsf{E}}[y] := c'$; return c';
213 If $y' = y$ and $c \notin \mathsf{Range}(\mathsf{E}_1)$ then
214 If $\mathsf{E}_0[y] \neq \perp$, then Bad =T; $\boxed{c = \mathsf{E}_0[y];}$
215 Else if $c \in \mathsf{Range}(\mathsf{E}_0)$ then Bad =T; $\boxed{c = \mathsf{sim}\bar{\mathsf{E}}[1, y];}$
216 $\mathsf{E}_1[y] := c$; $\bar{\mathsf{E}}[y] := c$; return c;

100 On $\mathcal{O}_1$ - query u
101 $v = P(u)$; $\beta \xleftarrow{\|} (u, v)$;
102 return v;

000 On $\mathcal{O}_0$ - query M
001 $z = \mathcal{F}(M)$; $y = H^P(M)$; $c = z \oplus y$;
002 If $\exists\, M'$ s.t. $M \neq M'$ and $(M', y) \in \mathsf{H}'$ then Bad =T; $\mathsf{H} \xleftarrow{\cup} (M, y)$; $\boxed{z = \mathcal{F}(M');}$ return z;
003 $\mathsf{H}' \xleftarrow{\cup} (M, y)$; $\mathsf{H} \xleftarrow{\cup} (M, y)$;
004 If $\bar{\mathsf{E}}[y] = \perp \wedge \bar{\mathsf{E}}^{-1}[c] = \perp$ then $\bar{\mathsf{E}}[y] := c$; $\mathsf{E}_0[y] := c$; return z;
005 If $\bar{\mathsf{E}}[y] = c$ then $\mathsf{E}_0[y] := c$; return z;
006 If $\bar{\mathsf{E}}[y] \neq \perp \wedge \bar{\mathsf{E}}[y] \neq c$ then Bad =T; $\boxed{z = \bar{\mathsf{E}}[y] \oplus y;}$ return z;
007 If $\bar{\mathsf{E}}[y] = \perp \wedge \bar{\mathsf{E}}^{-1}[c] \neq \perp$ then Bad =T; $\mathsf{sim}\bar{\mathsf{E}}[1, y] = c'$; $\bar{\mathsf{E}}[y] := c'$; $\mathsf{E}_0[y] = c'$; $\boxed{z = c' \oplus y;}$ return z;

Fig. 1. $G4$ executes with boxed statements whereas $G5$ executes without these. G_4 and G_5 perfectly simulate $(F^{P,E}, P, E, E^{-1})$ and $(\mathcal{F}, S_1, S_2, S_3)$, respectively. Clearly $G4$ and $G5$ are identical-until-Bad.

The Oracles $\mathcal{O}_2$ (or S_2) and $\mathcal{O}_3$ (or S_3)	The oracles $\mathcal{O}_0$ (or $\mathcal{F}$), $\mathcal{O}_1$ (or P) and Finalization
/The VIL random oracle $\mathcal{F}$ is simulated by C_A/	/The VIL random oracle $\mathcal{F}$ is simulated by C_A/
Initialize : $\mathsf{E}_1 = \mathcal{L} = \mathcal{L}_1 = \mathsf{F}' = \mathsf{H} = \beta = \phi$;	100 On $\mathcal{O}_1$ - query u
Run A and response its oracles	101 $v = P(u)$; $\beta \xleftarrow{\parallel} (u, v)$; return v;
300 On $\mathcal{O}_3$ - query c	
301 $S = \{X_1, \ldots, X_r\} = \mathcal{E}_{\text{comp}}(\beta)$;	000 On $\mathcal{O}_0$ (or $\mathcal{F}$)- query M
302 For all $1 \le i \le r$ do	001 $z = \mathcal{F}(M)$; $\mathcal{L} \xleftarrow{\cup} M$;
303 $y_i = H^{\mathcal{O}_1}(X_i) := \mathsf{H}[X_i]$; $\mathcal{F}(X_i) = z_i$;	
304 $c_i = y_i \oplus z_i$; $\mathsf{F}'[X_i] = c_i$; $\mathcal{L}_1 \xleftarrow{\cup} X$;	Finalization()
305 If $\exists$ unique i, $c_i = c$,	501 If collision in F' then Bad_{F1} =T;
306 If $\mathsf{E}_1[y_i] = \perp$ then $y = y_i$	502 If collision in H then Bad_{PrA} =T; $\mathsf{Finish}()$;
307 Else if $\mathcal{O}_3(c') = y_i$ was queried and	503 For all $M \in \mathcal{L}$ do
308 no i on that query then	504 $z = \mathcal{F}(M)$, $\mathsf{H}[M] = H^P(M) = y$;
309 $\mathsf{Bad}_{PI} = T$; $y = \mathsf{simE}_1[-1, c]$;	505 $\mathsf{F}'[M] = c = y \oplus z$;
310 Else if no i then $y = \mathsf{simE}_1[-1, c]$;	506 If $\mathsf{F}'[X] = c$, $X = M$ then Bad_{F1} =T;
311 Else Bad_{F1} =T;$y = \mathsf{simE}_1[-1, c]$;	507 If $\mathsf{H}[X] = y, X = M$ then Bad_{PrA} =T; $\mathsf{Finish}()$;
312 $\mathsf{E}_1[y] = c$; return c;	508 If $\mathsf{Ext}[y] = \perp, M$ then Bad_{PrA} =T; $\mathsf{Finish}()$;
	509 If $\mathcal{O}_2(y) = c_i$, $y = y_i$
200 On $\mathcal{O}_2$ - query $y := y_i$, $i = i + 1$	510 then Bad_{E1} =T;
201 $X = \mathcal{E}(y_i, \beta)$; $\mathsf{Ext} \xleftarrow{\cup} (y, X)$;	511 Else if $\mathcal{O}_3(c_i) = y = y_i$ after M_i-query
202 $y' = H^{\mathcal{O}_1}(X)$, $\mathsf{H} \xleftarrow{\cup} (X, y')$;$z = \mathcal{F}(X)$;	512 then Bad_{comp} =T;
203 $\mathcal{L}_1 \xleftarrow{\cup} X$; $c = z \oplus y$; $\mathsf{F}'[X] = c$;	513 Else if $\mathcal{O}_3(c_i) = y = y_i$ before M_i-query
204 If $y' = y$	514 then Bad_{F2} =T;
205 then $c = \mathsf{simE}_1[1, y]$;	515 Else if $\mathcal{O}_3(c) = y_i$ after M_i-query, $c = c_i$
206 If $y' = y$, $\mathcal{O}_3(c)$ was queried	516 then Bad_{E2} =T;
207 then Bad_{F1} =T; $c = \mathsf{simE}_1[1, y]$	517 Else if $\mathcal{O}_3(c) = y_i$ before M_i-query, $c = c_i$
208 If $y' = y$, $\mathcal{O}_2(y'') = c$ was queried	518 then Bad_{PI} =T;
209 then $c = \mathsf{simE}_1[1, y]$	519 return $\perp$;
210 $\mathsf{E}_1[y] = c$; return c;	

Fig. 2. The oracles simulated by PrA adversary C_A to response an PRO adversary A. It has a finalization procedure which also sets some bad event true. $\mathsf{Finish}()$ which is defined similarly as in Fig. 7.2. It mainly completes the PrA attack. It is easy to see that whenever $\mathsf{Finish}()$ is being executed either we have a collision in H^P or there is some message M such that $H^P(M) = y, (y, M) \in \mathsf{Ext}$.

1. $\Pr[\mathsf{Bad}_{mPI} = T] \le q_3 \times \mathbf{Adv}^{PI}_{H^P}(q^*, t^*)$. It is easy to see that whenever Bad_{mPI} sets true we have a preimage of some y_i which is generated from simE_1. Note that simE_1 responds exactly like a random permutation. So by lemma 1 we have the bound.
2. $\Pr[\mathsf{Bad}_{comp} = T] \le q_0 q_3 \epsilon$. Whenever Bad_{comp} sets true we must have $H^P(M_i) = y_i$ where M_i is not computable (since it is not in the list given by $\mathcal{E}_{\text{comp}}$). So from computable message awareness definition we know that $\Pr[H^P(M_i) = y_i] \le \epsilon$. The number of such M_i's and y_i's are at most $q_0 q_3$.
3. All other bad events hold due to either the special outputs of $\mathcal{F}(M)$ (when $\mathsf{Bad}_{F1} = T$ or $\mathsf{Bad}_{F2} = T$, we apply the lemma 3) or the special outputs of $\mathsf{simE}(c)$ (when $\mathsf{Bad}_{E1} = T$ and $\mathsf{Bad}_{E2} = T$). One can show the following:

$$\Pr[\mathsf{Bad}_{E1 \vee E2 \vee F1 \vee F2} = T] \le \frac{2q_0q_3 + q_2q_0}{2^n - q_0 - q_2 - q_3} + \frac{(q_H + q_2 + q_0)^2}{2^{n+1}}.$$

We have used Lemma 3 to bound the bad event Bad_{F1}. The other bad event probability calculations are straightforward to calculate. We leave details to readers. ■

The main theorem of the section follows from the above lemmas.

Theorem 1. *For any indifferentiability adversary A making at most (q_0, q_1, q_2, q_3) queries to its four oracles with bit-size l_{max} for the longest $\mathcal{O}_0$-query, there exists a PrA (q, q_2+1, t)-adversary C_A with runtime $t = \mathsf{Time}(A) + O(q_2 \cdot \mathsf{Time}(\mathcal{E}) + q_0 + q_1 + (q_2+q_0)\mathsf{NQ}[l_{max}])$ and*

$$\mathbf{Adv}^{\mathrm{pro}}_{F,S}(A) \leq \mathbf{Adv}^{\mathrm{pra}}_{H^P,P,\mathcal{E}}(C_A) + q_3 \times \mathbf{Adv}^{\mathrm{PI}}_{H^P}(q,t) + q_0 q_3 \epsilon + \frac{2q_0q_3 + q_2q_0}{2^n - q_0 - q_2 - q_3} + \frac{(q_H + q_2 + q_0)^2}{2^{n+1}},$$

where $H^P(\cdot)$ is preimage resistant and (q^*, q_H, ϵ)-computable message aware for an efficient computable message extractor $\mathcal{E}_{\mathrm{comp}}$ where $q^* = q_1 + q_2\mathsf{NQ}[l_{max}]$.

4 Application of Theorem 1: PRO analysis of a variant of Grøstl

As an application of Theorem 1 we prove the PRO analysis of a variant of Grøstl hash function in which the output transformation is based on a permutation independent of the permutations used in iteration. The compression function $f^{P,Q}(z,m) = P(z \oplus m) \oplus Q(m) \oplus z$, where P and Q are invertible permutations on n-bit modeled to be independent random permutations (adversary can also have access to inverses). The hash function $H^{P,Q}$ of Grøstl without output transformation is Merkle-Damgård with strengthening (SMD) and the output transformation is $\mathrm{trunc}_s(P(x) \oplus x)$. In case of the variant of the hash function, the output transformation is same as the previous section, i.e. $OT(x) = E(x) \oplus x$ where E is a random permutation independent with P and Q. Since SMD preserves preimage awareness and preimage resistance of the underlying compression function, we focus on the proof of the compression function $f^{P,Q}$ to prove PrA and preimage resistance.

Lemma 5. *For any advise string α_P and α_Q of sizes (q_1+q_2) and (q_3+q_4) (for (P, P^{-1}) and (Q, Q^{-1}) respectively) the number of computable messages is at most $q_f \leq (q_1+q_2)(q_3+q_4)$ and for any non-computable message (z,m), $\Pr[f^{P,Q}(z,m) = c|\alpha_P, \alpha_Q] \leq \frac{1}{2^n - max(q_1+q_2, q_3+q_4)}$. Moreover there is an efficient computable message extractor $\mathcal{E}^f_{\mathrm{comp}}$ which can list all computable messages.*

The proof of the above lemma is straightforward and is left to readers to verify. Let $\mathbf{q} = (q_1, q_2, q_3, q_4)$. Now given the computable message extractor one can define a PrA extractor $\mathcal{E}^f$ as follows: $\mathcal{E}^f(y, \alpha_P, \alpha_Q) = (z,m)$ if there exists an unique computable message (from the list given by $\mathcal{E}^f_{\mathrm{comp}}$) such that $f(z,m) = y$, otherwise it returns any arbitrary message.

Lemma 6. $\mathrm{Adv}^{\mathrm{PI}}_{H^{P,Q}}(\mathbf{q}, t) \leq \mathrm{Adv}^{\mathrm{PI}}_{f^{P,Q}}(\mathbf{q}, t) \leq \frac{(q_1+q_2)(q_3+q_4)}{2^n - max(q_1+q_2, q_3+q_4)}$ *for any t.*

$H^{P,Q}$ of Grøstl is a function based on SMD (Merkle-Damgård with strengthening). Since SMD preserves preimage awareness of the underlying compression function [7, 8], we focus on the proof of the compression function of $H^{P,Q}$. The compression function $f(h,m) = P(h \oplus m) \oplus Q(m) \oplus h$, where P and Q is invertible permutations on n-bit. For the preimage awareness proof of f, we assume that P and Q are independent ideal permutations.

Lemma 7. *Let $\mathbf{q} = (q_1, q_2, q_3, q_4)$ and let $f^{P,Q} = P(h \oplus m) \oplus Q(m) \oplus h$, where P and Q are invertible ideal permutations. For any preimage awareness $(\mathbf{q}, e, t)$-adversary A making at most $\mathbf{q}$ queries to the oracles P, P^{-1},Q, Q^{-1}, there exists an extractor $\mathcal{E}$ such that*

$$\mathrm{Adv}^{\mathrm{pra}}_{f^{P,Q},P,Q,\mathcal{E}}(A) \leq \frac{e(q_1+q_2)(q_3+q_4)}{2^n - max(q_1+q_2, q_3+q_4)} + \frac{(q_1+q_2)^2(q_3+q_4)^2}{2(2^n - max(q_1+q_2, q_3+q_4))},$$

Proof. We use Lemma 3.3 and Lemma 3.4 in [8]. The two lemmas are related to the definition of the weak preimage awareness, where the extractor is a honest multi-point extractor $\mathcal{E}^+$ which can output a set $\mathcal{X}$, all elements of $\mathcal{X}$ should be real preimages for a committed value. So, we have to show

the following inequalities. First, we need to show that there exists a honest multi-point extractor $\mathcal{E}^+$ such that for any weak preimage awareness $(\mathbf{q}, 1, t)$-adversary B, $\mathrm{Adv}^{1-\mathrm{wpra}}_{f^{P,Q},P,Q,\mathcal{E}^+}(B) \leq \frac{(q_1+q_2)(q_3+q_4)}{2^n}$. Second, we need to show that for any collision-finding adversary C making the same number of oracle queries of A, $\mathrm{Adv}^{\mathrm{cr}}_{f^{P,Q},P,Q}(C) \leq \frac{(q_1+q_2)(q_3+q_4)}{2^{n+1}}$. Then, by applying Lemma 3.3 and Lemma 3.4 in [8], the theorem holds.Let α_i be the advice string at the time when $A^{P,P^{-1},Q,Q^{-1}}_{\mathrm{guess}}$ commits y_i and initialize $\mathcal{X} = \emptyset$.

algorithm $\mathcal{E}^+(y_i, \alpha_i)$:
For all M's such that $f^{P,Q}$ is computable from α_i,
if $f^{P,Q}(M) = y_i$, then $\mathcal{X} = \mathcal{X} \cup \{M\}$
Return $\mathcal{X}$

It is easy to check that all elements of $\mathcal{X}$ are real preimages for the committed value y_i. Note that B is any weak preimage awareness $(\mathbf{q}, 1, t)$-adversary. So, once $B^{P,P^{-1},Q,Q^{-1}}_{\mathrm{guess}}$ commits y, B wins the weak preimage awareness game if B finds a new preimage image M' such that $f^{P,Q}(M') = y$ and $M' \notin \mathcal{X}$. Since P and Q are independent ideal permutations and B can make at most $\mathbf{q}$ queries to P, P^{-1},Q, Q^{-1}, there are $(q_1 + q_2)(q_3 + q_4)$ computable messages and $\Pr[f^{P,Q}(M) = y] \leq \frac{1}{2^n - max(q_1+q_2,q_3+q_4)}$ for each M. So the probability that B wins is at most $\frac{(q_1+q_2)(q_3+q_4)}{2^n - max(q_1+q_2,q_3+q_4)}$.

Next, we consider the advantage of collision-resistance of $f^{P,Q}$. Since B can make at most $\mathbf{q}$ queries, there exist at most $(q_1 + q_2)(q_3 + q_4)$ M's such that $f^{P,Q}(M)$ is computable. And the probability that $f^{P,Q}(M) = f^{P,Q}(M')$ for $M \neq M'$ is at most $\frac{1}{2^n - max(q_1+q_2,q_3+q_4)}$. Therefore, the probability that there exists a collision of $f^{P,Q}$ is at most $\frac{(q_1+q_2)^2(q_3+q_4)^2}{2(2^n - max(q_1+q_2,q_3+q_4))}$. ■

Now we restrict to all advise strings which do not have any collision on $f^{P,Q}$ and we call such an advise string *a non-collision advise string*. For any such advise string, the number of computable messages for a hash function should be at most the number of computable messages for the compression function since the last invocations of the compression function must be distinct for all computable messages (otherwise we have a collision). So $q_H \leq (q_1+q_2)(q_3+q_4)$. There is also an efficient computable message extractor $\mathcal{E}_{\mathrm{comp}}$ which can list all computable messages. One can construct the list in a backward manner. That is, starting from the final output of a computable message one can find out the last chaining value and message block (there cannot be more than one as there are no collisions). We can go back until we get a chaining value which is same as the initial value or the length of the message becomes more than ℓ_{max}. Let t_{comp} denote the runtime of the computable message extractor. The following lemma says that for any non-computable message the outputs are unpredictable and hence $H^{P,Q}$ is computable message aware.

Lemma 8. *For any non-computable message M,*

$$\Pr[H^{P,Q}(M) = y|\alpha] \leq \frac{(q_1 + q_2 + 1)(q_3 + q_4 + 1)}{2^n - max(q_1 + q_2, q_3 + q_4)} + \frac{l_{max}(q_1 + q_2 + l_{max} + 1)(q_3 + q_4 + l_{max})}{2^n - (q_1 + q_2 + l_{max})} := \epsilon,$$

where l_{max} is the maximum block-length of the padded input massage of $H^{P,Q}$. Hence $H^{P,Q}$ is $(\mathbf{q}, (q_1 + q_2)(q_3 + q_4), \epsilon, t_{\mathrm{comp}})$-CMA.

Proof. Let h_i and g_i be the input and the output of P of the i-th compression function $f^{P,Q}$. Let z_i be the output of the i-th compression function $f^{P,Q}$ and let α be the advise string at some point. Since the number of queries to (P,P^{-1},Q,Q^{-1}) is (q_1, q_2, q_3, q_4), α has the information of $(q_1 + q_2)$ input-output pairs of P and $(q_3 + q_4)$ input-output pairs of Q. Since the number of blocks of each padded message is bounded by l_{max}, there exist at most $h_{l_{max}}$ for each padded message. Now we want

to compute the computable message awareness advantage for any $(q = (q_1, q_2, q_3, q_4), t)$-adversary A. In other words, we want to compute the upper bound of the following probability: For any advise string α of at most size $q = (q_1, q_2, q_3, q_4)$ and any M which is not computable from α and any y, $\Pr[H^{P,Q}(M) = y|\alpha] \leq \epsilon$, where ϵ is as given in the theorem. Since M is not computable from α, there exists the smallest i such that we cannot compute z_i. We let the block length of the padded message corresponding to M be ℓ, where $\ell \leq l_{max}$.

We consider two cases. One is the case of $i = \ell$. The other is the case of $i < \ell$. In the first case, we can compute easily that $\Pr[H^{P,Q}(M) = y|\alpha] \leq max(\frac{q_3+q_4}{2^n-(q_1+q_2)}, \frac{q_1+q_2}{2^n-(q_3+q_4)})$. So we have only to compute the bound in the second case. Firstly, we compute an upper bound ϵ_1 of the probability that one of h_{i+1},...,h_ℓ is obtained as an input of P from α, which means that one of h_i,...,h_ℓ is not new. Secondly, we compute the upper bound ϵ_2 of the probability that $H^{P,Q}(M) = y$ under the condition that all h_i,...,h_ℓ are new and α is given.

$\Pr[H^{P,Q}(M) = y|\alpha]$
$\leq \Pr[\exists\ j\ s.t.\ h_j\ isn't\ new] + \Pr[H^{P,Q}(M) = y|\alpha \wedge (h_i, ..., h_\ell\ are\ new)]\Pr[h_i, ..., h_\ell\ are\ new]$
$\leq \Pr[\exists\ j\ s.t.\ h_j\ isn't\ new] + \Pr[H^{P,Q}(M) = y|\alpha \wedge (h_i, ..., h_\ell\ are\ new)] \leq \epsilon_1 + \epsilon_2$

Claim 1. $\Pr[\exists\ j\ s.t.\ h_j\ isn't\ new] \leq \frac{(q_1+q_2+1)(q_3+q_4+1)}{2^n-max(q_1+q_2,q_3+q_4)} + \frac{l_{max}(q_1+q_2+l_{max})(q_3+q_4+l_{max})}{2^n-(q_1+q_2+l_{max})} = \epsilon_1$
Proof. Note that no attacker obtains z_i and $i < \ell$. Let C_j be the event that h_j is not new for $i + 1 \leq j \leq \ell$. So, $\Pr[\exists\ j\ s.t.\ h_j\ isn't\ new] = \Pr[C_{i+1}] + \sum_{j=i+2}^{\ell} Pr[C_j|\overline{C_{j-1}}]$. Since z_i is not known from α, and P and Q are independent random permutations, $\Pr[C_{i+1}] \leq \frac{(q_1+q_2+1)(q_3+q_4+1)}{2^n-max(q_1+q_2,q_3+q_4)}$. And for $j \geq i + 2$ $Pr[C_j|\overline{C_{j-1}}] \leq \frac{(q_1+q_2+j-i)(q_3+q_4+j-i)}{2^n-(q_1+q_2+j-i)}$. ■

Claim 2. $\Pr[H^{P,Q}(M) = y|\alpha \wedge (h_i, ..., h_\ell\ are\ new)] \leq \frac{q_3+q_4+\ell-i}{2^n-(q_1+q_2+\ell-i)} \leq \frac{q_3+q_4+l_{max}}{2^n-(q_1+q_2+l_{max})} = \epsilon_2$
Proof. That h_ℓ is new means that the input of P of the final compression function is new, that is, the output g_ℓ of P is almost random. And y is a given fixed value. So, with the distribution of g_i we can compute the probability regardless of the output of Q of the final compression function. Note that h_i's are not the same. So, we need the term ℓ in the above inequality. ■ ■

Theorem 2. *Let* $Grøstl'(M) = P'(H^{P,Q}(M)) \oplus H^{P,Q}(M)$ *where* $H^{P,Q}$ *is Grøstl without the output transformation and* P, Q, P' *are independent random permutations. Then for any adversary making at most* q *queries to all its oracles the PRO-advantage is bounded by* $\ell_{max}q^2q'^2/2^{n-2}$ *if If* $q' = q_1 + q_2 + q_3 + q_4 + l_{max} \leq 2^{n-1}$.

Proof. The result follows from Lemma 9, 10, 11 and 12, and Theorem 1. ■

Remark 1. Our bound $\ell_{max}q^2q'^2/2^{n-2}$ in the variant of grostl seems to be reasonable as we indeed have collision on the compression function in $2^{n/4}$ complexity i.e. the collision advantage is $q^4/2^n$. We believe the designers also noted that and this is why they consider at least double length hash function. We also strongly believe that the same bound can be achieved for original Grostl. However to prove that we cannot apply Theorem 2 directly. This would be our one of the future research work.

5 PRO Analysis of Hash Functions with PGV Output Transformations

In the previous section, we considered the case that the finial output transformation is $OT(x) = E(x) \oplus x$. In this section, we consider 20 PGV compression functions shown in Table 1 as candidates of the final output transformation OT. Such 20 PGV hash functions based on them were proved

to be collision resistant in the ideal cipher model [4]. More precisely, we will consider the case that $F^{P,E}(M) = OT(H^P(M_1), M_2)$, where E is an ideal cipher, $M = M_1||M_2$, $H^P(M_1)$ corresponds to h_{i-1} and M_2 corresponds to m_i in Table 1. Except for PGV 11, 13, 15-20 (See Example 2 and 3), Theorem 3 holds. The proof of the Theorem 3 is same as Davis-Meyer case. However we give the proof idea so that the reader could justify themselves.

Theorem 3. *[PRO Construction via 12 PGVs] Let $F^{P,E}(M) = OT(H^P(M_1), M_2)$, where $M = M_1||M_2$, OT is any PGV constructions except for PGV 11, 13, 15-20, E^{-1} is efficiently computable, and E is an ideal cipher. For any indifferentiability adversary A making at most (q_0, q_1, q_2, q_3) queries to its four oracles with bit-size l_{max} for the longest $\mathcal{O}_0$-query, there exists a PrA (q, q_2+1, t)-adversary C_A with runtime $t = \mathsf{Time}(A) + O(q_2 \cdot \mathsf{Time}(\mathcal{E}) + q_0 + q_1 + (q_2 + q_0)\mathsf{NQ}[l_{max}])$ and*

$$\mathbf{Adv}^{\mathrm{pro}}_{F,S}(A) \leq \mathbf{Adv}^{\mathrm{pra}}_{H^P,P,\mathcal{E}}(C_A) + q_3 \times \mathbf{Adv}^{\mathrm{PI}}_{H^P}(q,t) + q_0 q_3 \epsilon + \frac{2q_0q_3 + q_2q_0}{2^n - q_0 - q_2 - q_3} + \frac{(q_H + q_2 + q_0)^2}{2^{n+1}},$$

where $H^P(\cdot)$ is preimage resistant and (q^*, q_H, ϵ)-computable message aware for an efficient computable message extractor $\mathcal{E}_{\mathrm{comp}}$ where $q^* = q_1 + q_2\mathsf{NQ}[l_{max}]$.

Proof Idea for Theorem 3: Like to the Davis-Meye case we only need to worry about the E^{-1} query since we have chosen those PGV compression functions which behave like random oracle if adversary makes only E queries. Note that 5-10 and 12 and 14 PGV have w_i as keys. So given a $E_w^{-1}(y)$ query simulator can make the list of all h which can be computed, i.e. $H^P(M) = h$, and guess $m = h \oplus w$. Once simulator guesses m he can make $\mathcal{F}$ queries (M, m) and obtains responses z's. Now simulator can find a correct m if y is really obtained by some $\mathcal{F}(M, m)$. If there is no such m, simulator can response randomly and any bad behavior would be either bounded by the collision probability or by the preimage attack. The same argument works for PGV 1-4 since $H^P(M)$ is xored with the the $E()$ output. So simulator can verify the correct h among all computable hash outputs. ■

Example 2. See PGV 15 in Table 1, which is $E_{m_i}(w_i) \oplus v$, where $w_i = h_{i-1} \oplus m_i$ and v is a constant. Now we want to give an indifferentiable attack on $F^{P,E}$ based on PGV 15, even though H^P is preimage aware, preimage resistant, and (q, q_H, ϵ)-computable message aware with feasible q and q_H and negligible ϵ. Let H^P be an Merkle-damgård construction with strengthening, where the underlying compression function is a preimage aware function based on the ideal primitive P. As shown in [7, 8], SMD (Merkle-damgård construction with strengthening) preserves preimage awareness of the compression function. Also SMD preserves preimage resistance. So, H^P is also preimage aware and preimage resistant. We assume that H^P is (q, q_H, ϵ)-computable message aware with feasible q and q_H and negligible ϵ. Now we construct an indifferentiability adversary A for $F^{P,E}(M) = OT(H^P(M_1), M_2)$, where $OT(x, y) = E_y(x \oplus y) \oplus v$ is PGV 15, and v is a constant. First, A chooses a random query $M = M_1||M_2$ to $\mathcal{O}_1$, where $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \mathcal{O}_4)$ is $(F^{P,E}, P, E, E^{-1})$ or $(\mathcal{F}, S_1^{\mathcal{F}}, S_2^{\mathcal{F}}, S_3^{\mathcal{F}})$ for any simulator $S^{\mathcal{F}} = (S_1^{\mathcal{F}}, S_2^{\mathcal{F}}, S_3^{\mathcal{F}})$. A gets its response z from $\mathcal{O}_1$. And A hands $(M_2, z \oplus v)$ over to $\mathcal{O}_4$. Then, A gets its response h from $\mathcal{O}_4$. A makes a new query $(M_2', h \oplus M_2 \oplus M_2')$ to $\mathcal{O}_3$. Then, A gets its response c from $\mathcal{O}_3$. Finally, A hands $(M_1||M_2')$ over to $\mathcal{O}_1$ and gets its response z'. If $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \mathcal{O}_4)$ is $(F^{P,E}, P, E, E^{-1})$, $c \oplus v = z'$. On the other hand, since any simulator cannot know M_1, $c \oplus v \neq z'$ with high probability. Therefore, $F^{P,E}$ based on PGV 15 is not indifferentiable from a VIL random oracle $\mathcal{F}$. In the similar way, cases of PGV 11, 13, and 16 are not secure.

Example 3. See PGV 17 in Table 1, which is $E_{h_{i-1}}(m_i) \oplus m_i$. Firstly, we define a hash function $H^P(x) : \{0,1\}^* \to \{0,1\}^n$ as follows, where c is any n-bit constant and P is a VIL random oracle with n-bit output size.

$$H^P(x) = c \qquad \text{if } x = c$$
$$= P(x) \qquad \text{otherwise,}$$

In the similar way with the proofs of Section 6, we can prove that H^P is preimage aware, preimage resistant, $(q, q_H(= q+1), 1/2^n)$-computable message aware, where q_H is the number of computable messages obtained from q input-output pairs of P. Now we want to give an indifferentiable attack on $F^{P,E}$ based on PGV 17. We construct an indifferentiability adversary A for $F^{P,E}(M) = OT(H^P(M_1), M_2)$, where $OT(x, y) = E_x(y) \oplus y$ is PGV 17. First, A chooses a query $M = c||M_2$ to $\mathcal{O}_1$, where M_2 is a randomly chosen one-block message. A gets its response z from $\mathcal{O}_1$. And A hands $(c, z \oplus M_2)$ over to $\mathcal{O}_4$. Then, A gets its response m from $\mathcal{O}_4$. If $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \mathcal{O}_4)$ is $(F^{P,E}, P, E, E^{-1})$, $m = M_2$. On the other hand, since any simulator cannot know M_2, $m \neq M_2$ with high probability. Therefore, $F^{P,E}$ based on PGV 17 is not indifferentiable from a VIL random oracle $\mathcal{F}$. In the similar way, cases of PGV 18-20 are not secure.

Case	PGV	Case	PGV
1	$E_{m_i}(h_{i-1}) \oplus h_{i-1}$	11	$E_{m_i}(h_{i-1}) \oplus v$
2	$E_{m_i}(w_i) \oplus w_i$	12	$E_{w_i}(h_{i-1}) \oplus v$
3	$E_{m_i}(h_{i-1}) \oplus w_i$	13	$E_{m_i}(h_{i-1}) \oplus m_i$
4	$E_{m_i}(w_i) \oplus h_{i-1}$	14	$E_{w_i}(h_{i-1}) \oplus w_i$
5	$E_{w_i}(m_i) \oplus m_i$	15	$E_{m_i}(w_i) \oplus v$
6	$E_{w_i}(h_{i-1}) \oplus h_{i-1}$	16	$E_{m_i}(w_i) \oplus m_i$
7	$E_{w_i}(m_i) \oplus h_{i-1}$	17	$E_{h_{i-1}}(m_i) \oplus m_i$
8	$E_{w_i}(h_{i-1}) \oplus m_i$	18	$E_{h_{i-1}}(w_i) \oplus w_i$
9	$E_{w_i}(m_i) \oplus v$	19	$E_{h_{i-1}}(m_i) \oplus w_i$
10	$E_{w_i}(m_i) \oplus w_i$	20	$E_{h_{i-1}}(w_i) \oplus m_i$

Table 1. 20 Collision Resistant PGV Hash Functions in the Ideal Cipher Model [4]. ($w_i = m_i \oplus h_{i-1}$)

6 PRO Attacks on Hash Functions with Some DBL Output Transformations

In this section, we consider DBL (Double Block Length) output transformations. Unfortunately, many constructions with DBL output transformations are not indifferentiably secure, even though H^P satisfies all requirements as mentioned before.

6.1 The case of $OT(x) = f(x)||f(x \oplus p)$

There are several DBL compression functions of the form $OT(x) = f(x)||f(x \oplus p)$ [18, 12] where p is a non-zero constant and f is any function. See Fig. 3, where F_1 was proposed by Nandi in [18] and $F_2 \sim F_7$ were proposed by Hirose in [12]. In fact, the T in F_1 Fig. 3 is a permutation without any fixed point and $T^2 = id$. Here, we consider only $T(x) = x \oplus p$, where p is a non-zero constant. We define a hash function $H^P(x) : \{0,1\}^* \to \{0,1\}^n$ as follows, where c is any n-bit constant and P is a VIL random oracle with n-bit output size. 1^n and 0^n indicate the n-bit one and zero strings.

$$H^P(x) = c \oplus p \qquad \text{if } x = 0^n$$
$$= c \qquad \text{if } x = 1^n$$
$$= P(x) \qquad \text{otherwise,}$$

Theorem 4. *Let H^P be the above hash function. For any preimage awareness (q, e, t)-adversary A making at most q queries to the oracles P, there exists an extractor $\mathcal{E}$ such that*

$$\mathrm{Adv}^{\mathrm{pra}}_{H^P,P,\mathcal{E}}(A) \leq \frac{eq}{2^n} + \frac{(q+2)^2}{2^{n+1}}, \text{ and } q_H \leq q+2.$$

Proof. We use Lemma 3.3 and Lemma 3.4 in [8]. First, we show that there exists a multi-point extractor $\mathcal{E}^+$ such that for any weak preimage awareness $(q, 1, t)$-adversary B, $\mathrm{Adv}^{1-\mathrm{wpra}}_{H^P,P,\mathcal{E}^+}(B) \leq \frac{q}{2^n}$. Second, we show that for any collision-finding adversary C making the same number of P queries as that of P queries of A, $\mathrm{Adv}^{\mathrm{cr}}_{H^P,P}(C) \leq \frac{(q+2)^2}{2^{n+1}}$. Then, by applying Lemma 3.3 and Lemma 3.4 in [8], the theorem holds. α_i is the advice string at the time when $A^{P,P^{-1},Q,Q^{-1}}_{\mathrm{guess}}$ commits y_i and $\mathcal{X} = \emptyset$.

algorithm $\mathcal{E}^+(y_i, \alpha_i)$:

For all M's such that H^P is computable from α_i,
if $H^P(M) = y_i$, then $\mathcal{X} = \mathcal{X} \cup \{M\}$
Return $\mathcal{X}$

Note that B is any weak preimage awareness $(q, 1, t)$-adversary. So, once B^P_{guess} commits y, B wins the weak preimage awareness game if B finds a new preimage image M' such that $H^P(M') = y$ and $M' \notin \mathcal{X}$. Since P is a VIL random oracle and B can make at most q queries to P, the probability that B wins is at most $\frac{q}{2^n}$.

Next, we consider the advantage of collision-resistance of H^P. Since C can make at most q queries to P, and $H^P(0)$ and $H^P(1)$ are computed without any query, there exist at most $q+2$ M's such that $H^P(M)$ is computable. And the probability that $H^P(M) = H^P(M')$ for $M \neq M'$ is 2^{-n}. Therefore, the probability that there exists a collision of H^P is at most $\frac{(q+2)^2}{2^{n+1}}$.

Finally, we will show that $q_H \leq q+2$. Since $H^P(0)$ and $H^P(1)$ are computable without any query to P and the maximum number of queries to the oracle P is q, the number q_H of messages M's such that $H^P(M)$ is computable from all query-response pairs to P is at most $q+2$. ∎

Theorem 5. *Let H^P be the above hash function. Let q be the maximum number of queries to P. For any preimage-finding adversary A with q queries to P, $\mathbf{Adv}^{\mathrm{PI}}_{H^P}(A) \leq \frac{3+q}{2^n}$. For any n-bit y and any M not computable from any advise string α which consists of q query-response pairs of P, $\Pr[H^P(M) = y|\alpha] \leq 1/2^n$.*

Proof. The preimage-finding adversary A will receive a random n-nit value h. If h is one of $c \oplus p$ and c, A can find a preimage of it very easily, because 0^n is a preimage for the output $c \oplus p$, and 1^n is a preimage for the output c. If h is not $c \oplus p$ or c, A can access to the oracle P until he gets its preimage. Note that P is a VIL random oracle and q is the maximum number of queries to P. So, the probability that A finds a preimage from responses of queries to P is $q/2^n$. If A doesn't find any preimage for h from the query-response pairs of P, A can just output any M^* expecting that M^* is a preimage for h, which occurs with probability $1/2^n$. Therefore, we can get the following relation: $\mathbf{Adv}^{\mathrm{PI}}_{H^P}(A) = \Pr[H^P(M^*) = h : M^* \leftarrow A^P(h); h \leftarrow \{0,1\}^n] \leq \Pr[h = c \oplus p] + \Pr[h = p] + \Pr[H^P(M^*) = h : M^* \leftarrow A^P(h); h \leftarrow \{0,1\}^n | (h \neq c \oplus p) \wedge (h \neq c)] \leq \frac{3+q}{2^n}$.

Note that a non-computable message M should not be 0^n or 1^n, because outputs for them are already defined as $c \oplus p$ and c respectively. So, the non-computable message M should be used as the input of the VIL random oracle P and be different from any query to P, which means that $\Pr[H^P(M) = y|\alpha] \leq 1/2^n$ by the property of the random oracle. ∎

Indifferentiability Attack on $F(M) = OT(H^P(M))$, where $OT(x) = f(x)||f(x \oplus p)$. Let $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3)$ be $(F^{P,f}, P, f)$ or $(\mathcal{F}, S_1^{\mathcal{F}}, S_1^{\mathcal{F}})$ for any simulator S. Now we define an adversary A as follows. First,

A makes query '0' and '1' to $\mathcal{O}_1$. Then, A obtains responses $(a_1||a_2)$ and $(b_1||b_2)$. If $\mathcal{O}_1 = F$, then $a_1 = b_2$ and $a_2 = b_1$. But, if $\mathcal{O}_1 = \mathcal{F}$, $a_1 = b_2$ and $a_2 = b_1$ with the probability $1/2^n$. So, F is not indifferentiably secure.

6.2 PRO Attack on the cases of $OT(x) = F_i(x)$ for $i = 8, 12$, (Fig. 3)

In the case of F_8 proposed by Lai and Massey in [14], which is called Tandem DM, there is the following structural weakness. If for any a $g_{i-1} = h_{i-1} = M_i = a$ in F_8 of Fig. 3, then $h_i \oplus g_i = a$. We can show an indifferentiability attack on $F(M) = F_9(H^P(M))$, where H^P is preimage aware and q_H is small. We define a hash function $H^P(x) : \{0,1\}^* \to \{0,1\}^n$ as follows, where c is any constant and P is a VIL random oracle with n-bit output size.

$$\begin{aligned} H^P(x) &= c||c && \text{if } x = 0, \text{ where } c \text{ is a } n/2\text{-bit constant} \\ &= P(x) && \text{otherwise,} \end{aligned}$$

And we can easily show that H^P is preimage aware, preimage resistant and $(q_P, q_H(= q_P+1), 1/2^n)$-computable message aware, where q_H is the number of computable messages obtained from q_P input-output pairs of P. and q_H is small.

Then, we show that $F(M_1||M_2) = F_8(H^Q(M_1), M_2)$ is not indifferentiable from a VIL random oracle $\mathcal{F}$ as follows. A makes a query '$(0||c)$' to $\mathcal{O}_1$ and get its response $z = (z_1||z_2)$. A checks if $z_1 \oplus z_2 = c$. If $\mathcal{O}_1$ is $\mathcal{F}$, $z_1 \oplus z_2 = c$ with the probability $1/2^{n/2}$. On the other hand, if $\mathcal{O}_1$ is F, $z_1 \oplus z_2 = c$ with probability 1. So F is not indifferentiably secure. In the case of F_{12}, which is called MDC-2, if the values of M_i and h_{i-1} are fixed, the half of bits of the output of F_{12} is also fixed regardless of what g_{i-1} is. Using this weakness, in a similar way as shown in above, we also can construct H^P such that $F(M_1||M_2) = F_{12}(H^P(M_1), M_2)$ is not indifferentiably secure.

7 Conclusion

In this paper we extend the applicability of preimage-awareness in those hash functions whose output transformation cannot be modeled as a random oracle. We choose Davis-Meyer as an output transformation based on a random permutation and show that the hash function is PRO if H^P is PrA, preimage resistant and computable message aware. The computable message awareness is a new notion introduced here similar to PrA. However this is not same as PrA as we can see the separation among these notions. As an application to our result we prove the PRO property of a variant of Grøstl hash function. We similarly prove that 12 PGV compression function out of 20 collision resistant PGV hash functions can be employed as output transformation with the similar assumption on H^P. However, some the popular double length hash function can not be used as we have shown PRO attacks. In summary, we study the choice of output transformation beyond the random oracle model and found both positive and negative results.

References

1. Elena Andreeva, Bart Mennink and Bart Preneel, On the Indifferentiability of the Grøstl Hash Function, Security and Cryptography for Networks, Lecture Notes in Computer Science, vol 6280, 2010, pp 88-105.
2. M. Bellare, T. Kohno, S. Lucks, N. Ferguson, B. Schneier, D. Whiting, J. Callas, J. Walker, *Provable Security Support for the Skein Hash Family*, http://www.skein-hash.info/sites/default/files/skein-proofs.pdf.
3. M. Bellare and P. Rogaway, *The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs*, Advances in Cryptology – EUROCRYPT'06, LNCS 4004, Springer-Verlag, pp. 409-426, 2006.

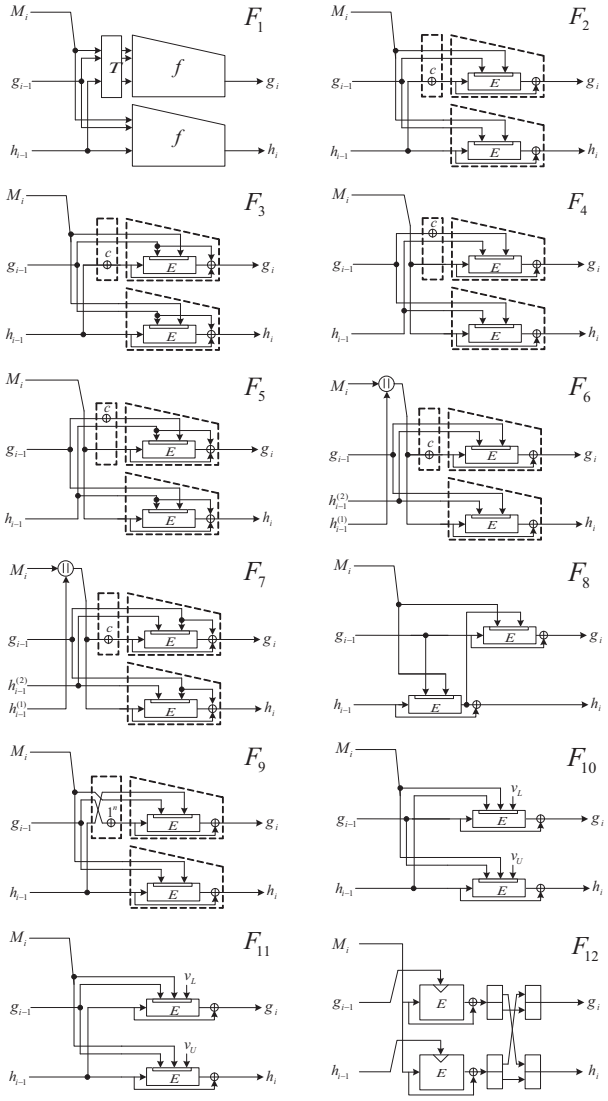

Fig. 3. Double Block Length constructions : F_i=DBL$_i$ for $1 \leq i \leq 12$

4. J. Black, P. Rogaway and T. Shrimpton, *Black-box analysis of the block-cipher-based hash function constructions from PGV*, Advances in Cryptology – CRYPTO'02, LNCS 2442, Springer-Verlag, pp. 320-335, 2002.
5. 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, Springer-Verlag, pp. 430-448, 2005.
6. I. B. Damgard, *A design principle for hash functions*, Advances in Cryptology-CRYPTO'89, LNCS 435, Springer-Verlag, pp. 416-427, 1990.
7. Y. Dodis, T. Ristenpart and T. Shrimpton, *Salvaging Merkle-Damgård for Practical Applications*, Advances in Cryptology – EUROCRYPT'09, LNCS 5479, Springer-Verlag, pp. 371-388, 2009.
8. Y. Dodis, T. Ristenpart and T. Shrimpton, *Salvaging Merkle-Damgård for Practical Applications*, full version of [6], Cryptology ePrint Archive: Report 2009/177.
9. N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas and J. Walker, *The Skein Hash Function Family*, Submission to NIST, 2008.
10. P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, Søren S. Thomsen, *Grøstl - a SHA-3 candidate*, Submission to NIST, 2008.
11. S. Hirose, *Secure Double-Block-Length Hash Functions in a Black-Box Model*, ICISC 2004, LNCS 3506, Springer-Verlag, pp. 330-342, 2005.

12. S. Hirose, *How to Construct Double-Block-Length Hash Functions*, In second Hash Workshop, 2006.
13. J. Kelsey, *Some notes on Grøstl*, http://ehash.iaik.tugraz.at/uploads/d/d0/Grostl-comment-april28.pdf, 2009.
14. X. Lai and J. L. Massey, *Hash Functions Based on Block Ciphers*, Advances in Cryptology – EUROCRYPT'92, LNCS 658, Springer-Verlag, pp. 55-70, 1993.
15. Ueli Maurer, *Indistinguishability of Random Systems*, Advances in Cryptology – EUROCRYPT02, volume 2332 LNCS, Springer-Verlag, pp 110 - 132, 2002.
16. U. Maurer, R. Renner and C. Holenstein, *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*, TCC'04, LNCS 2951, Springer-Verlag, pp. 21-39, 2004.
17. R. C. Merkle, *One way hash functions and DES*, Advances in Cryptology-CRYPTO'89, LNCS 435, Springer-Verlag, pp. 428-446, 1990.
18. M. Nandi, *Towards Optimal Double-Length Hash Functions*, INDOCRYPT'05, LNCS 3797, Springer-Verlag, pp. 77-89, 2005.
19. B. Preneel, R. Govaerts and J. Vandewalle, *Hash Functions based on Block Ciphers : A Synthetic approach*, Advances in Cryptology-CRYPTO'93, LNCS 773, Springer-Verlag, pp. 368-378, 1994.
20. R. Winternitz, *A Secure Hash Function built from DES*, In Proceedings of the IEEE Symp. on Information Security and Privacy, pp. 88-90. IEEE Press. 1984.

Appendix A. Revisiting the Proof of "RO(PrA(·)) = PRO(·)"

In [7, 8] it was proved that $F^{R,P}(M) = \mathcal{R}(H^P(M))$ is indifferentiable from a VIL random oracle $\mathcal{F}$, where $\mathcal{R} : \{0,1\}^m \to \{0,1\}^n$ is a FIL random oracle, P is an ideal primitive, and $H^P : \mathcal{M} \to \{0,1\}^m$ is preimage-aware. The result can be used to prove the indifferentiable security of any hash function which uses a post-processor defined independently from the underlying iteration function P. In the course of our studies we have found that the proof given in [7, 8] is not completely correct (though the claims remain correct). We have reported this in a limited distribution abstracts on February and May 2010 (recently in October 2010, a correction on the e-print version has appeared by the original coauthors, further confirming our findings). Let us review the issues. There are two main flaws (to be described below) in the proof, and we need to provide alternative definitions of simulator and preimage-aware attacker to fix them. (We note that while somewhat technical, the revision is crucial). Let $\mathsf{NQ}[l]$ be the number of P-queries required for the computation of $H^P(M)$ for $|M| = l$. We denote $\mathsf{Time}(\cdot)$ and $\mathsf{STime}(\cdot)$ to mean the run time and simulation run time of an algorithm. Now we restate the Theorem 4.1. in [7, 8] (in terms of our PrA terminologies) and provide a sketch of the proof given in [7].

Theorem 4.1 of [7, 8]
For any given efficient extractor $\mathcal{E}$, there exists a simulator $S = (S_1, S_2)$ with $\mathsf{Time}(S) = O(q_1 \cdot \mathsf{STime}(P) + q_2 \cdot \mathsf{Time}(\mathcal{E}))$. The simulator makes at most q_2 $\mathcal{F}$-queries. For any indifferentiability adversary $A^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2}$ making at most (q_0, q_1, q_2) queries to its three oracles with bit-size l_{max} for the longest $\mathcal{O}_0$-query, there exists a $(q_1 + q_0 \cdot \mathsf{NQ}[l_{max}], q_2 + 1, t)$-PrA adversary B_A^P with runtime $t = \mathsf{Time}(A) + O(q_0 \cdot \mathsf{NQ}[l_{max}] + q_1 + q_2\mathsf{Time}(\mathcal{E}))$ such that

$$\mathbf{Adv}^{\mathrm{pro}}_{F,S}(A) \leq \mathbf{Adv}^{\mathrm{pra}}_{H^P,P,\mathcal{E}}(B_A).$$

Outline of Proof of Theorem 4.1. of [8]. Let $\mathcal{E}$ be an arbitrary extractor for H. Then $S = (S_1; S_2)$ works as follows. It maintains an internal advice string α (initially empty) that will consist of pairs $(u; v)$ corresponding to A's queries to P (via S_1). When A queries u to S_1, the simulator simulates $v \leftarrow P(u)$ appropriately, sets $\alpha \leftarrow \alpha \| (u; v)$, and returns v. For a query Y to S_2, the simulator computes $X \leftarrow \mathcal{E}(Y; \alpha)$. If $X = \perp$ then the simulator returns a random point. Otherwise it simulates $Z \leftarrow \mathcal{F}(X)$ and returns Z to the adversary. The games $R0, I1, G0, G1$ and B_A have been defined in [8] and the authors claimed the following:

$$(1)\ \ G1 \equiv I1 \equiv (\mathcal{F}, S_1, S_2), \quad (2)\ \ G0 \equiv R0 \equiv (F^{P,\mathcal{R}}, P, \mathcal{R}).$$

Due to the above claim the PRO-advantage of any adversary A is nothing but $|\Pr[A^{G0}=1]-\Pr[A^{G1}=1]$. From the pseudocodes of games $G0$ and $G1$, it is easy to see that they are identical-until-Bad. Hence $\mathbf{Adv}^{\mathrm{pro}}_{F,S}(A) \leq \Pr[A^{G1}$ sets Bad true$]$. The proof proceeds by defining a PRA-adversary B_A which makes preimage-aware attack successfully whenever B_A sets Bad true. Since B_A sets Bad true only if it finds a collision of H^P or finds a message M such that $\mathcal{E}(\alpha, Y) \neq M$ where $Y = H^P(M)$. So $\Pr[B_A$ sets Bad true$] \leq \mathbf{Adv}^{\mathrm{pra}}_{H^P,P,\mathcal{E}}(B_A)$. The theorem follows immediately from the following claim:

(3) $\Pr[A^{G1}$ sets Bad true$] \leq \Pr[B_A$ sets Bad true$]$. ■

7.1 Problems in Proof of Theorem 4.1. of [8]

In this section we explain the flaws we observe in the proof of Theorem 4.1. of [8]. To understand it one needs to go through the definitions of the games $G0$, $R0$, $G1$ and $G(B)$ (the tuple of three oracles simulated by B) described in [8].[2]

Flaw 1. $G0$ **is not equivalent to** $R0$.
If $\mathcal{O}_0$ in $G0$ has not been queried before (so the Bad event in $G0$ would not occur) then the output of $\mathcal{O}_2(Y)$ query is $\mathsf{F}(X)$ whenever $X = \mathcal{E}(Y, \alpha) \neq \perp$, otherwise it returns $\mathsf{R}(X)$ where F and R perfectly simulate two independent random oracles $\mathcal{F}$ and $\mathcal{R}$ respectively. We show that $\mathcal{O}_2$ cannot be equivalent to a random oracle. Suppose $\mathcal{E}$ is an extractor which returns a special message M^* whenever the advise string α is empty. If A makes first two successive distinct $\mathcal{O}_2$ queries $Y_{2,1}$ and $Y_{2,2}$ then $X_{2,1} = X_{2,2} = M^*$ and hence the outputs of $\mathcal{O}_2$ in game $G0$ are identical (same as $\mathsf{F}[M^*]$).

To get rid of the above problem, we can do the following steps in $\mathcal{O}_2$ (also in simulator S_2) immediately after it obtains $X = \mathcal{E}(Y, \alpha)$: Compute $H^P(X) = Y'$ and check whether it is the same as Y or not. If extractor returns a correct message, i.e. $Y = Y'$, then S_2 or $\mathcal{O}_2$ returns $\mathcal{F}(M)$. Otherwise, it returns randomly. To compute $H^P(X)$ one might need to simulate some P outputs (in case of $G0$) or make P-queries (in case of $\mathcal{O}_2$ of B).

Flaw 2. $G(B) \not\equiv G1$ **and** $G1, G(B)$ **are not identical-until-Bad.**
We first observe that the advise string α in $G1$ is not the same as that of B since the advice string α is updated whenever A has access to the oracle $\mathcal{O}_1$ in Game $G1$, but the advice string is updated whenever A has access to the oracle $\mathcal{O}_0$ and $\mathcal{O}_1$ of B in Fig 3 of [8]. For example, let $\mathcal{E}(Y, \alpha)$ return a message M whenever $H^P(M) = Y$ is "computable' from α otherwise return $\perp$. Any adversary which can guess $H^P(M)$ correctly and turn it to $\mathcal{O}_2$ query then $\mathcal{O}_2^B(Y|\tau)$ returns z. However, $\mathcal{O}_2^{G1}(Y|\tau)$ returns a random string $\mathsf{R}[Y]$ since α is the empty string in A^{G1}. So $G(B) \not\equiv G1$. One can similarly show that $G1, G(B)$ are not identical-until-Bad.

A possible attempt is to update the advise string for $\mathcal{O}_0$ queries in all games, in particular $G1$. However, if we do so then the simulator is not independent of $\mathcal{F}$-queries (since the advise string is updated whenever there is a $\mathcal{O}_0$-query and the advise string is used to define the response of S_2). On the other hand, we cannot ignore the $H^P(M)$ computation in B for $\mathcal{O}_0$ queries of A. This computation is essential to making PrA attack successfully. It seems impossible to handle the advise string so that it is updated in the same way for all games as well as $H^P(\cdot)$ computations are made for $\mathcal{O}_0$-queries. We can solve the problem if **we postpone the computation of** H^P **until all queries of** A **are made**. So we need a finalization procedure in B which essentially does all $H^P(M)$ computations of $\mathcal{O}_0(M)$-queries.

7.2 Revised Proof of Theorem 4.1 of [8]

We state the corrected version of theorem 4.1. below. The revised version of $B := B_A$, simulators and the games $G0$ and $G1$ are defined in Fig. 1. The adversary B_A has a subroutine called Finish() which is

[2] We have defined the revised version of these games in the paper. We refer readers to [8] to see the original definitions to understand the flaws.

defined trivially. It mainly completes the PrA attack. It is easy to see that whenever $\mathsf{Finish}()$ is being executed either we have a collision in H^P or there is some message M such that $H^P(M) = y, (y, M) \notin \mathsf{Ext}$. For simplicity we ignore the details of the subroutine. Let $q = q_1 + (q_0 + q_2) \cdot \mathsf{NQ}[l_{max}]$.

Game $\boxed{G0}$ and $G1$	Adversary B_A^P and Simulator $S^{\mathcal{F}} = (S_1, S_2)$
Initialize : $\mathsf{H} = \mathsf{R}_2 = \mathsf{R}_0 = \phi$; $\mathcal{L} = \beta = \phi$, $i = 1$, Bad =F;	Initialize : $\mathsf{H} = \mathsf{R}_2 = \mathsf{R}_0 = \mathcal{L} = \beta = \phi$; $i = 1$, Bad =F; Run A and respond queries of A's as follows:
200 On $\mathcal{O}_2$ - query $y := y_i$, $i = i + 1$	200 On $\mathcal{O}_2$ (or S_2)-query $y := y_i$, $i = i + 1$
201 $X = \mathcal{E}(y_i, \beta)$; $\mathsf{Ext} \xleftarrow{\cup} (y, X)$;	201 $X = \mathcal{E}(y_i, \beta)$; $\mathsf{Ext} \xleftarrow{\cup} (y, X)$;
202 $y' = H^P(X)$ and update β;	202 $y' = H^P(X)$ and update β;
203 If $y' = y$	203 If $y' = y$
204 then $z = \mathcal{R}(y)$;	204 then $z = \mathcal{R}(y)$;
205 If $y' = y \wedge (M, y) \in \mathsf{H}$	
206 then Bad =T; $\boxed{z = \mathsf{R}_0[y];}$	
207 If $y' = y$	207 If $y' = y$
208 then $z = \mathcal{F}(X)$;	208 then $z = \mathcal{F}(X)$;
209 If $y' = y \wedge (M, y) \in \mathsf{H} \wedge M = X$	
210 then Bad =T; $\boxed{z = \mathsf{R}_0[y];}$	
211 $\mathsf{R}_2 \xleftarrow{\cup} (y, z)$; return z;	211 $\mathsf{R}_2 \xleftarrow{\cup} (y, z)$; return z;
100 On $\mathcal{O}_1$ - query u	100 On $\mathcal{O}_1$ (or S_1)-query u
101 $v = P(u)$; $\beta \xleftarrow{\parallel} (u, v)$;	101 $v = P(u)$; $\beta \xleftarrow{\parallel} (u, v)$;
102 return v;	102 return v;
000 On $\mathcal{O}_0$ - query M	000 On $\mathcal{O}_0$ (or $\mathcal{F}$)- query M
001 $z = \mathcal{F}(M)$; $\mathcal{L} \xleftarrow{\cup} M$;	001 $z = \mathcal{F}(M)$; $\mathcal{L} \xleftarrow{\cup} M$;
002 $y = H^P(M)$; $\mathsf{H} \xleftarrow{\cup} (M, y)$;	002 $\mathsf{R}_0 \xleftarrow{\cup} (y, z)$; return z;
003 If $\mathsf{R}_0[y] = \perp$	400 **Finalization:** (after A finishes queries.)
004 then Bad = T; $\boxed{z = \mathsf{R}_0[y];}$	401 If $\exists M = M' \in \mathcal{L}$, $H^P(M) = H^P(M')$
005 Else if $\mathsf{R}_2[y] = \perp \wedge (y, M) \in \mathsf{E}$	402 then bad =T, $\mathsf{Finish}()$;
006 then Bad = T; $\boxed{z = \mathsf{R}_2[y];}$	403 If $\exists M \in \mathcal{L}$, $(y, X) \in \mathsf{E}$, $X = M$, $H^P(M) = y$
007 $\mathsf{R}_0 \xleftarrow{\cup} (y, z)$; return z;	404 then bad =T, $\mathsf{Finish}()$;
	405 return $\perp$;

Fig. 4. $G0$ executes with boxed statements whereas $G1$ executes without these. Clearly $G0$ and $G1$ are identical-until-Bad and whenever $G1$ set bad true the adversary B_A^P set also bad true. In this case, $\mathsf{Finish}()$ subroutine executes which makes PrA successful. The tuple of simulated oracles of B_A is equivalent to $(\mathcal{F}, S_1, S_2)$.

Lemma 9. *$G1 \equiv (\mathcal{O}_0^{B_A}, \mathcal{O}_1^{B_A}, \mathcal{O}_2^{B_A}) \equiv (\mathcal{F}, S_1, S_2)$. Games $G0$ and $G1$ are identical-until-Bad.*

The Lemma is obvious from the games described in Fig. 1. We leave readers to verify. The following lemma essentially says that $G0$ is equivalent to $(\mathcal{R}(H^P), P, \mathcal{R})$. The proof of the lemma is easy to verify and we skip it for the full version.

Lemma 10. *$G0 \equiv (\mathcal{R}(H^P), P, \mathcal{R})$, i.e. for any distinguisher A, the output distribution of A^{G0} and $A^{\mathcal{R}(H^P), P, \mathcal{R}}$ are identically distributed.*

Lemma 11. *Whenever A^{G1} sets Bad true, B_A sets Bad true and B_A makes PrA attack successful. So we have $\Pr[A^{G1}$ sets $\mathsf{bad}] \leq \Pr[B_A$ sets $\mathsf{bad}] \leq \mathbf{Adv}^{\mathrm{pra}}_{H^P, P, \mathcal{E}}(B_A)$.*

Proof. We already know from Lemma 9 that $G1$ is equivalent to the oracles simulated by B_A. However, the two games defined bad event in different manners. The game $G1$ sets bad during the computation

of responses whereas the adversary B_A sets bad after all responses of the queries. A^{G1} sets bad true in line 209, 003 and in line 206, 005. We can see that if the conditional statements written in 209 and 003 in game $G1$ hold then we have a collision in H^P (there exist $M \neq X$ such that $H^P(M) = H^P(X)$). So we have PrA attack which is taken care of in 401 in the second step of B_A. For the lines 205 and 005 we have M such that $H^P(M) = y$ and $\mathsf{Ext}[y] \neq M$, i which case PrA attack is possible due to incorrect guess of the extractor. This has been taken care of in 403. ■

By using the above lemmas the theorem follows immediately

Theorem 6 (Ro domain extension via PrA). *For any given extractor $\mathcal{E}$ we can construct a simulator $S = (S_1, S_2)$ with $\mathsf{Time}(S) = O((q_1 + q_2 \cdot \mathsf{NQ}[l_{max}]) \cdot \mathsf{STime}(P) + q_2 \cdot \mathsf{Time}(\mathcal{E}))$. For any indifferentiability adversary $A^{\mathcal{O}_0,\mathcal{O}_1,\mathcal{O}_2}$ making at most (q_0, q_1, q_2) queries to its three oracles with bit-size l_{max} for the longest $\mathcal{O}_0$-query, there exists a (q, q_2+1, t)-adversary B with runtime $t = \mathsf{Time}(A) + O(q_2 \cdot \mathsf{Time}(\mathcal{E}) + q_0 + q_1 + (q_2 + q_0)\mathsf{NQ}[l_{max}])$ and*

$$\mathbf{Adv}^{\mathrm{pro}}_{F,S}(A) \leq \mathbf{Adv}^{\mathrm{pra}}_{H^P,P,\mathcal{E}}(B),$$