Side Channel Analysis of Some Hash Based MACs: A Response to SHA-3 Requirements

Praveen Gauravaram[1,⋆] and Katsuyuki Okeya[2]

[1] DTU Mathematics, Technical University of Denmark, Denmark
p.gauravaram@mat.dtu.dk
[2] Hitachi, Ltd., Systems Development Laboratory, Japan
katsuyuki.okeya.ue@hitachi.com

Abstract. The forthcoming NIST's Advanced Hash Standard (AHS) competition to select SHA-3 hash function requires that each candidate hash function submission must have at least one construction to support FIPS 198 HMAC application. As part of its evaluation, NIST is aiming to select either a candidate hash function which is more resistant to known side channel attacks (SCA) when plugged into HMAC, or that has an alternative MAC mode which is more resistant to known SCA than the other submitted alternatives. In response to this, we perform differential power analysis (DPA) on the possible smart card implementations of some of the recently proposed MAC alternatives to NMAC (a fully analyzed variant of HMAC) and HMAC algorithms and NMAC/HMAC versions of some recently proposed hash and compression function modes. We show that the recently proposed BNMAC and KMDP MAC schemes are even weaker than NMAC/HMAC against the DPA attacks, whereas multi-lane NMAC, EMD MAC and the keyed wide-pipe hash have similar security to NMAC against the DPA attacks. Our DPA attacks do not work on the NMAC setting of MDC-2, Grindahl and MAME compression functions.

Keywords: Applied cryptography, hash functions, side channel attacks, HMAC.

1 Introduction

The cryptanalysis of the MD5 and SHA-1 hash functions [41, 42] and its impact on several applications [5, 10, 13, 16, 40] have triggered a kind of feeding frenzy among the cryptographers. On the other hand, generic attacks [20, 23] on the popular Merkle-Damgård (MD) hash framework [12, 30] have exposed several of its undesirable properties.

In the wake of this active cryptanalysis of hash functions and its impact on applications, NIST is conducting an international competition to define an Advanced Hash Standard (AHS) which would be referred to as SHA-3 family [34].

⋆ Author and this research project are supported by The Danish Research Council for Technology and Innovation grant number 274-08-0052.

NIST requires that each candidate hash function must have at least one construction to support the current applications of hash functions specified in the FIPS or NIST special publications that include FIPS 198 HMAC [33]. As part of its evaluation, NIST is also considering side channel attacks (SCA) on the hash based MACs. NIST intends to select as SHA-3, either a candidate hash which is more resistant to known SCA attacks when plugged into HMAC, or that has an alternative MAC mode which is more resistant to known SCA attacks than the other submitted alternatives [9, 21, 22].

Considering this state of the art of research in hash functions, we believe that AHS competition would receive hash as well as compression function modes as candidates for SHA-3 using structures and chaining modes different from the ones used in the broken hash functions. It is prominent that such proposals define provably secure MAC modes with a protection from the SCA attacks.

This research problem has motivated us to assess the security of several recently proposed MAC alternatives to NMAC (a thoroughly analysed variant of HMAC) and HMAC algorithms [3, 2] and some compression function and hash function modes in the NMAC/HMAC setting from differential power analysis (DPA) attacks. We analyse MACs that are assumed to be instantiated with the compression functions built over ideal block ciphers that are secure against SCA attacks as was done in [36, 17]. If the proposed MAC or hash function mode does not specify any block cipher based compression function then we analyse it using twelve secure compression functions based on block ciphers proposed by Preneel, Govaerts and Vandewalle (PGV) [38]. Such analysis allows the designers to construct hash and DPA resistant MAC modes whose security can be formally reduced to the compression function modes that are real and whose security was formally established [7]. In a related work, HMAC based on the dedicated hash functions SHA-1 and SHA-256 was shown to be vulnerable to the side channel attacks [28, 26].

1.1 Our Approach

We analyse several recently proposed provably secure MAC alternatives to the NMAC/HMAC algorithms and NMAC/HMAC settings of some compression function and hash function prototypes proposed in the literature against the DPA attacks. Although the list of MAC algorithms that we have analysed may not be thorough, our analysis can be easily extended to the other similar MAC proposals that we might have missed. The outcome of the DPA attacks on many of these MAC schemes is the recovery of their secret keys. The MAC schemes that are vulnerable to the complete key recovery attacks can be *universally forged*; forgery for any given message. The MACs for which we can partially recover the key or internal state, we can either *existentially forge* the scheme by computing a valid authentication tag for a random message or cannot guarantee its security against forgery attacks. We perform DPA analysis of MACs by dividing them into *Type-1* and *Type-2* categories:

Type-1: Provably secure MAC alternatives to NMAC/HMAC. These MAC schemes, in general, are based on the alternative hash frameworks to the

MD structure. They include BNMAC and its single key variants [43], MAC based on Enveloped Merkle-Damgård (EMD) transform [4], keyed version of Merkle-Damgård with permutation (MDP)-KMDP [18], Multi-Lane NMAC [44] and One-keyed NMAC (O-NMAC) [14]. These schemes do not emphasize using any specific compression function; hence, we analyse their security using twelve secure PGV schemes.

Type-2: NMAC setting of the compression and hash function modes. This category includes the DPA analysis of the NMAC settings of MDC-2 [11,32], MAME [45] and Grindahl [24] compression functions which is also applicable to their HMAC versions. MDC-2 has been chosen for its rigorous analysis and specification in the standards ANSI X9.31 [1] and ISO/IEC 10118-2 [19] and the new proposals Grindahl and MAME are chosen due to their novelty. We assume that the keyed versions of these compression functions define a family of pseudorandom functions and hence their NMAC settings retain the proof of security of NMAC [2]. We also analyse the security of the wide-pipe hash [27] instantiated with twelve PGV schemes in the setting of NMAC.

Our DPA analysis of MACs based on EMD, MDP and wide-pipe hash [27] and the NMAC setting of the hashes in the *Type-2* category meet the criteria of the AHS evaluation process where a MAC version of a secure hash is expected to resist SCA attacks. EMD and KMD modes preserve the pseudorandom oracle and collision resistance properties of the compression functions whereas wide-pipe hash was shown to be secure against the generic attacks of [20, 23]. The unkeyed version of O-NMAC was recently shown to be weak against the generic attacks of [23,20] in [15] and collisions can be easily found for the unkeyed version of BNMAC as shown later in the paper.

1.2 Our Results and Their Significance

In Table 1, we outline the results of the DPA attacks on the MAC schemes in the *Type-1* category. While we have analysed MAC functions, where applicable, instantiated with twelve PGV schemes, here we outline our results for the MAC instantiations based on the popular compression functions that include Matyas-Meyer-Oseas (MMO), Miyaguchi-Preneel (MP) and Davies-Meyer (DM). In Table 1, the following notation holds: CK-complete key recovery, PK-partial key recovery, N/A-not applicable, NO-key recovery is not possible, EF-existential forgery, UF-universal forgery, NG-no guarantee on the MAC security. For the sake of comparison, we have also included the results of the DPA analysis of NMAC [36, 17] in the last row of Table 1. While MP scheme can be implemented in three different ways for a given block cipher as shown later in the paper, Table 1 outlines the results based on its strongest implementation. The NMAC settings of the compression functions in the *Type-2* category are not vulnerable to the DPA attacks. NMAC based on the wide-pipe hash has similar security as NMAC [36] with respect to the DPA attacks.

Our research indirectly provides some ground work on building DPA resistant hash based MACs using cryptographic primitives such as block ciphers that are often implemented as DPA resistant hardware modules. Our work provides the

Table 1. Our results on the *Type-1* MACs based on three popular PGV schemes

MAC function	Matyas-Meyer-Oseas	Miyaguchi-Preneel	Davies-Meyer
BNMAC [43]	PK(EF)	CK(UF)	CK(UF)
EMD [4]	N/A	N/A	PK(NG)
KMDP [18]	NO	NO	CK(UF)
Multi-lane NMAC [44]	N/A	N/A	PK(NG)
O-NMAC [25]	NO	NO	NO
NMAC [36]	NO	NO	PK(NG)

following significant contributions: Firstly, we analysed several alternatives to NMAC/HMAC that use hash function modes different from that of MD [36,17]. Secondly, we analysed MACs in the MD mode instantiated with the compression functions different from the PGV schemes. Finally, we show the first example of a MAC scheme, in the form of BNMAC, vulnerable to the DPA attacks and can be *existentially forged* when instantiated with any secure compression function.

Our results are the outcome of theoretical DPA attacks on the possible smart card implementations of some MAC proposals. To our knowledge, currently, no smart card implementations of the MAC schemes analysed in this paper are available. So, experimental verification of our analysis has not been done so far. However, since our DPA attacks follow the same attack model used to attack HMAC implementations on an IC chip in [36], it is possible to realise the practicality of our attacks on the MAC implementations on a real smart card system or its emulated system.

1.3 Guide to the Paper

In Section 2, we introduce hash functions and NMAC and HMAC functions. In Section 3, we generalise DPA attacks on MACs. In Sections 4 and 5, we analyse *Type-1* and *Type-2* MAC schemes against the DPA attacks. We conclude the paper in Section 6 with some open questions.

2 Hash Functions

A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ processes an arbitrary length message into a fixed length n-bit hash value. It is a common approach to design H by iterating a compression function $h : \{0,1\}^n \times \{0,1\}^b \rightarrow \{0,1\}^n$ which processes a fixed length b-bit message and an n-bit input state producing an n-bit output state. The message to be processed using H is always padded using any secure one-to-one padding technique such that $\{0,1\}^* \rightarrow \{0,1\}^{b.t}$ where t is the number of b-bit blocks. The padded message is represented with b-bit message blocks as $m = m[1]\| \ldots \|m[t]$. Each block $m[i]$ is processed using h to compute intermediate hash values $H[i] = h_{H[i-1]}(m[i])$ where $i = 1, \ldots, t$ and $H[0]$ is the fixed initial value (IV) of H. The final state $H[t] = h_{H[t-1]}(m[t])$ is the hash value of m.

The MD iterative structure [30, 12] has been a popular iterated hash function framework used in the design of standard hash functions such as SHA-1 and SHA-2 family [35]. Let e^c be the concatenation of e bit c times where e is either 0 or 1. MD hash functions specify an upper bound of 2^l bits on the length of m and always pad m by appending it with a 1 bit and $0^{b-l-d-1}$ where d is the number of message bits in the incomplete block of m. The last l bits of m are filled in with the binary encoded representation of the length of m in bits (depending on the size of d, an additional block may be used for padding).

Often, compression function modes are constructed using block ciphers. Twelve out of sixty-four PGV compression function modes [38] are provably secure when their underlying block cipher is ideal [7]. This model of PGV uses parameters $p, q, r \in \{H[i-1], m[i], H[i-1] \oplus m[i], 0\}$ and a block cipher G to derive a compression function. See Table 2 for the description of these 12 schemes denoted with h^j where j is from 1 to 12. Table 2 also includes three possible implementations of the compression functions h^3 and h^7. The subscript to G denotes its key input which is either a message block $m[i]$ or a hash state $H[i-1]$ when G is turned into the compression function h using one of the 12 PGV modes. In this paper, we shall often assume that $b = n$ for these twelve PGV schemes.

Table 2. 12 provably secure PGV compression functions

Compression function	Description
h^1	$H[i] = G_{H[i-1]}(m[i]) \oplus m[i]$
h^2	$H[i] = G_{H[i-1]}(m[i] \oplus H[i-1]) \oplus m[i] \oplus H[i-1]$
h^3	$H[i] = G_{H[i-1]}(m[i]) \oplus (m[i] \oplus H[i-1])$
h^4	$H[i] = G_{H[i-1]}(m[i] \oplus H[i-1]) \oplus m[i]$
h^5	$H[i] = G_{m[i]}(H[i-1]) \oplus H[i-1]$
h^6	$H[i] = G_{m[i]}(H[i-1] \oplus m[i]) \oplus (H[i-1] \oplus m[i])$
h^7	$H[i] = G_{m[i]}(H[i-1]) \oplus (m[i] \oplus H[i-1])$
h^8	$H[i] = G_{m[i]}(H[i-1] \oplus m[i]) \oplus H[i-1]$
h^9	$H[i] = G_{H[i-1] \oplus m[i]}(m[i]) \oplus m[i]$
h^{10}	$H[i] = G_{H[i-1] \oplus m[i]}(H[i-1]) \oplus H[i-1]$
h^{11}	$H[i] = G_{H[i-1] \oplus m[i]}(m[i]) \oplus H[i-1]$
h^{12}	$H[i] = G_{H[i-1] \oplus m[i]}(H[i-1]) \oplus m[i]$
$h^{(3,1)}$	$H[i] = (H[i-1] \oplus m[i]) \oplus G_{H[i-1]}(m[i])$
$h^{(7,1)}$	$H[i] = (H[i-1] \oplus m[i]) \oplus G_{m[i]}(H[i-1])$
$h^{(3,2)}$	$H[i] = (G_{H[i-1]}(m[i]) \oplus m[i]) \oplus H[i-1]$
$h^{(7,2)}$	$H[i] = (G_{m[i]}(H[i-1]) \oplus m[i]) \oplus H[i-1]$
$h^{(3,3)}$	$H[i] = (G_{H[i-1]}(m[i]) \oplus H[i-1]) \oplus m[i]$
$h^{(7,3)}$	$H[i] = (G_{m[i]}(H[i-1]) \oplus H[i-1]) \oplus m[i]$

2.1 NMAC and HMAC

Let k_2 and k_1 be any two random and independent secret keys. If H is an MD hash, the NMAC function [3, 2] is defined by $\mathrm{NMAC}_{k_1,k_2}(m) = h_{k_1}(H_{k_2}(m))$

where the keys k_2 and k_1 replace the IVs of the inner and outer H where outer H is expected to perform only one iteration. If k is an n-bit random secret key then $\mathrm{HMAC}_k(m) = H_{IV}((k\|0^{b-|k|} \oplus \texttt{const1})\|H_{IV}((k\|0^{b-|k|} \oplus \texttt{const2})\|m)))$. HMAC and NMAC are related by $\mathrm{HMAC}_k(m) = h_{k_1}(H_{k_2}(m))$ where $k_1 = h_{IV}((k\|0^{b-|k|}) \oplus \texttt{const1})$, $k_2 = h_{IV}((k\|0^{b-|k|}) \oplus \texttt{const2})$, `const1` and `const2` are the constants defined in [3] and $\|$ is the concatenation operation.

3 Side Channel Attacks on Hash Based MACs

Here, we generalize the DPA attacks [36, 17] mounted on the NMAC/HMAC algorithms instantiated with the 12 secure PGV compression functions based on the DPA resistant block ciphers.

3.1 Differential Power Analysis (DPA) Attack

The main objective of mounting a DPA attack on a MAC function is to detect target regions in the power consumption of a cryptographic device having a MAC function implementation correlated with particular bits of the secret key.

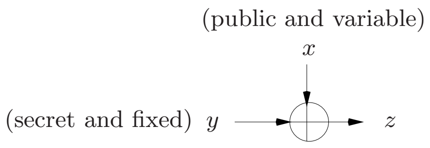

Fig. 1. The DPA attack model

Lemma 1. *A DPA attack is mounted on a MAC function with the target XOR operation $z = x \oplus y$ (See Figure 1) to detect the fixed secret key input y where x is a public variable input.*

The DPA attack on such a MAC function is outlined below:

1. Guess a certain bit b of the secret input y and run the MAC algorithm having the above target XOR operation for N random values of message input x_i where $i = 1 \ldots N$.
2. For each of the N message inputs x_i, a discrete time power signal S_{it} is collected and the corresponding output z_i of the XOR operation is also collected. The index i corresponds to the message input x_i that produced the signal and t corresponds to the time of the sample.
3. Let $x_{i,k}$ be the k^{th} bit of input x_i. Sort the inputs x_i depending on whether the target k^{th} bit of z is 0 or 1. Let $S_b = \{S_{it}|x_{i,k} \oplus b = 0\}$ and $S_{\overline{b}} = \{S_{it}|x_{i,k} \oplus b = 1\}$ where b is the guessed k^{th} bit of y.
4. Compute average power signal for each of the sets S_b and $S_{\overline{b}}$ where $|S_b| + |S_{\overline{b}}| = N$:

$$AP_b[t] = 1/|S_b| \sum_{S_{it} \in S_b} S_{it}$$

$$AP_{\overline{b}}[t] = 1/|S_{\overline{b}}| \sum_{S_{it} \in S_{\overline{b}}} S_{it}$$

5. Following the Hamming weight model of [31], the power consumed by the target XOR operation depends on the Hamming weight of the manipulated data. When there is a large power consumption, that is when $AP_{\overline{b}}[t] \gg AP_b[t]$, the target bit of z is 1 since the other bits behave randomly and the averaging eliminates their effect. This DPA bias signal is used to verify the guess of the secret key bit b of y.
6. By repeating the above steps, the whole secret key y can be recovered. However, it is enough to reclassify the input x and compute average power signal for the new sets using the power signal samples collected in the first instance.

Reverse DPA (RDPA) attack. It is a minor variant of DPA where instead of known input a known output is used.

Lemma 2. *An RDPA attack is mounted on a MAC function with the target XOR operation $z = x \oplus y$ (see Figure 2) to detect the fixed secret key input y where z is a public variable which may not be controlled.*

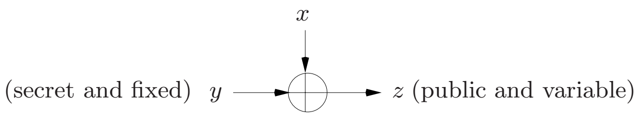

Fig. 2. The RDPA attack model

The RDPA attack on such a MAC function is outlined below:

1. Guess a certain bit of the secret input y and run the MAC algorithm for N random values of input x and collect the discrete time power signals for the XOR operation.
2. Observe the output z for all these N values and sort it out into two groups depending on whether the target bit of input x is 0 or 1.
3. Compute the average power consumption for each group and verify the correctness of the original guess bit of y using the averages.

These results are also applicable when other SCA attacks such as timing or electro magnetic analysis are mounted on the MACs [36].

4 DPA Analysis of *Type-1* Schemes

In this section, we perform DPA analysis of the *Type-1* MAC schemes. Some of these MACs have their own padding rules whereas some others follow the padding functionality defined for the MD hashes.

4.1 BNMAC and Its One-Key Variants

Yasuda [43] proposed BNMAC as an alternative to HMAC to achieve higher performance than HMAC when it is implemented with the slower SHA-2 family. BNMAC uses an alternative hash function framework to the MD called hyper Merkle-Damgård (HMD). Yasuda also proposed two practical variants of BNMAC that use only one secret key.

Hyper Merkle-Damgård. An arbitrary length message m to be processed using a HMD hash function H is split into blocks as $m = m[1]\|m[2]\| \ldots m[2t-1]\|m[2t]$ such that $|m[2i-1]| = n$ and $|m[2i]| = b$ for $i = 1, 2, \ldots, t$. The intermediate hash value $H[i]$ at any iteration i is given by $H[i] = h_{H[i-1]\oplus m[2i-1]}(m[2i])$ where $H[0]$ is the IV of H.

BNMAC and its variants. If k_1 and k_2 are two independent and random secret keys then the BNMAC function is defined by:

$$\text{BNMAC}_{k_1,k_2}(m) = h_{k_1}(H_{k_2}(m)\|1^{b-n})$$

BNMAC always uses a secure one-to-one padding for m so that the size of m is always a multiple of $b+n$ blocks. The first variant is $\text{BNMAC1}_k(m) = h_{k_1}(H_{k_2}(m)\|1^{b-n})$ where the two keys k_1 and k_2 are derived from the key k as given by $k_1 = h_k(1^b)$ and $k_2 = h_k(0^b)$. The second variant is $\text{BNMAC2}_k(m) = h_k(H_{h_k(0^b)}(m)\|1^{b-n})$.

DPA attacks on BNMAC and its variants. By assuming $b = n$, we can instantiate BNMAC with any of the twelve PGV schemes. Then we have $|k_1| = |H_{k_2}(m)|$. In this setting, the first and second set of n input bits to the compression function h correspond to the chaining value and the message block of h. At every iteration i of $\text{BNMAC}_{k_1,k_2}(m)$, the XOR operation $H[i-1]\oplus m[2i-1]$ would become the target on which we mount the DPA attack to recover the previous secret state information $H[i-1]$ where $H[0]$ is the secret key k_2. In fact, this inner key recovery attack on BNMAC is independent of the security of the compression function as the target XOR operation on which we mount the DPA attack is external to the compression function.

Once the key k_2 is recovered, we can mount the DPA attack on the target operation $k_1 \oplus H_{k_2}(m) = \text{BNMAC}_{k_1,k_2}(m)$ based on h^j where $j \in \{2, 4, 6, 8, 9, 10, 11, 12\}$ to recover the secret key k_1. We can mount RDPA attack on this target operation for BNMAC based on h^j where $j \in \{5, (3,1), (3,2), (7,1), (7,2)\}$ to recover the secret key k_1. There is no target XOR operation for BNMAC based on h^j where $j \in \{1, (3,3), (7,3)\}$ on which we could mount the DPA attacks to recover the key k_1. These attacks also apply to BNMAC1 and BNMAC2.

Forging BNMAC. Recovering only k_2, BNMAC based on any secure h can be *existentially forged* with just one oracle query as follows:

1. Query the BNMAC oracle with $m = m[1]\|m[2]\| \ldots m[2t]$ and obtain its tag $\text{BNMAC}_{k_1,k_2}(m)$.

2. Forge BNMAC by computing the tag $\mathrm{BNMAC}_{k_1,k_2}(m^*)$ of the message $m^* = m[1]\|m[2]\|h(m[1] \oplus k_2\|m[2]) \oplus (m[1] \oplus k_2)\|m[2]\| \ldots m[2t]$ such that $\mathrm{BNMAC}_{k_1,k_2}(m^*) = \mathrm{BNMAC}_{k_1,k_2}(m)$.

Note that one can trivially find collisions for the HMD hash. BNMAC invoked with the PGV schemes for which both keys can be recovered can be *universally forged* by computing the tag for any given message.

4.2 Enveloped Merkle-Damgård (EMD) Transform

Bellare and Ristenpart [2] proposed a variant of MD called EMD which works as a MAC when its keyed compression function is a PRF.

EMD construction. An arbitrary length message m to be processed using EMD scheme H is split into b-bit blocks $m[1]\|m[2]\| \ldots m[t-1]$ and incomplete bits are filled in the last block $m[t]$ where $b \geq n + 64$. The last 64 bits of $m[t]$ contain the binary format of $|m|$. At any iteration i of H, the intermediate hash value is given by $H[i] = h_{H[i-1]}(m[i])$ where $1 \leq i \leq t-1$ and $H[0]$ is the IV of H. The hash value is $H[t] = h_{H[0]^*}(H[t-1]\|m[t])$ where $H[0]^*$ is the IV of the final compression function h and $H[0] \neq H[0]^*$. The constraint $b > n$ allows us to use only h^5 as the compression function for the EMD hash.

Keying EMD. The EMD construction keyed through its IVs is defined by $\mathrm{EMD}_{k_1,k_2}(m) = h_{k_1}(H_{k_2}(m[1]\|m[2]\| \ldots m[t-1])\|m[t])$.

Trail secret key-recovery of keyed EMD. We can mount the RDPA attack on the target operation $k_1 \oplus G_{H_{k_2}(m)\|m[t]}(k_1) = \mathrm{EMD}_{k_1,k_2}(m)$ in the outer compression function of EMD MAC based on h^5 to recover the secret key k_1. Note that the control over $b-n-64$-bit input in the block $m[t]$ does not provide us any additional advantage to recover the key k_1. Once we know the key k_1, the current security proof of EMD MAC does not guarantee its security against forgery attacks.

4.3 Merkle-Damgård with Permutation (MDP)

Hirose, Park and Yun [18] proposed a minor variant of the MD called Merkle-Damgård with permutation (MDP). MDP keyed through its chaining value works as a MAC when the underlying keyed compression function is a PRF and secure against a very mild related-key attack.

MDP construction. The MDP construction is obtained by processing the last intermediate hash value of MD using a fixed permutation π. An arbitrary length message m to be processed using MDP is split into b-bit blocks $m[1]\|m[2]\| \ldots \|m[t]$. At any iteration i of the MDP construction, the intermediate hash value is given by $H[i] = h_{H[i-1]}(m[i])$ where $1 \leq i \leq t-1$. The hash value of m is $H[t] = h_{\pi(H[t-1])}(m[t])$. The message m is padded such that the last l bits of $m[t]$ contain $|m|$ in the binary format. We can instantiate MDP using any of the twelve PGV schemes.

Keyed MDP. The MDP scheme keyed through its IV works as a MAC and is called KMDP in [18]. For $1 \leq i \leq t$, the KMDP function is defined by $\text{KMDP}_k(m) = h_{\pi(H[t-1])}(h_k(m[i]))$.

DPA analysis of KMDP. The DPA attack can be mounted on the target XOR operation $H[i-1] \oplus m[i] = H[i]$ of KMDP based on h^j for $j \in \{2, 3(1), 4, 6, 7(1), 8, 9, 10, 11, 12\}$ to recover the secret key k where $H[0] = k$. KMDP based on h^5 is vulnerable to a variant of the RDPA attack as outlined below:

1. Consider a variable 2-block message $m = m[1] \| m[2]$ where the first $b - l - 2$ bits of $m[2]$ contain the information followed by the padding bits $1 \| 0$ and then the binary format of $b + b - l - 2$-bit length of the true message in the last l bits of $m[2]$.
2. We repeat the following for N^2 number of random values of m to collect N values of $H[1]$:
   - Choose $m[1]$ and fix it. Mount the RDPA attack on the target operation $\pi(H[1]) \oplus G_{m[2]}(\pi(H[1])) = \text{KMDP}_k(m)$ in the second iteration of KMDP to recover the secret $\pi(H[1])$ by collecting tags $\text{KMDP}_k(m)$ for N values of m by varying the first $b-l-2$ bits of $m[2]$. We then compute $\pi^{-1}(\pi(H[1]))$ to obtain the output $H[1]$ of the first iteration.
3. Finally, we recover the key k by mounting the RDPA attack on the target operation $k \oplus G_{m[1]}(k) = H[1]$ in the first compression function by using N values of $H[1]$ recovered in step 2.

Similarly, RDPA attack can also be mounted on the BNMAC function based on $h^{3(2)}$ and $h^{7(2)}$ to detect the secret key k. In practice, about $N = 100,000 \approx 2^{17}$ samples are required to mount the RDPA attack once on the target operation of a MAC function implemented in an IC chip [36]. Hence, about 2^{35} runs of the compression function of KMDP based on h^j where $j \in \{5, 3(2), 7(2)\}$ implemented on an IC chip are required to recover the secret key.

KMDP based on h^1, $h^{3(3)}$ and $h^{7(3)}$ does not have a target XOR operation on which we could mount the DPA attacks. KMDP based on the PGV compression functions that are vulnerable to the DPA attacks can be *universely* forged for any given message.

4.4 Multilane NMAC

L-lane NMAC and HMAC algorithms. Yasuda [44] proposed a provably secure n-bit L-Lane NMAC ($L \geq 2$), which we call LNMAC, to increase the security level of n-bit NMAC from $2^{n/2}$ to 2^n evaluations against forgery attacks. LNMAC uses L lanes of an MD hash to process an arbitrary length message. Each lane of LNMAC uses an independent random secret key of size n bits as the IV of the hash function in that lane. The proof of security of LNMAC as a PRF and hence as a MAC requires $b \geq 2n$. This condition on b allows us to invoke the LNMAC algorithm with *only* h^5 out of twelve PGV schemes. The 2NMAC function is defined below where $i = 1, 2, \ldots, t$ and τ is the authentication tag:

$$2\text{NMAC}_k(m) = h_{k'}(h_{k'_1}(m[i]) \| h_{k'_2}(m[i]) \| 0^{b-2n}) = \tau$$

Trail secret key-recovery of LNMAC. There is no target XOR operation in 2NMAC based on h^5 on which we can mount the DPA attack to recover the secret keys k'_1 and k'_2. Now let $u = G_{(h_{k'_1}(m[i]) \| h_{k'_2}(m[i]) \| 0^{b-2n})}(k')$. We can mount the RDPA attack on the target operation $k' \oplus u = \tau$ in the last compression function to recover the secret key k'. Similarly, we can recover the key k' for LNMAC as this RDPA attack is independent of the number of lanes. Once we know the trail secret key of LNMAC, its security proof does not guarantee its MAC security.

4.5 O-NMAC

The MAC function O-NMAC was proposed as a one-key variant of NMAC in [14]. O-NMAC computes a linear-XOR checksum using the intermediate hash values of the MD hash function and process it as a final message block. While O-NMAC was analysed informally in [14], a security proof for O-NMAC as a MAC function was provided in [25] under the name Enveloped Checksum Merkle-Damgård (ECM) transform. The O-NMAC function keyed through its IV with a random key k is defined by:

$$\text{O-NMAC}_k(m) = h_{\oplus_{i=1}^{t} h_{H[i-1]}(m[i])}(h_k(m) \| m')$$

where m is split into b-bit blocks $m[1] \| m[2] \| \ldots m[t]$ with the last block $m[t]$ having the binary encoded format of m in its last l bits, $H[0] = k$, $H[i] = h_{H[i-1]}(m[i])$ and m' contains the padding bits including the length encoding of the n-bit value $h_k(m)$ in its last l bits. We assume $|k| = b = n$. Then there is no need to pad $h_k(m)$ with the bits m'. Note that if the message has only one block then a separate block is used to pad it. Hence, at least three iterations of the compression function h are required to compute the authentication tag of an arbitrary length message using O-NMAC.

DPA analysis of O-NMAC. The DPA attack can be mounted on the target XOR operation $H[i-1] \oplus m[i] = H[i]$ of O-NMAC based on h^j for $j \in \{2, 3(1), 4, 6, 7(1), 8, 9, 10, 11, 12\}$ to recover the secret key k where $H[0] = k$. Hence, O-NMAC instantiated with these PGV schemes can be *universally* forged.

When we try to mount the RDPA attack on O-NMAC instantiated with h^5, say using a 2-block message $m = m[1] \| m[2]$, both operands on the left hand side of the expression $H[2] \oplus G_{H[1] \oplus H[2]}(H[2]) = \text{O-NMAC}_k(m)$ are variable. Hence, we cannot mount the RDPA attack. Similar analysis holds for O-NMAC implemented with $h^{3(2)}$ and $h^{7(2)}$. There is no target XOR operation in O-NMAC based on h^j where $j \in \{1, 3(3), 7(3)\}$ on which we could mount the DPA attacks.

5 DPA Analysis of *Type-2* Schemes

In this section, we perform DPA analysis of the NMAC setting of the *Type-2* hash schemes which can also be extended to their HMAC version.

5.1 MDC-2 Hash Function in the NMAC Setting

MDC-2 hash function. MDC-2 [11,32] is a $2n$-bit provably secure hash function based on an n-bit ideal block cipher [39]. MDC-2 is an MD mode of MMO scheme (h^1) in parallel paths. We follow the description of MDC-2 in [39] which is generalised for any ideal block cipher G with the same key and block sizes. If $H[0]$ and $H'[0]$ are two different IVs then the intermediate hash value of MDC-2 at any iteration i is defined by $H[i]\|H'[i] = h^1_{H[i-1]}(m[i])\|h^1_{H'[i-1]}(m[i])$.

MDC-2 hash function in the NMAC setting. Let $k_1 = k'_1\|k^*_1$ and $k_2 = k'_2\|k^*_2$ be any two $2n$-bit keys such that k'_1,k^*_1,k'_2 and k^*_2 are four random and independent keys each of n bits. Then the MDC-2 hash in the NMAC setting is defined by MDC-$2_{k_1,k_2}(m) = h^1_{k_1}(H^1_{k_2}(m))$. At any iteration i, the intermediate hash value of this MAC is given by $H[i]\|H'[i] = h^1_{H[i-1]}(m[i])\|h^1_{H'[i-1]}(m[i])$ where $H[0] = k_1$ and $H'[0] = k_2$.

DPA analysis. There is no target XOR operation in the compression function of MDC-$2_{k_1,k_2}(m)$ on which we could mount the DPA attacks to recover the secret keys.

Remark 1. MDC-4 [8,29] is an extended MDC-2 which uses two sequential executions of MDC-2 to process one message block. The DPA analysis of keyed MDC-2 is also applicable to keyed MDC-4. Similarly, NMAC version of the MD mode of the MAME compression function [45] which uses a novel block cipher algorithm in the MMO mode is also secure against our DPA attacks when its block cipher algorithm is assumed to be ideal and secure against side channel attacks. The MAME compression function was claimed to be transformed to a light weight hash function using any domain extension algorithm and such hash function can withstand side channel attacks when it is used as a key derivation function. We note that not all MAC or key derivative versions of such hash modes may resist DPA attacks even when the block cipher of MAME is ideal. For example, BNMAC based on MAME can be *existentially forged*.

5.2 Grindahl Compression Function in the NMAC Setting

Grindahl compression function design. The Grindahl compression function [24] h processes every block $m[i]$ by concatenating it with the previous state $H[i-1]$ using the permutation G and then truncates the output of G to n-bit state $H[i]$. At any iteration i, its intermediate hash value is defined by $H[i] = h'(G(m[i]\|H[i-1]))$ where h' is the truncation function. When h is iterated in the MD mode, the output of the permutation of the last message block (padded block) is not truncated; instead an output transformation with a pre-defined number of blank rounds is executed followed by the truncation step to output n-bit hash value.

Keying Grindahl compression function. The MD hash iteration of Grindahl compression function can be defined in the NMAC setting using two random and independent secret keys k_2 and k_1. The initial value $H[0]$ of the hash function is replaced with the key k_2 and a key k_1 is used as a key to the outer compression function h. The outer function may require more than one iteration if blank rounds are also defined for the outer function. Its HMAC version can be defined as in HMAC where the two keys k_1 and k_2 are derived from a master key k.

DPA analysis of Keyed Grindahl. There is no target XOR operation in the NMAC setting of Grindahl on which we could mount the DPA attacks when the block algorithm G is ideal. This result complements the claim of [24] on using a side channel resistant AES implementation as the underlying block cipher to protect the keyed implementations of Grindahl members from side channel attacks. Note that the collision attack on Grindahl-256 [37], a specific 256-bit hash function following the design strategy of Grindahl compression function has no influence on our analysis as our analysis assumes an ideal G independent of any specific details.

5.3 Wide-Pipe Hash Construction in the NMAC Setting

Wide-pipe hash. The wide-pipe hash construction [27] uses a large compression function $h : \{0,1\}^{2n} \times \{0,1\}^b \rightarrow \{0,1\}^{2n}$ to process a b-bit message block where $b \geq 2n$ and once the complete message is processed, it uses a function $h' : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ to truncate $2n$-bit output to an n-bit hash value. We assume that the least n significant bits of the output are truncated to produce high order n bits as the hash value.

NMAC with wide-pipe. As noted in [44], a variant of HMAC can be constructed using wide-pipe hash with an n-bit key k. Similarly, we can construct NMAC using wide-pipe hash with two independent n-bit keys k_1 and k_2. Let $k = k_1 \| k_1$ and $k' = k_2 \| k_2$ be two $2n$-bit keys keyed through the IV of the inner/outer wide-pipe hashes of NMAC. We call keyed wide-pipe as WNMAC and define it by $\text{WNMAC}_{k,k'}(m) = h'(h_{k'}(h'(H_k(m))))$.

DPA analysis of WNMAC. For WNMAC based on h^j for $j \in \{2, 3(1), 4, 6, 7(1), 8, 9, 10, 11, 12\}$, the secret key k can be recovered by mounting the DPA attack on the target XOR operation $H[i-1] \oplus m[i] = H[i]$ in the function h^j where $H[0] = k$ as for NMAC. For WNMAC based on h^5, we can mount the RDPA attack on the target XOR operation $G_{h'(H_k(m))}(k') \oplus k' = h_{k'}(h'(H_k(m)))$ to recover the high order n bits of k' which are equal to the tag $\text{WNMAC}_{k,k'}(m)$ and then recover k'. Similarly, we can mount the RDPA attack on WNMAC based on h^j where $j \in \{3(2), 7(2)\}$. There is no target XOR operation in WN-MAC based on h^j where $j \in \{1, 3(3), 7(3)\}$ on which we could mount the DPA attacks.

Remark 2. Protecting a hash based MAC function from the DPA attacks by masking target XOR or addition operations requires developing a whole new hardware module for that MAC function instead of using a widely implemented DPA resistant hardware module of a cryptographic algorithm such as a block cipher. Hence, constructing DPA resistant hash based MACs by using combinations of appropriate key settings and provably secure hash and compression function modes that do not expose any target XOR or addition operations when they are combined with DPA resistant cryptographic hardware modules allows us to reuse these hardware modules.

6 Conclusion

Our research leaves a number of questions open: Among these, the most interesting is, how to design a secure hash mode which can be turned into a DPA resistant provably secure MAC when it is instantiated with any of the secure PGV schemes. The other interesting question is on defining provably secure MAC versions or NMAC settings for some new hash function frameworks such as HAIFA [6] and double-pipe hash [27] invoked with 12 PGV schemes and analyse them against DPA attacks. The final question is how to plug an alternative hash framework to MD into NMAC/HMAC? We believe that our work and future developments in this area of research would provide much needed insights to the designers of hash functions who compete in the AHS process.

Acknowledgments. Many thanks to William Burr and John Kelsey for their encouragement, discussions on the subject and timely responses to our questions regarding SHA-3 evaluation process. We also thank Shoichi Hirose and Krystian Matusiewicz for comments on our paper.

References

1. ANSI. ANSI X9.31:1998: Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA). American National Standards Institute (1998)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117. Springer, Heidelberg (2006)
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
4. Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)

5. Bellovin, S.M., Rescorla, E.K.: Deploying a New Hash Algorithm. In: Proceedings of NDSS. Internet Society (February 2006)
6. Biham, E., Dunkelman, O.: A framework for iterative hash functions - HAIFA. Cryptology ePrint Archive, Report 2007/278 (2007) (Accessed on 5/14/2008), `http://eprint.iacr.org/2007/278`
7. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
8. Bosselaers, A., Preneel, B.: Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040. In: Bosselaers, A., Preneel, B. (eds.) RIPE 1992. LNCS, vol. 1007, pp. 31–67. Springer, Heidelberg (1995)
9. Burr, W.: Personal Communication regarding Frequently Asked Questions on AHS Competition (March 2008)
10. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
11. Coppersmith, D., Pilpel, S., Meyer, C.H., Matyas, S.M., Hyden, M.M., Oseas, J., Brachtl, B., Schilling, M.: Data authentication using modification dectection codes based on a public one way encryption function. U.S. Patent No. 4,908,861, March 13 (1990)
12. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
13. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
14. Gauravaram, P.: Cryptographic Hash Functions: Cryptanalysis, Design and Applications. PhD thesis, Information Security Institute, Queensland University of Technogy (June 2007)
15. Gauravaram, P., Kelsey, J.: Linear-XOR and Additive Checksums Don't Protect Damgård-Merkle Hashes from Generic Attacks. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 36–51. Springer, Heidelberg (2008)
16. Gauravaram, P., McCullagh, A., Dawson, E.: Collision Attacks on MD5 and SHA-1: Is this the "Sword of Damocles" for Electronic Commerce?. In: AusCERT R & D Stream, pp. 1–13 (2006)
17. Gauravaram, P., Okeya, K.: An Update on the Side Channel Cryptanalysis of MACs Based on Cryptographic Hash Functions. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 393–403. Springer, Heidelberg (2007)
18. Hirose, S., Park, J.H., Yun, A.: A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 113–129. Springer, Heidelberg (2007)
19. ISO/IEC 10118-2. Information Technology - Security Techniques- Hash Functions-Hash functions using an n-bit block cipher. ISO (2000)
20. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
21. Kelsey, J.: How Should We Evaluate Hash Submissions?. In: ECRYPT Hash Function Workshop (2007) (Accessed on 02/13/2008), `http://csrc.nist.gov/groups/ST/hash/documents/kelsey-ECRYPT2007.pdf`

22. Kelsey, J.: How to Choose SHA-3?.In: ECRYPT Hash Function Workshop (2008) (Accessed on 07/26/2008), `http://www.lorentzcenter.nl/lc/web/2008/309/presentations/Kelsey.pdf`
23. Kelsey, J., Schneier, B.: Second Preimages on n-bit Hash Functions for Much Less than 2n̂ Work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
24. Knudsen, L.R., Rechberger, C., Thomsen, S.S.: The Grindahl Hash Functions. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 39–57. Springer, Heidelberg (2007)
25. Lei, D., Chao, L.: Extended Multi-Property-Preserving and ECM-construction. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 361–372. Springer, Heidelberg (2007)
26. Lemke, K., Schramm, K., Paar, C.: DPA on n-bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC-Construction. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 205–219. Springer, Heidelberg (2004)
27. Lucks, S.: A Failure-Friendly Design Principle for Hash Functions. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 474–494. Springer, Heidelberg (2005)
28. McEvoy, R.P., Tunstall, M., Murphy, C.C., Marnane, W.P.: Differential power analysis of HMAC based on SHA-2, and countermeasures. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 317–332. Springer, Heidelberg (2008)
29. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography, ch. 9, pp. 321–383. CRC Press, Boca Raton (1997)
30. Merkle, R.: One way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
31. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of power analysis attacks on smartcards. In: Proceedings of the USENIX Workshop on Smartcard Technology, pp. 151–162. USENIX Association (1999)
32. Meyer, C., Schilling, M.: Secure program load with manipulation detection code. In: Proceedings of the 6th Worldwide Congress on Computer and Communications Security and Protection (SECURICOM 1988), Paris, pp. 111–130 (1988)
33. NIST. Federal Information Processing Standard (FIPS PUB 198) The Keyed-Hash Message Authentication Code (HMAC) (March 2002)
34. NIST. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Docket No: 070911510-7512-01 (November 2007)
35. NIST. Federal Information Processing Standard (FIPS PUB 180-3) Secure Hash Standard (2007)
36. Okeya, K.: Side Channel Attacks Against HMACs Based on Block-Cipher Based Hash Functions.. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 432–443. Springer, Heidelberg (2006)
37. Peyrin, T.: Cryptanalysis of Grindahl. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 551–567. Springer, Heidelberg (2007)
38. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
39. Steinberger, J.P.: The collision intractability of MDC-2 in the ideal-cipher model. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 34–51. Springer, Heidelberg (2007)

40. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)
41. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
42. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
43. Yasuda, K.: Boosting Merkle-Damgård Hashing for Message Authentication. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 216–231. Springer, Heidelberg (2007)
44. Yasuda, K.: Multilane HMAC - Security beyond the Birthday Limit. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 18–32. Springer, Heidelberg (2007)
45. Yoshida, H., Watanabe, D., Okeya, K., Kitahara, J., Wu, H., Küçük, Ö., Preneel, B.: MAME: A Compression Function with Reduced Hardware Requirements. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 148–165. Springer, Heidelberg (2007)