Self-evaluation Report
PC-MAC-AES

NEC Corporation

Contents

1 Overview

This document is a self-evaluation report on a message authentication code (MAC) called PC-MAC-AES.

As introduction, we first explain the background and the design criteria of PC-MAC-AES. From the development of AES [7], it is now the most widely deployed block cipher in the world. The security of AES has been thoroughly studied and is believed to be very high. Following this situation, it is quite natural to build a message authentication code based on AES, i.e., a MAC mode of operation. We have in fact many MAC modes, such as CBC-MAC, EMAC[9], CMAC[8], etc. An advantage of these MAC modes are their provable security, which means that it is secure as long as the underlying block cipher is also secure. This property is also called a security reduction.

However, since they use a block cipher in a black-box manner they inevitably need at least one block cipher invocation to process one message block. That is, if AES is used with CBC-MAC, it can not be faster than AES itself. An approach to break this limitation is to introduce another keyed function called universal hash function, which is usually based on some algebraic operations such as polynomial computation over a finite field, and combine it with AES. It is known (e.g. [11][15]) that this approach can yield a much faster MAC than AES. However the performance of popular universal hash functions are highly depending on the target platform. For example, the polynomial hash (which is one of the most popular universal hash functions) can be very fast if much key-dependent precomputations are done and the results are stored, which implies a fast software implentation. On the other hand, this implementation is generally infeasible for hardwares.

Here, one may naturally think of using AES in a non-black-box manner, that is, using a reduced-round version of AES, to obtain a faster MAC than AES. The first one of such an attempt is, to our knowledge, due to Daemen et al .[13]. Their proposal was called alpha-MAC. There also exists a successor called Pelican[14]. Though these proposals received many attentions, they had a critical drawback that they lost the provable security based on the security of AES. This implies the existence of attack against them irrespective of the existence of attack against AES.

Our proposal, PC-MAC-AES, solves the above-mentioned drawback. We use a 4-round AES in a specific way, which allows us to exploit the known result on the security against differential cryptanalysis against (a reduced-round) AES[19][20]. As a result, PC-MAC-AES is 1.4 to 2.5 (for recommended parameters 1.4 to 2.0) times faster than AES while it has a provable security based on the security of AES. PC-MAC-AES is almost the same as PC-MAC proposed by Minematsu and Tsunoo [23]. The basic structure is similar to CBC-MAC, where AES and its 4-round version are periodically invoked. Unlike alpha-MAC and Pelican, PC-MAC-AES derives the keys of 4-round AES function from an AES-based key schedule.

2 Summary of Security Evaluation

Following reference [23], it is proved that if AES is computationally hard to be distinguished from a uniform random permutation, PC-MAC-AES is also computationally hard to be distinguished from a variable-input-length uniform random function, which is an ideal of deterministic MAC function. This assumption on AES is the same as most known MAC mode of operations (such as CBC-MAC or CMAC [8]) using AES. In the proof, the analytic results by Keliher et al.[19][20] about the maximum expected differential probability (MEDP) of 4-round AES plays an important role. Combining this result and a previous study [10], we can prove that it is computationally hard to make a forgery against PC-MAC-AES under (adaptive) chosen plaintext attacks, which is the standard security requirement of MAC functions. The details of security evaluation are given below. Hereafter we use the word “AES” to mean AES with a 128-bit key. For detailed specification of PC-MAC-AES (e.g. the meaning of parameters, π, d), we refer to the technology specification document [1].

3 Security Evaluation

3.1 Preliminaries

3.2 Security Notion and Attack Model

An adversary, $\mathcal{A}$, against a deterministic MAC function, F, is assumed to be accessible to the tagging oracle and verification oracle, denoted by O_T^F and O_V^F. A tagging query to O_T^F made by $\mathcal{A}$ is a message of any length, M. When M is queried to O_T^F, it returns $F(M)$ to $\mathcal{A}$. A verification query to O_V^F is a pair (M', T'), where M' is a message that has not been queried to O_T^F before and T' is a string of tag length (which is predetermined). O_V^F returns 1 if $F(M') = T'$ and 0 otherwise. The goal of $\mathcal{A}$ is to obtain a response 1 from O_V^F, since this implies the $\mathcal{A}$ succeeds in computing the real tag value for a message without querying it to O_T^F. We call this event a (successful) forgery. We define the occurrence of response 1 from O_V^F as the success of $\mathcal{A}$.

Let $\mathcal{A}$'s time complexity be τ, the number of tagging queries be q, the number of verification queries be q_v, and the maximum block length of a message (for both tagging and verification queries) be ρ (here the bit length of one block is assumed to be predetermined: in our case it is 128 bits). Then $\mathcal{A}$ is called a (q, q_v, τ, ρ)-forger. Also, $\mathtt{FP}_F(\mathcal{A})$ denotes the maximum probability of receiving (at least one) response 1 from O_V^F. Here,

$$\mathtt{FP}_F(q, q_v, \tau, \rho) = \max_{\mathcal{A}:(q,q_v,\tau,\rho)\text{-forger}} \mathtt{FP}_F(\mathcal{A})$$

is called the maximum forgery probability and we deem it the security notion for MACs.

3.3 Pseudorandomness

For a deterministic MAC F, we can derive the maximum forgery probability defined above by evaluating the pseudorandomness of F, that is, the difficulty of distinguishing F from a truly random function, R, having the same domain and range as F. For this purpose, we consider a game in which an adversary can access one of two oracles, O_T^F and O_T^R, without knowing which one is true. The adversary makes a binary decision using a certain number of queries. Here queries can be adaptively chosen, that is, the adversary performs a chosen plaintext attack (CPA).

If an adversary, $\mathcal{B}$, is in the game defined above and performs q queries with time complexity τ, $\mathcal{B}$ is called a (q, τ, ρ)-distinguisher. In addition, let $\mathcal{B}^F = 1$ be the event that $\mathcal{B}$ makes a final decision being 1.

Now we have

$$\mathtt{Adv}_{F,R}^{\mathtt{cpa}}(\mathcal{B}) = |\Pr[\mathcal{B}^F = 1] - \Pr[\mathcal{B}^R = 1]|,$$

which is called the advantage of $\mathcal{B}$ in distinguishing F from R. The pseudorandomness of F is evaluated as the maximum advantage in distinguishing it from R. That is, we say F is pseudorandom if

$$\mathtt{Adv}_F^{\mathtt{prf}}(q, \tau, \rho) = \max_{\mathcal{B}:(q,\tau,\rho)\text{-distinguisher}} \mathtt{Adv}_{F,R}^{\mathtt{cpa}}(\mathcal{B})$$

is negligibly small (for parameters (q, τ, ρ)).

Specifically, if F is an n-bit permutation with a key (i.e. an n-bit block cipher), we also define

$$\mathtt{Adv}_F^{\mathtt{prp}}(q, \tau) = \max_{\mathcal{B}:(q,\tau)\text{-distinguisher}} \mathtt{Adv}_{F,P}^{\mathtt{cpa}}(\mathcal{B}),$$

where P is an n-bit uniform random permutation as a more appropriate measure for pseudorandomness. Note that ρ is omitted in the above as a query is always n bits.

Moreover, if F accepts inputs of any length we define R^* as a uniform random function with inputs of any length and let

$$\mathtt{Adv}_F^{\mathtt{vilprf}}(q,\tau,\rho) = \max_{\mathcal{B}:(q,\tau,\rho)\text{-distinguisher}} \mathtt{Adv}_{F,R^*}^{\mathtt{cpa}}(\mathcal{B})$$

to use the measurement of pseudorandomness.

3.4 Security of PC-MAC-AES

The security of PC-MAC-AES can be proved as follows. First, we prove the pseudorandomness of PC-MAC-AES based on the assumption that AES is pseudorandom. Using this and a known result of [10] we then prove that the maximum forgery probability against PC-MAC-AES is negligibly small.

To prove the pseudorandomness of PC-MAC-AES we need to know the differential probability of the component, the 4-round AES function G_U written in Fig. 1. More specifically the followings are needed.

Definition 3.1 If F_K is an n-bit permutation with a key, K, the maximum expected differential probability (MEDP) and maximum expected self differential probability (MESDP) of F_K are defined as

$$\begin{aligned}\mathrm{MEDP}(F_K) &= \max_{a\neq 0,b} \Pr(F_K(X)\oplus F_K(X\oplus a)=b), \text{ and}\\ \mathrm{MESDP}(F_K) &= \max_{a\in\{0,1\}^n} \Pr[X\oplus F_K(X)=a],\end{aligned}$$

where X is distributed uniformly over the n-bit space and the probabilities are defined by the distributions of X and K.

For 4-round AES function G_U, Keliher et al. [19][20] proved that $\mathrm{MEDP}(G_U)\le 2^{-113}$ if the 384-bit key U is uniform. For MESDP of G_U, it is easy to observe that $\mathrm{MESDP}(G_U)=2^{-128}$ as G_U's round sub keys are XORed to the intermediate message, implying that X and $G_U(X)$ are independent.

We mention that the reference [23] suggested using 4-round AES function with omission of the last ShiftRows and MixColumns functions. Note that we here specified PC-MAC-AES without this omission. However, this difference does not affect the provable security. This is because

$$\mathrm{MEDP}(F) = \mathrm{MEDP}(g\circ F)$$

holds for any n-bit permutation (with key), F, and an n-bit linear permutation, g. That is, the above equation implies that the result of Keliher et al. holds true regardless of if last ShiftRows and MixColumns exist or not.

In Theorem 2 of reference [23], the following is proved.

Thorem 3.1 For $c=\lceil d|\mathcal{U}|/n\rceil + d$ we have

$$\mathtt{Adv}^{\mathtt{vilprf}}_{\text{PC-MAC}_d[E_K,L|G_U]}(q,\tau,\rho) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(\rho q+c,\tau') + \frac{2.5(\rho q+c)^2}{2^n} + (d\epsilon_{\mathrm{dp}}+\epsilon_{\mathrm{sdp}})\frac{q^2}{2},$$

where U is in $\mathcal{U}$, $t'=t+O(\rho q)$, $\epsilon_{\mathrm{dp}}=\mathrm{MEDP}(G_U)$, $\epsilon_{\mathrm{sdp}}=\mathrm{MESDP}(G_U)$.

Here, $\text{PC-MAC}_d[E_K,L|G_U]$ corresponds to a generalized version of $\text{PC-MAC-AES}_{(128,d)}$ specified by the technology specification document [1]. In fact if we assume E_K as AES encryption function and G_U as 4-round AES function, $\text{PC-MAC}_d[E_K,L|G_U]$ corresponds to $\text{PC-MAC-AES}_{(128,d)}$.

Hence, we can obtain a result on the pseudorandomness of $\text{PC-MAC-AES}_{(\pi,d)}$ by setting $n=128$, $|\mathcal{U}|=384$, $\mathrm{MEDP}(G_U)\le 2^{113}$, and $\mathrm{MESDP}(G_U)=2^{-128}$.

Corollary 3.1 For any $d\ge 1$ and $1\le\pi\le 128$, we have

$$\mathtt{Adv}^{\mathtt{vilprf}}_{\text{PC-MAC-AES}_{(\pi,d)}}(q,\tau,\rho) \le \mathtt{Adv}^{\mathtt{prp}}_{E_K}(\rho q+4d,\tau') + \frac{2.5(\rho q+4d)^2}{2^{128}} + \left(\frac{d}{2^{114}}+\frac{1}{2^{129}}\right)q^2,$$

where $\tau'=\tau+O(\rho q)$.

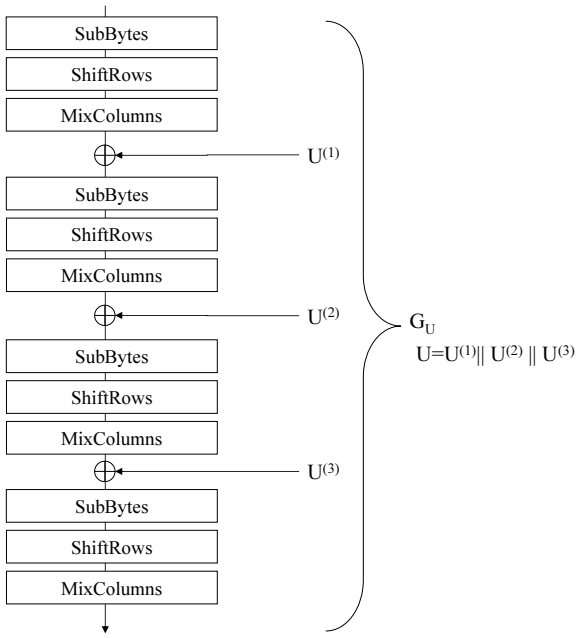

Fig. 1 4-round AES function, G_U

Note that the security bound proved here is independent of the tag length π because the tag truncation does not help adversary.

Finally, we derive the maximum forgery probability of PC-MAC-AES$_{(\pi,d)}$, which is as follows.

If we use a variable-input-length deterministic MAC function, F, we have

$$\mathtt{FP}_F(q, q_v, \tau, \rho) \leq \mathtt{Adv}_F^{\mathtt{vilprf}}(q + q_v, \tau', \rho) + \frac{q_v}{2^\pi}, \tag{1}$$

where $\tau' = \tau + O((q + q_v)\rho)$. This is proved by Bellare et al. [10], Proposition 7.3.

Therefore we combine Corollary 3.1 and Equation (1) to have

$$\begin{aligned}
\mathtt{FP}_{\text{PC-MAC-AES}_{(\pi,d)}}(q, q_v, \tau, \rho) &\leq \mathtt{Adv}^{\mathtt{vilprf}}_{\text{PC-MAC-AES}_{(\pi,d)}}(q + q_v, \tau', \rho) + \frac{q_v}{2^\pi}, && (2)\\
&\leq \mathtt{Adv}^{\mathtt{prp}}_{E_K}(\rho(q + q_v) + 4d, \tau') && (3)\\
&\quad + \frac{2.5(\rho(q + q_v) + 4d)^2}{2^{128}} + \left(\frac{d}{2^{114}} + \frac{1}{2^{129}}\right)(q + q_v)^2 + \frac{q_v}{2^\pi}. && (4)
\end{aligned}$$

Here, the first term of the right hand side of Equation (4) is representing the pseudorandomness of AES, thus, the maximum forgery probability against PC-MAC-AES$_{(\pi,d)}$ is negligibly small if AES is pseudorandom (computationally hard from distinguishing it from uniform random permutation).

In the above-mentioned maximum forgery probability bound of PC-MAC-AES$_{(\pi,d)}$, the dominant term is $\frac{d}{2^{114}}(q + q_v)^2$ in practice. Hence, if we set $d \leq 5$ and $\pi \geq 64$ following the technology specification [1], PC-MAC-AES$_{(\pi,d)}$ is secure if q and q_v are sufficiently smaller than 2^{56}. Although this bound is generally smaller than those provided by CBC-MAC and its variants (e.g. EMAC and CMAC): they provide the bound about 2^{64} However, we think this difference in security bounds has no impact on practical applications.

3.5 On Secondary Key L

The key of PC-MAC-AES consists of AES's key K and another 128-bit key, L. However, the purpose of introducing L is not to achieve 256-bit security. Here, L is introduced to implement a tweakable

block cipher[22], a generalization of block cipher, for achieving the high efficiency in a simple way. If L does not exist, the adversary can easily invoke an input collision among two calls of AES encryption in the key schedule KeySch and tag generation TagGen, which implies an attack. The similar problem can happen for the processing of the last message block. Moreover it is not safe to produce L from an encryption of AES, E_K. From these observations, we think the introduction of secondary key L is adequate.

3.6 On Attack Reported at CRYPTO 2009

In CRYPTO 2009, Yuan et al. [31] reported a key-recovery attack against PC-MAC-AES$_{(128,d)}$ (with one difference that the underlying 4-round AES omits the last ShiftRows and MixColumns, following the suggestion of [23]). However, the attack needs $q = 2^{85.5}$ and 2^{128} computations, thus is totally impractical. Moreover such amount of queries is beyond the security bound we provided. Therefore the result of Yuan et al. does not contradict with our security proof. A novelty of Yuan et al. is that it shows the theoretical complexity of key-recovery rather than forgery. From the result of previous section it is strongly recommended to change the key before q or q_v reaches 2^{56}. As Yuan et al. pointed out, their result indicates that PC-MAC-AES's 256-bit key can be recovered with much less computation than 2^{256}. However, as mentioned in Section 3.5, it is well-expected since the role of L is to improve the efficiency (via implementing a tweakable block cipher), not to strengthen the security.

For reference, it is known that $q = 2^{64}$ is enough to produce a forgery against popular MACs including CMAC (e.g. [24]), by finding a collision of some 128-bit internal values (called birthday attack). As our PC-MAC-AES also has 128-bit data path, a similar attack is possible (see e.g. [18]). However, this is a common limitation to most MAC functions with 128-bit data path, as pointed out by, e.g., [18][16].

3.7 On Side-channel Attacks

A side-channel attack (SCA) exploiting a collision of internal variable was reported on alpha-MAC [12], a MAC function using 4-round AES. For PC-MAC-AES we are not aware of any report of SCA, however, it is natural to guess that an SCA against AES itself can also be applied to PC-MAC-AES. We think the security against SCA needs to be separately considered, even though it has provable security based on AES.

In general any countermeasure against SCA for AES, for instance ones from references [25][26][28][30][29] can be used for hardware implementations of PC-MAC-AES. As applications of them do not always assure the overall security of MAC, a careful investigation would be needed for their effectiveness.

4 Implementation Aspects

4.1 General Property

As PC-MAC-AES is based on AES and its 4-round function, its rough performance is estimated irrespective of the target platform. From reference [23], we obtain the Table 1. It shows that we obtain a speed up of about 1.4 times when $d = 1$ and 1.8 times when $d = 3$ from CMAC. The ratio of speed up approaches to 2.5 as d increases. However, we do not recommend to set d too large as the preprocessing time and memory linearly grow with d.

Table. 1 Comparison with CMAC-AES. Rounds denotes the mean AES rounds to process one message block, and Preproc denotes the number of AES encryptions for preprocessing.

MAC	Rounds	Preproc	Key size
PC-MAC-AES$_{(d)}$	$4 + \frac{6}{d+1}$	$4d - 1$	256
CMAC-AES	10	1	128

In the following we briefly describe implementations on software and hardware.

4.2 Software Implementation

It is basically easy to implement PC-MAC-AES on software if AES code is available. Alternatively, it is possible to use a library containing AES round function or a dedicated AES instruction, such as AES-NI [4]. AES-NI will be implemented on high-end Intel CPUs and will contain an instruction for the AES round function [27].

Here we implemented PC-MAC-AES via C language. The AES code is taken from a public domain code from [5] . For details of the code and development environment, see the specification of reference code [2]. The implementation result on a standard PC is as follows. For comparison we also implemented CMAC-AES.

- CPU : Intel Core Duo 1.66 GHz
- Operating system : Microsoft Windows XP Professional
- Programming language : ANSI C
- Compiler : Microsoft Visual C++ 2008 Express Edition
- Optimization : Optimize speed (/O2 option)

The above environment was used to evaluate the mean CPU cycles for processing 32 blocks (4096 bytes) with PC-MAC-AES$_{(d)}$ for $d = 1, \ldots, 5$. The evaluation was done for both key schedule and tag generation, and the mean was taken for 1024 trials for each d. The result is in Table. 2. Here Key schedule in the table includes the timing of AES key schedule, and the key schedule of CMAC also includes the timing for computing $E_K(0^{128})$.

Table. 2 Speed (cycles) of Reference Code.

MAC	Key schedule	Tag generation
PC-MAC-AES$_{(1)}$	1532	10158
PC-MAC-AES$_{(2)}$	3101	9204
PC-MAC-AES$_{(3)}$	4601	8803
PC-MAC-AES$_{(4)}$	6327	8614
PC-MAC-AES$_{(5)}$	7711	8396
CMAC-AES	759	12991

The object file of PC-MAC-AES was 60,715 bytes and that of CMAC-AES was 58,925 bytes, including the AES code, which was 46,176 bytes, for both.

We mention that our reference code aims at high readability and portablity. Thus it is not as fast as the implementation of [23]. For example, ours contains some redundant operations on data units, function configurations, and resolution of functionalities. Improvements in size and speed will be possible by some code optimizations.

4.3 Hardware Implementation

Hardware implementations of PC-MAC-AES can also be based on the existing hardware description language (HDL) of AES. Here, we implemented PC-MAC-AES$_{(1)}$ in Verilog-HDL.

The result is as follows. See the specification of reference hardware design [3] for the details.

■Development environment

- Target device : Virtex5 xc5vlx50-3
- Design tool : Xilinx ISE ver9.2i
- Synthesis option : Speed first, effort high (default for other options)

■Synthesis result

- Number of slices : 4356
- Maximum operating frequency : 128 MHz

We also compare our implementation with AES in ECB mode implemented on the same device. Our is about 1.37 times faster and about 1.4 times larger. As described by [3], the HDL of AES we used was rather focusing on ASIC. Hence we expect the improvement in size and speed is possible with a more FPGA-oriented HDL. In addition, as mentioned in Section 3.7, it is basically possible to embed an SCA countermeasure applicable to AES into our implementation.

References

[1] Specification of Cryptographic Technique PC-MAC-AES, NEC Corporation, 2010.
[2] Specification of Reference Code for PC-MAC-AES, NEC Corporation, 2010.
[3] Specification of Reference Hardware Design for PC-MAC-AES, NEC Corporation, 2010.
[4] Intel Advanced Encryption Standard (AES) Instructions Set - Rev 3, `http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set/`
[5] V. Rijmen, A. Bosselaers, and P. Barreto. Optimised ANSI C code for the Rijndael cipher ver.3.0.
[6] Tetsu Iwata. "On the security of block cipher mode of operation." CRYPTREC technical report, 2003. `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/documents/rep_ID0204.pdf` (in Japanese)
[7] NIST FIPS-197. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`
[8] NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. `http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf`
[9] B. den Boer, J.P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, M. Dichtl, W. Fumy, M. van der Ham, C.J.A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, and J. Vandewalle, *RIPE Integrity Primitives*, final report of RACE Integrity Primitives Evaluation. 1995.
[10] M. Bellare, O. Goldreich, and A. Mityagin. "The Power of Verification Queries in Message Authentication and Authenticated Encryption." *Cryptology ePrint Archive*, 2004/309.
[11] D. J. Bernstein. "The Poly1305-AES Message-Authentication Code." *Fast Software Encryption*, FSE'05, LNCS 3557, pp. 32-49, 2005.
[12] A. Biryukov, A. Bogdanov, D. Khovratovich, and T. Kasper. "Collision Attacks on AES-Based MAC: Alpha-MAC." *Cryptographic Hardware and Embedded Systems*- CHES '07, LNCS 4727, pp.166-180, 2007.
[13] J Daemen and V. Rijmen. "A New MAC Construction ALRED and a Specific Instance ALPHA-MAC." *Fast Software Encryption*, FSE'05, LNCS 3557, pp. 1-17, 2005.
[14] J Daemen and V. Rijmen. "The Pelican MAC Function." *IACR ePrint Archive*, 2005/088.
[15] S. Halevi and H. Krawczyk. "MMH:Software Message Authentication in the Gbit/second rates." *Fast Software Encryption*, FSE'97, LNCS 1267, pp. 172-189, 1997.
[16] T. Iwata. "Comments on " On the security of XCBC, TMAC and OMAC " by Mitchell." `csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/.../Iwata3.pdf`
[17] T. Iwata and K. Kurosawa. "OMAC: One-Key CBC MAC." *Fast Software Encryption*- FSE'03, LNCS 2887, pp. 129-153, 2003.
[18] K. Jia, X. Wang, Z. Yuan, and G. Xu. "Distinguishing Attack and Second-Preimage Attack on the CBC-like MACs." *Cryptology ePrint Archive*, Report 2008/542.
[19] L. Keliher and J. Sui. "Exact Maximum Expected Differential and Linear Probability for 2-Round Advanced Encryption Standard (AES)." *IACR ePrint Archive*, 2005/321.
[20] L. Keliher and J. Sui. "Exact maximum expected differential and linear cryptanalysis for two-round Advanced Encryption Standard." *IET Information Security*, Vol. 1, No. 2, pp. 53-57, June. 2007.
[21] K. Kurosawa and T. Iwata. "TMAC: Two-Key CBC MAC." *Topics in Cryptology*- CT-RSA 2003,

LNCS 2612, pp. 33-49, 2003.
[22] M. Liskov, R. Rivest, and D. Wagner. "Tweakable Block Ciphers." *Advances in Cryptology-*CRYPTO'02, LNCS 2442, pp. 31-46, 2002.
[23] K. Minematsu and Y. Tsunoo. "Provably Secure MACs From Differentially-uniform Permutations and AES-based Implementations." *Fast Software Encryption*, FSE '06, LNCS 4047, pp. 226-241, 2006.
[24] C.J. Mitchell. "On the security of XCBC, TMAC and OMAC." *Technical Report*, RHUL-MA-2003-4, 19, 2003. `http://www.rhul.ac.uk/mathematics/techreports`
[25] S. Nikova and C. Rechberger, and V. Rijmen, " Threshold Implementations Against Side-Channel Attacks and Glitches, " The 8th International Conference on Information and Communications Security (ICICS 2006), LNCS4307, pp. 529-545, Springer-Verlag, Dec. 2006.
[26] T. Pop and S. Mangard, " Masked Dual-Rail Pre-charge Logic: DPA-Resistance without Routing Constrain, " Workshop on Cryptographic Hardwar and Embedded Systems (CHES2005), LNCS 3659, pp. 172-186, Springer-Verlag, Aug. 2005.
[27] S. Gueron. "Intel's New AES Instructions for Enhanced Performance and Security.", *Fast Software Encryption*, FSE '09, LNCS 5665, pp. 51-66, 2006.
[28] D. Suzuki, M. Saeki, and T. Ichikawa, " Random Switching Logic: A New Countermeasure against DPA and Second-Order DPA at the Logic Level, " IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E90-A, no. 1, pp. 160-168, Jan. 2007.
[29] E. Trichina, " Combinational Logic Design for AES SubByte Transformation On masked Data, " Cryptology ePrint Archive, 2003/236, 2003.
[30] K. Tiri and I. Verbauwhede, " A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation, " Proc. 2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), pp. 246-251, Feb. 2004.
[31] Z. Yuan, W. Wang, K. Jia, G. Xu, X. Wang: "New Birthday Attacks on Some MACs Based on Block Ciphers." *Advances in Cryptology*, CRYPTO '09, LNCS 5677, pp. 209-230.