Implementation of AES Using NVM Memories Based on Comparison Function

Jérémie Clément, Bruno Mussard, David Naccache, Lionel Torres

Implementation of AES Using NVM Memories Based on Comparison Function

Jeremie CLEMENT, Bruno MUSSARD
Crocus Technology
Rousset, France
Email: {jclement, bmussard}@crocus-technology.com

David NACCACHE
Ecole Normale Supérieure
Paris, France
Email: david.naccache@ens.fr

Lionel TORRES
LIRMM, UMR CNRS
Université de Montpellier, France
Email: lionel.torres@lirmm.fr

***Abstract*—Nowadays, cryptography is widely used in many different devices (smartphones, tablets, computers, etc.). One of the most used cryptographic algorithms is AES (Advanced Encryption Standard). Crocus Technology is developing a technology called Match-In-Place^{TM}(MIP). This is a memory based on comparison function. With this technology, data stored in the memory can be compared to an input data. The main expectation of this technology is that, during this comparison, no information leaks from the memory, which is interesting against side channel attacks. This assumption has been verified by simulation but not in practice yet. This paper proposes a new secure AES implementation based on the MIP technology. The use of this particular technology allows to protect the key in a secure environment to prevent attackers from retrieving it. MIP also allows a very low silicon area implementation of AES.**

***Keywords*-Advanced Encryption Standard (AES), MRAM, cryptography, security, memory**

I. Introduction

In 2001, the National Institute of Standards and Technology (NIST) specified the Advanced Encryption Standard (AES). AES is a symmetric block cipher designed by Daemen and Rijmen in [1]. Nowadays, AES is widely used in a variety of applications and devices. So, many works were performed to secure AES against side-channel attacks ([2], [3], [4]).

Crocus Technology is a semiconductor company developing Magnetoresistive Random-Access Memories (MRAM). They are currently developing a technology called Match-In-Place (MIP). MIP is a memory which has an embedded comparison function. It consists in comparing an input data, given to the memory to a secret data stored in the memory. The comparison operation itself takes place in the memory. The advantage of MIP is that the sensitive data never leaves the memory, so it is impossible for an attacker to intercept this data during the comparison operation. Each MIP cell is a non-volatile memory cell combined with a virtual XOR gate. Multiple cells can be connected in series to form a NAND chain acting as a linear MIP engine. This technology can be used to secure user authentication (for example by comparing passwords in a secure way). As all operations take place in the memory, there is no data leakage, so it is impossible to retrieve sensitive data (such as a key or a password) using power analysis. Although this security features have been proved by simulation, these assumptions still have to be verified on actual MIP chips.

In this paper, a secure AES implementation is described, using MIP to secure the key. In this implementation, MIP cells are used in different ways :

- As a Content Addressable Memory (CAM) to replace AES S-Boxes;
- As a bitwise XOR by processing bit-to-bit matching operation;

By integrating MIP into AES operations, the key and the processed intermediate data during the different operations are secured. This algorithm was implemented in Verilog and simulated using the Icarus Verilog simulator.

The rest of this paper is organized as follows. Section 2 recalls AES operations. Section 3 describes the MIP technology. Section 4 explains how MIP is connected into AES algorithm. Section 5 shows the simulation results of the proposed implementation. Section 6 concludes.

II. AES Operation

AES encryption and decryption are based on four different transformations which are repeated N_r times. AES has a fixed 128 bits data block and a key size of 128, 192 or 256 bits. For the sake of simplicity, only the 128-bit key variant was implemented, so the rest of this paper only refers to AES-128. For this variant the number of rounds is $N_r = 10$. Fig. 1 describes the flow of operations for AES encryption. Encryption and decryption flows are slightly different as, compared to encryption, decryption executes the reverse transformations in the reverse order.

AES considers a 128-bit block data as a 4×4 byte state. In parallel to encryption or decryption, the 128-bit key is processed by a key schedule operation to generate N_r round subkeys. Each round subkey and the cipher key are considered as a 4×4 byte state. A full round is decomposed into four steps.

A. The SubBytes step

In the SubBytes step, each byte $a_{i,j}$ in the state matrix is replaced with another byte $S(a_{i,j})$ using an 8-bit substitution box (called S-box). This operation provides the non-linearity of the cipher. Many works on improving S-Boxes have been realized. Composite field S-Boxes are widely used, as they

978-1-4799-8719-1/15 $31.00 © 2015 IEEE
DOI 10.1109/ISVLSI.2015.37

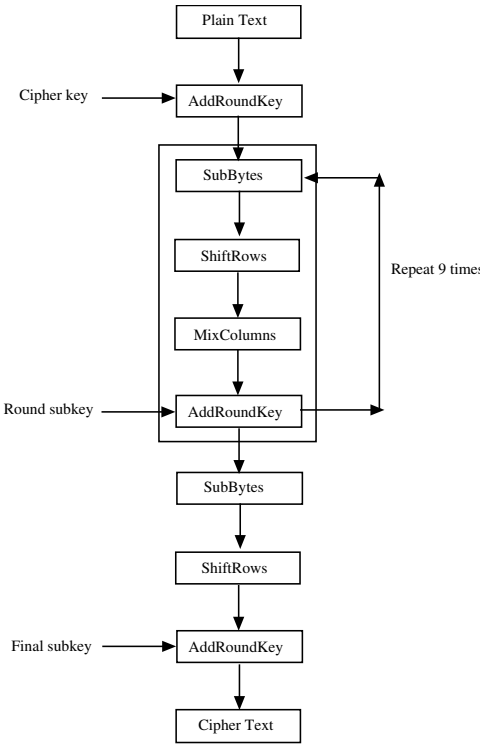

Figure 1. AES encryption flow

allow to minimize the gate count and improve performances on S-Box calculation. Many works on composite fields focus on the tower field $\mathbb{F}((2^2)^2)^2$ ([5], [6], [7]).

B. The ShiftRows step

The ShiftRows step cyclically shifts the bytes in each row of the state by a certain offset. The first row of the state is left unchanged. Then, the row n is shifted left circularly by $n-1$ bytes. For decryption, the bytes are shifted circularly the same number of positions but to the right.

C. The MixColumns step

In this step, each column of the 4×4 byte state is treated as a polynomial over $GF(2^8)$. It is then multiplied with the polynomial $c(x) = 3x^3+x^2+x+2$ modulo x^4+1. Usually, the MixColumns step is merged with the ShiftRows step, as these two steps provide diffusion of the cipher. This step is not present in the final round.

D. The AddRoundKey step

In this step, each byte of the state is simply XORed (exclusive OR operation) with the corresponding byte of the current round subkey.

E. The key schedule operation

Executed in parallel to the main AES operation, this operation derives the main key into N_r round subkeys. This operation creates a 176 byte array called the expanded key. Several operations are performed to generate 4-byte words W_i where $i > N_k$ and N_k is the number of 4-byte words in the cipher key, $N_k = 4$ for a 128-bit key:

- $W_i = K_i$ for $0 \leq i < N_k$, where K_i is the i^{th} 4-byte word from the cipher key,
- $W_i = W_{i-1} \oplus W_{i-N_k}$ for $i > N_k$ and i is not a multiple of N_k,
- If i is a multiple of N_k, the word W_{i-1} is first rotated circularly 8 bits to the left. Then it is transformed using the S-Box. Finally the leftmost byte of the result is XORed with a round-dependent constant $R_{con} = 2^{r-1}$ in $GF(2^8)$, where r is the current round number.

Once all the expanded key array is filled, it is divided into $N_r + 1$ 128-bit round subkeys. In the 128-bit key version of AES, 11 round subkeys are generated in this way, the first round subkey being the cipher key.
To improve the security of this algorithm, a technology called MIP was used.

III. The Match-In-Place technology

A. Description of MIP technology

MIP [8] is a device developed by Crocus Technology, based on its Magnetic Logic Unit (MLU) [9], to authenticate users without exposing any confidential data to external attackers. The MLU technology is an evolution of the Thermally Assisted Switching MRAM described in [10]. In a TAS-MRAM, each Magnetic Tunnel Junction (MTJ) is composed of a storage layer and a fixed reference layer. In the MLU, the fixed reference layer of the MTJs is replaced with a ferromagnetic free layer, referred as the sense layer. By doing so, this self-reference MRAM has the particularity to perform logic comparisons. Considering the storage layer and the sense layer as two different inputs, it can perform a XOR operation by measuring the resistance of the memory cells. By connecting the self-referenced MRAM cells to form a NAND chain, it is possible to create a MIP engine, comparing the data stored into the memory, which is a sensitive data (passwords for example) and the data given as input. A mismatch between the two data results in a higher resistance than a match. Fig. 2 shows an example of a MIP engine composed of four MLU cells connected as a NAND chain. In this example, the stored data is "1010" and the input data is "1100". As the stored data and the input data

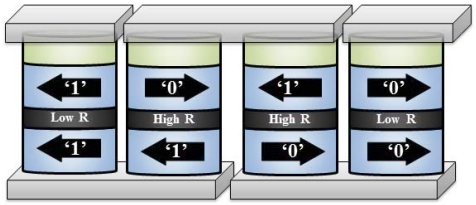

Figure 2. MIP composed of 4 chained MLU cells

are different, the total resistance of the chain will be higher than the minimum resistance. If the input data match the stored data, all the cells would have a low resistance, then the total resistance of the chain would be minimal.
For AES implementation, instead of matching, two modified applications of the MIP were used. First it was used to perform a XOR operation between two data words. MIP XOR blocks contain 32 MTJs. Each MTJ performs a matching operation on one bit, and is not connected to the other MTJs of the block. In the end of the matching operation, 32 resistance values are obtained. Each value is then interpreted as bits, '0' for a low resistance (if the stored data and the input data are identical), '1' for a high resistance (if the stored data and the input data are different). So, a 32-bit value is obtained, result of the XOR operation between the stored data word and the input data word. Four of these blocks are used for the AddRoundKey operation (which performs a XOR operation between a 128-bit key and a 128-bit data). Four other blocks are used for the MixColumns operation, and another one for the key schedule operation.
The MIP can also be used as a fast associative memory like a Content Addressable Memory (CAM) used in routers and search engines for example. By placing many MIP engines in parallel, it makes the search faster. The input data is compared to all the entries of the CAM, handled by MIP. All the comparisons are made in parallel. When this is done, only one engine will return a perfect match notified by the chain which has the minimum resistance. With this the incoming packet can be directly routed to the recipient address. Fig.3 describes an example of CAM using MIP technology. In this example, there are 4 MIP engines. The input pattern "1010" is then compared to the memory. For AES implementation, two CAM blocks are used :

- The SBox block : It contains 2048 MTJs divided into 256 engines, each one composed of 8 MTJs. This block is used for the SubBytes step and the key schedule operation. The input of this block is the current byte processed from the state.
- The R_{con} block : It is used to determine the round constant used in the key schedule. It contains 10 MIP engines. Each engine is a NAND chain of 8 MTJs. In each engine is stored a possible round number (from 1 to 10). The input of each of these MIP engines is the current round number.

This technology has two advantages. First, the comparison operation is performed into the memory. The secret data stored into the memory never leaves it during this operation. The second advantage is that it provides a protection against side-channel attacks.

B. Side channel attack on MIP technology

In order to demonstrate the security features provided by MIP, a Differential Power Analysis (DPA) have been simulated based on the MIP netlists. This attack focused

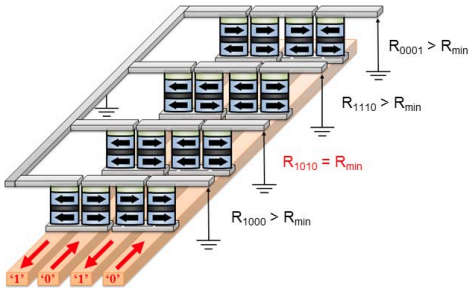

Figure 3. Example of a CAM using MIP

on the matching of two 32-bit words using MIP. The MIP matching operation was simulated using Spectre in order to obtain simulated power traces. Using these traces, a DPA attack was conducted in order to try to retrieve the stored key. To perform this attack, a random 8-bit key has been stored in MIP memory. This key is unknown to the attacker. The size of the key (8-bits) has been chosen because it fits the size of the matching used for the S-Boxes. Then all the possibilities for 8-bits were proposed as input of the MIP. Only one of these inputs results as a match. The attack has then been realized using these 256 power traces. But, whatever the input of the MIP operation was, the power traces were all similar, which led the attack to fail. Thus, if a DPA attack fails on a 32-bit matching operation performed by MIP, the proposed AES implementation can be also considered secure against these attacks as it uses the same principles. The MIP technology is still in development. Differential Fault Analysis (DFA) attacks have not been tested on MIP as no functional test chips are available yet.

IV. Integrating MIP in AES implementation

In order to make use of the protection offered by MIP against side channel attacks, AES algorithm was implemented using this technology. The proposed implementation was made using the Verilog Hardware Description Language. At every step of the algorithm, the integration of MIP provides additional security features, preventing an attacker from reading the cipher key or the internal state. Another advantage is that these operations can be executed directly into the MIP memory. In other words, the cipher key and data (plaintext or ciphertext) are stored into the memory, which triggers AES operation. Once it is finished, the memory outputs the result of the operation.

A. The AddRoundKey step : A secure Xor operation

The AddRoundKey step is just a Xor between two 128 bits values (the key and the data). To do this, 4 blocks of MIP XOR are used. Each block is composed of 32 bit cells. Each cell is not connected to the others and acts as a XOR gate. The round subkey is stored into the MIP memory as soon as it is generated, so it is kept safe during the entire operation. Then, the data is given to the blocks. For each block, the 32

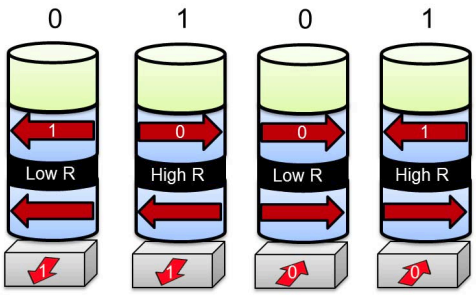

Figure 4. Example of 4-bit XOR operation managed by MIP

bits will check separately if the given data bit match with the corresponding key bit. These 32 operations are made in parallel. At the end of this operation, 32 resistance values were obtained, each value is then interpreted as bits, '0' for a low resistance (if the key and the data are identical), '1' for a high resistance (if the key and the data are different). So a 32-bit match value per block is obtained, each bit of this value corresponding to each 1-bit MIP operation. This match value gives the result of a Xor operation between the key and the data. Fig. 4 shows the XOR operation between two 4-bit data. The stored data ('1001') is Xored with an input data ('1100'). All the 1-bit matching operation are processed in parallel and in the end, the result of the XOR operation is obtained ('0101').

B. The SubBytes step : The S-Box

In classical AES implementations, there are two ways to calculate the output of the S-Box based on the inputs :

- Computation on-the-fly of the output value
- Use of a look-up table

For the proposed implementation, the look-up table approach is chosen. Here the MIP SBox block is used. At the beginning of AES operation, all the MIP NAND chains are programmed with a single value, corresponding to the different possible input values for the S-Box (from $0x01$ to $0xFF$). Then when the SubBytes operation starts, all the NAND chains receive the same input byte which is the input of the S-Box. Thus only one MIP engine will result as a match between the the input of the S-Box and their corresponding unique data. According to the number of the matching instance and if it is an encryption of a decryption process (due to the differences between the encryption and decryption look-up tables), the output of the S-Box will be different. In the proposed implementation the ShiftRows step is merged with the SubBytes step. Indeed all the output values of the S-Boxes, are written at their new location in the state, according to the ShiftRows step.

C. The MixColumns step : Make Multiplications with Xors

This step consists in a polynomial multiplication of each column of the internal state. But it can also be expressed as the product between a fixed matrix and a column of the state. The positions of the bytes into the state are shifted according to the ShiftRows step described earlier. Hence, the new positions of the bytes in the state are taken into account to determine the inputs of the MixColumns step.

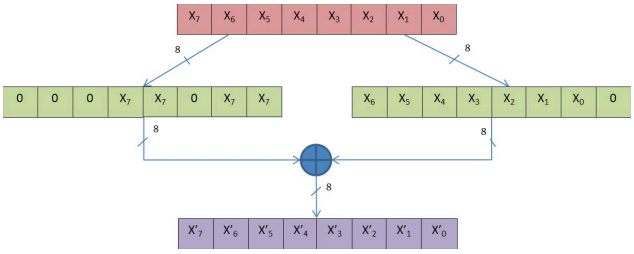

Figure 5. Multiplication by two using only XOR operations.

Here is described the matrix multiplication used for the MixColumns step for the encryption process. A,B,C,D are the input bytes of the current column of the state and Y_1, Y_2, Y_3, Y_4 are the output bytes, constituting the new state column.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} Y_1 = 2A \oplus 3B \oplus C \oplus D \\ Y_2 = A \oplus 2B \oplus 3C \oplus D \\ Y_3 = A \oplus B \oplus 2C \oplus 3D \\ Y_4 = 3A \oplus B \oplus C \oplus 2D \end{bmatrix}$$

Here, the XOR operations are made with the MIP by using the same method than in the AddRoundKey step, in order to preserve the inputs corresponding to the state from attackers. To do the multiplication by 2, a technique that only involves XORs is used as described in Fig. 5. In this figure the input (X) and the output (X') are bytes with $X = X_7X_6X_5X_4X_3X_2X_1X_0$ and $X' = 2 * X$. As all the operations are made in $GF(2^8)$, the multiplication by 3 is just a multiplication by two combined with a XOR.

For the proposed implementation, four 32-bit XOR blocks of MIP are used. Each of these blocks performs the MixColumns operation on a different column of the state and are executed in parallel.

There are two interests of doing that. The first one is the security of the state, which will be protected against side-channel attacks during this step. The second interest is that this entire step can be implemented into a memory, which means that one just need to give the state data to the memory and it outputs the new state.

D. The key schedule operation: Different uses of MIP

For the key schedule operation, many different MIP blocks are used:

- The main S-Box is reused as for the SubBytes step described earlier.
- Another S-Box on the same principle than the SubBytes S-Box is used. This one is called the R_{con} S-Box and is used to determine the round constants. This S-Box takes

as input the number of the current round, and outputs the current round constant. There are 10 entries for this S-Box, so 10 blocks of 8 chained-bits of MIP are used to handle the inputs.
- A MIP block is used for the bitwise XOR operations by having 32 bit cells, each one unconnected with the others, as described earlier.

In this implementation, the key schedule operation executes in parallel of the main AES operation. At each round of AES operation, the next round subkey is generated.
Integrating MIP into this operation provides an additional security feature, by preventing an attacker from extracting a key during this operation. Another advantage of using MIP for logical operations is that the entire key schedule operation can be made by the MIP memory. To do this one just need to store into the memory the current round subkey and the round number. Once the operation is finished, it outputs the new round subkey at a predefined address into the memory. This address can be reused in the AddRoundKey step, the round subkey being stored into a MIP memory, allowing to perform the XOR with the data without the subkey being read by the processor and extracted from the memory.

V. Results and future works

A. Simulation Results

This architecture has been implemented in Verilog. A Known Answer Test (KAT) vector has been used in order to verify that the proposed implementation gives the same result than a classical AES implementation.
Compared to classical Verilog AES implementations, the described method protects the cipher key and the S-Box from being collected by an attacker, due to the protection provided by MIP. Besides, the main operations of this implementation only uses the characteristics of the Magnetic Logic Unit technology used by the MIP and can be entirely executed into the memory, the processor is just interfacing with the MIP memory without executing these operations. In return, there is a lack of performances compared to other AES implementations.

Table I describes the area and the delay taken by the different steps of the proposed implementation. The delay is the execution time for one round of the algorithm. Due to the fact that all these operations are executed directly into the MIP memory, the area expressed in this table is an estimation based on the number of MTJs used for each step.

The proposed implementation only takes into account the AES-128 variant.In the cases of AES-192 and AES-256 variants, the needed area would be increased due to the increase on the key size, although performances decrease due to the fact that the key schedule has more steps than in the AES-128. However, the other steps of the algorithm are unchanged, the only difference would be in the key schedule part.

Table I
Estimations on area and delay for each AES step

Step	Number of MTJ	Area (μm^2)	Delay per round (ns)
AddRoundKey	128	3.7	100
SubBytes	2048	78.7	2410
MixColumns	512	14.74	780
KeySchedule	112	3.9	1200
Total	**2416**	**101**	**4490**

Table II
Comparison of AES implementations (*: Estimations)

Implementation	Throughput (Mbps)	Area (mm^2)	Energy ($\mu W/MHz$)
This paper	3.2	0.01*	261*
Verbauwhede et al. [11]	1600	3.96	432
Hwang et al. [12]	3840	2.45	4000
Hodjat et al. [13]	3840	0.79	1080
Sumanth et al. [14]	32820	0.74	534.30

Table II compares the proposed implementation with other AES implementations on throughput, area and energy. The values for the area and energy consumption for the implementation proposed in this paper are estimated values. The area presented in this table is the total area estimation of the proposed implementation. As shown in Table II, the proposed implementation suffers from a lack of performances compared to other implementations, but compensates it by having a very low area. The energy consumption estimation is realized using the technique described in [15]. This technique calculates the energy spent by a system during the execution of an algorithm using the following formula : $E = I * V_{DD} * N * \tau$, where E is the energy consumed by the implementation in Joules, I is the average current consumption per clock cycle, V_{DD} is the operating voltage of the MIP, N is the number of clock cycles taken for the execution of this algorithm and τ is the clock period. According to this method, the proposed implementation spends 1.440 μJ with a clock frequency at 40 MHz. When E is obtained, it is divided by the execution time in order to get the power consumption in W. In order to have a more accurate comparison with the other implementations, the power consumption in Table II is expressed in $\mu W/MHz$.

B. Future works

One of the possible future works will be making a compromise between the loss of performances and the gain of security. To do this, the use of MIP will be restrained to only sensitive operations, the AddRoundKey and the S-Box. The other AES operations will be executed by the processor. The last step will be to test this implementation on actual MIP chips.

VI. Conclusion

In this paper a new technology called Match-In-Place (MIP), which will be used in secure functions, is introduced. This technology, which is an evolution of MRAM memories, allows secure matching operations such as password comparison, to be directly executed into a secure memory. This way, sensitive data are never exposed to an external attacker. A simulated DPA attack has been performed on MIP and failed, proving that MIP is secure. But this assumption was only observed on simulated power traces and is still to be proven on actual MIP chips, which are not available yet. MIP can also be used to secure look-up tables searching operations and to execute a secure XOR operation. Here, this technology has been integrated into a well-known and very used cryptographic algorithm, AES. This algorithm was implemented in Verilog and compared to a classical implementation. The described method has the advantage to perform the operations directly into the memory. Also once the secret key is stored into the memory, it is locked up and never leaves it, preserving it from eavesdropping. This method is also low power and low area, based on estimations. But the main drawback of this method is that it suffers from a lack of performances, compared to classical Verilog implementations of AES.

This work is not finished yet, and many improvements can be done on this implementation. Then it is possible to reduce the use of MIP to critical functions in order to make a compromise between the performance loss and the security. Critical functions would be the AddRoundKey step, because the key is directly used and combined to the message, and the main S-Box, which introduces the confusion in the algorithm. Finally, when actual MIP chips are functional, tests must be made in order to validate the results of the simulated DPA attack.

References

[1] J. Daemen and V. Rijmen, "Aes proposal: Rijndael," AES submission, 1999.

[2] M.-L. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," in *Cryptographic Hardware and Embedded Systems CHES 2001*, ser. Lecture Notes in Computer Science, vol. 2162. Springer Berlin / Heidelberg, 2001, pp. 309–318.

[3] R. K., K. W., P. M., and Y. K., "Fault-based side-channel cryptanalysis tolerant rijndael symmetric block cipher architecture," in *16th IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2001)*, 2001, pp. 427–435.

[4] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 492–505, April 2003.

[5] D. Canright, "A very compact s-box for aes," in *Cryptographic Hardware and Embedded Systems CHES 2005*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3659, pp. 441–455.

[6] X. Zhang and K. Parhi, "On the optimum constructions of composite field for the aes algorithm," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53, no. 10, pp. 1153–1157, Oct 2006.

[7] Y. Nogami, K. Nekado, T. Toyota, N. Hongo, and Y. Morikawa, "Mixed bases for efficient inversion in $f((2^2)^2)^2$ and conversion matrices of subbytes of aes," *IEICE Transactions*, vol. 94-A, no. 6, pp. 1318–1327, 2011.

[8] B. Cambou, "Match-in-place$^{\mathrm{TM}}$, a novel way to perform secure and fast user authentication," White Paper, 2012.

[9] Q. Stainer, L. Lombard, K. Mackay, R. C. Sous, I. L. Prejbeanu, and B. Dieny, "Mram with soft reference layer: In-stack combination of memory and logic functions," IEEE International Memory Workshop (IMW), pp. 84–87, 2013.

[10] I. L. Prejbeanu, M. Kerekes, R. C. Sousa, H. Sibuet, O. Redon, B. Dieny, and J. P. Nozires, "Thermally assisted mram," *Journal of Physics: Condensed Matter*, vol. 19, no. 16, p. 165218, 2007.

[11] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and performance testing of a 2.29 gb/s rijndael processor," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 3, pp. 569–572, 2003.

[12] D. Hwang, K. Tiri, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "AES-based security coprocessor IC in 0.18-μm CMOS with resistance to differential power analysis side-channel attacks," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 4, pp. 781–792, Apr 2006.

[13] A. Hodjat, D. Hwang, B. Lai, K. Tiri, and I. Verbauwhede, "A 3.84 gbits/s aes crypto coprocessor with modes of operation in a 0.18um cmos technology," in *Proceedings of the 15th ACM Great Lakes symposium on VLSI (GLSVLSI 2005)*, 2005, pp. 60–63.

[14] S. Sumanth Kumar Redy, R. Sakthivel, and P. Praneeth, "Vlsi implementation of aes crypto processor for high throughput," *International Journal of Advanced Engineering Sciences And Technologies (IJAEST)*, vol. 6, no. 1, pp. 22–26, 2011.

[15] K. Naik and D. S. L. Wei, "Software implementation strategies for power-conscious systems," *Mob. Netw. Appl.*, vol. 6, no. 3, pp. 291–305, Jun. 2001.